UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Plataforma educativa para la asignatura de Programación Orientada a Objetos

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero en computación

P R E S E N T A

Juan Carlos Del Camino Rojas

ASESOR DE MATERIAL DIDÁCTICO

Ing. Jorge Alberto Solano Gálvez

Ciudad Universitaria, Cd. Mx., 2019

Índice

Agradecimientos

Este trabajo no hubiese sido posible sin el apoyo del Ingeniero Jorge Alberto Solano Gálvez, gracias por su apoyo, confianza y orientación al momento de realizar este proyecto.

Agradezco a los comentarios realizados por el ingeniero Aurelio Adolfo Millan Najera los cuales permitieron que el material presentado fuese de la mejor calidad posible, sus comentarios fueron pieza clave para la creación de este documento.

Al Doctor Eduardo Espinoza, no solamente por sus comentarios en la revisión de este documento, si no por las enseñanzas brindadas cuando tuve la grata experiencia de tomar una de sus clases, por mostrarme lo bella que puede llegar a ser esta carrera, muchas gracias.

A la Ingeniera Tanya Itzel Arteaga Ricci por ser un ejemplo para seguir durante toda mi carrera y siempre ser un referente personal sobre lo que significa ser un gran docente, por todos los apoyos brindados y enseñanzas, muchas gracias.

Quiero agradecer a mi madre y hermana por su apoyo incondicional desde que inicié la carrera hasta esta ultima etapa, su apoyo ha sido fundamental a través de todo este proceso para llegar hasta este punto, sin ustedes esto jamás hubiese sucedido.

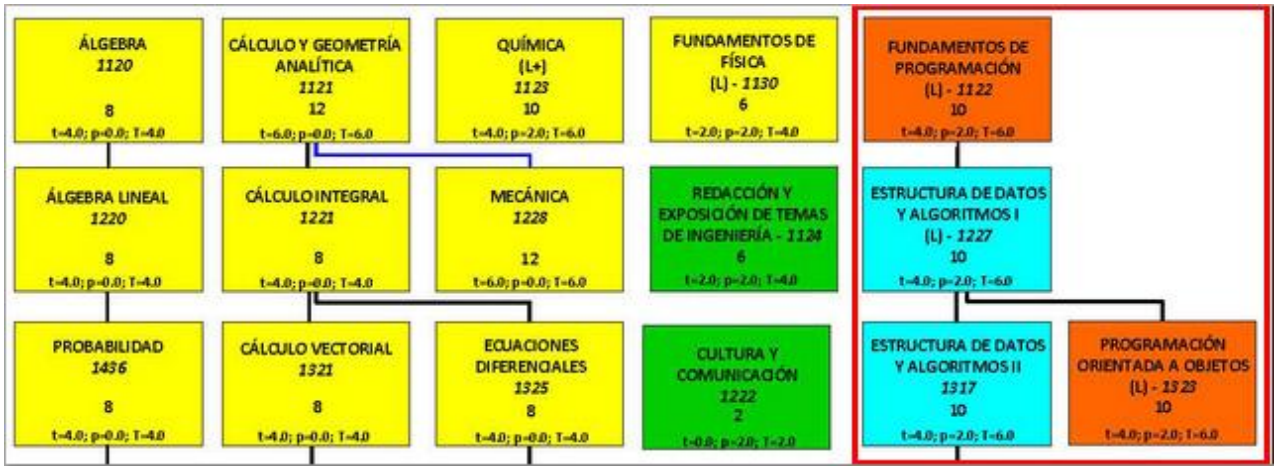
Finalmente a mi padre que a pesar de no estar más aquí, fue clave para mi formación académica y personal, sus enseñanzas serán clave para el futuro, espero llenarlo de orgullo, de el aprendí a ser fuerte incluso en las situaciones mas adversas y que toda situación tiene una solución, solo falta encontrarla.

1.- Objetivo de las actividades.

1.1 Antecedentes

En agosto de 2015 entraron en vigor en la Facultad de Ingeniería nuevos planes y programas de estudio. Entre las carreras involucradas en el cambio estuvo Ingeniería en Computación.

El plan de estudios 2016 de la carrera de Ingeniería en Computación recibió un impulso base importante, al tener un ‘core’ de 4 asignaturas afines a la carrera, las cuales se toman desde el primer semestre:



La asignatura Programación Orientada a Objetos (POO) se encuentra en el 3er semestre de la carrera de IC. En esta asignatura se imparte el paradigma orientado a objetos y permite conocer a fondo la teoría del paradigma con su aplicación utilizando un lenguaje de programación.

1.2 Objetivos y alcance

1.2.1 Objetivos

* Crear material didáctico-pedagógico para apoyar a la asignatura de Programación orientada objetos, tanto en la teoría como en el laboratorio, con información teórica sobre el temario de la asignatura, información teórica sobre las prácticas de la asignatura, ejercicios y exámenes, en una plataforma en línea disponible para los alumnos que estén inscritos en la asignatura.

* Crear una plataforma en línea que permita a los académicos que imparten la asignatura de Programación orientada a objetos controlar las calificaciones de los alumnos inscritos,

subir documentos, tareas y exámenes a la plataforma, así como aprovechar los recursos didácticos-pedagógicos contenidos en la plataforma, como un apoyo a su clase.

1.2.2 Alcance

El proyecto tiene como alcance la generación de material didáctico-pedagógico, abarca desde la generación de material teórico-práctico de apoyo a la asignatura, que pueda ser utilizado por miembros de la UNAM (alumnos, profesores-académicos, personal del laboratorio, etcétera), hasta la evaluación y control de calificaciones de las prácticas determinadas por la academia de la asignatura de Programación orientada a objetos de cada alumno inscrito a la asignatura.

2.- Definición del problema

2.1 Descripción del problema

La asignatura de Programación orientada a objetos (L) es una materia teórico-práctica. En el plan de estudios tiene asignadas 4 horas teóricas y 2 horas prácticas, por lo que al semestre se imparten 64 horas teóricas y 32 prácticas. Esta asignatura se imparte en los laboratorios de computación salas A y B, los cuales están certificados bajo la norma ISO 9001:2015.

Debido a la cantidad de horas de la asignatura, es posible profundizar el conocimiento de un lenguaje de programación orientado a objetos, para ello se requiere una plataforma que esté alineada al temario de la asignatura y que contenga el material necesario para entender y desarrollar programas con diferentes niveles de profundización que permitan afianzar las bases teóricas del paradigma y generar programas robustos y útiles para la carrera y para la vida profesional.

Además, dentro de los estándares de calidad de los laboratorios, está en constante evaluación el desarrollo de la práctica y el cumplimiento de los objetivos por práctica, para generar el aprendizaje deseado en los estudiantes. Estas evaluaciones y el control de calificaciones se llevarán a cabo en la plataforma, de tal manera que estén automatizadas las encuestas y sus estadísticas y se tenga un resguardo de las calificaciones de los alumnos por semestre.

3.- Metodología

3.1 Justificación

Los laboratorios de la asignatura de Programación orientada a objetos se encuentran certificados bajo la norma ISO 9001:2015, lo cual implica que se debe mantener un Sistema de Gestión de Calidad (SGC) con retroalimentación continua por parte de los miembros involucrados (alumnos, profesores, personal del laboratorio, etcétera) lo cual asegura la calidad de la asignatura y permite la mejora continua del mismo.

3.2 Forma de trabajo

Debido a que la asignatura es teórico-práctica, se contempla la generación de material teórico y se planea la realización de ejercicios prácticos que afiancen el conocimiento teórico adquirido.

Las acciones que se pueden realizar dentro de la plataforma dependen del rol asignado al usuario que la esté utilizando, los roles existentes son:

- Administrador: Es el rol encargado de crear grupos, controlar alta, baja y modificación de usuarios además de la creación de ejercicios y manejo del catálogo de preguntas de examen.
- Alumno: Es el rol que tomarán los alumnos inscritos a la asignatura, este rol permite pertenecer a un grupo de la asignatura, entregar tareas, realizar prácticas, exámenes y ver las evaluaciones de estos rubros.
- Profesor: Este rol se le asigna a un usuario que vaya a hacer las labores de docente, permite asignar los porcentajes de evaluación para cada rubro utilizado, añadir tareas al grupo y visualizar las calificaciones de los alumnos.

3.2.1 Teoría

La plataforma tendrá en línea todas las bases teóricas del paradigma orientado a objetos y las bases teóricas del lenguaje orientado a objetos utilizado por la academia, cumpliendo con cada uno de los temas y subtemas de la asignatura. Al final de cada unidad se hará una evaluación de realimentación sobre los conceptos tratados.

3.2.2. Laboratorio

La plataforma tendrá las guías prácticas de la asignatura para que cualquier persona pueda revisar su contenido. Sin embargo, para evaluar las prácticas se tienen dos esquemas diferentes:

- Usuarios de consulta (usuarios anónimos): Estos usuarios tendrán acceso a ejercicios que no generan calificación. Dichos ejercicios que se pueden repetir un

número indefinido de veces y constan de preguntas teóricas de opción múltiple con retroalimentación por cada respuesta para indicar al que lo realiza por que está ó no en lo correcto, no se requiere estar registrado a un grupo activo para realizar este tipo de actividades ya que son públicas y no almacenan ningún tipo de calificación ó estado en el sistema.

- Usuarios inscritos en la asignatura (usuarios registrados): Estos usuarios tendrán acceso a ejercicios que generan calificación. Estos ejercicios estarán únicamente disponibles para alumnos inscritos en un grupo activo de la plataforma (registrados por el administrador), estos ejercicios generan una calificación al usuario que lo realice por lo cual almacena los resultados al sistema y no están disponibles a un usuario de consulta (usuario anónimo).

Además, se realizarán exámenes para los alumnos que pertenezcan a un grupo y así poder evaluar sus conocimientos, dichos exámenes serán proporcionados por los profesores y el sistema gestionará los resultados de cada alumno.

Al finalizar cada una de las prácticas se aplicará una encuesta al alumno donde podrá evaluar si la práctica cumple el objetivo planteado, esto permite que se tenga una retroalimentación por parte del alumno para una mejora continua del material utilizado, el cual es un requisito del SGC.

Al terminar el curso práctico se aplicará una nueva encuesta al alumno para que evalué la calidad del curso completo (el cual también es otro requisito del SGC), las respuestas se utilizarán para realizar cambios al material en caso de ser necesario. Finalmente, el sistema evaluará los resultados del alumno en los exámenes, prácticas y ejercicios realizados y determinará su promedio basado en los porcentajes de los rubros que el profesor del grupo haya determinado y ésta será la calificación del curso del alumno.

4.- Descripción del material didáctico

Se realizará una plataforma utilizando Django, un framework basado en el lenguaje de programación Python que permite el desarrollo de aplicaciones web, en la cual se colocarán las prácticas de la coordinación de la materia de programación orientada a objetos y los ejercicios correspondientes a cada práctica los cuales van incrementando su dificultad de forma gradual y reutilizan los conceptos de prácticas anteriores para un aprendizaje completo.

4.1 Desglose de actividades por práctica

Cada práctica contiene dos tipos de actividades, la primera permite que el alumno evalúe los conocimientos adquiridos sin necesidad de generar una calificación en el sistema, esta actividad consiste en un cuestionario de realimentación de la práctica, con base en el contenido de la misma. Por tanto, para la realimentación se crea un cuestionario de opción múltiple con retroalimenación para cada uno de los casos, es decir, para cuando la respuesta es correcta (retroalimentación positiva) y para cuando la respuesta es incorrecta (retroalimentación negativa).

Además se tiene una actividad que sí genera calificación y requiere que el alumno ponga en práctica los conocimientos adquiridos hasta ese momento en el curso. Para este tipo de actividades se crean diversos recursos didáctico-pedagógicos que motiven el aprendizaje empírico del alumno, tratando de que estas actividades sean autogestivas, para que el alumno reciba retroalimentación inmediata. En algunos casos la actividad a realizar debe ser revisada por el profesor de la asignatura, en tal caso se da una rúbrica para el profesor.

4.2 Material desarrollado

Práctica 1

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Cuál es la función de la máquina virtual de Java (JVM)?

a) Realizar la ejecución de programas realizados con Java.
b) Compilar el código Java.
c) Permitir la instalación del Java Development Kit (JDK).
d) Facilitar al programador la creación de código.

Opción correcta:

a) Realizar la ejecución de programas realizados con java. Correcto: La máquina virtual de Java (JVM) es la pieza encargada de entender y ejecutar el código que se genera tras la compilación del código fuente.

Retroalimentación de las opciones incorrectas:

b) Compilar el código Java. Incorrecta: el compilado de aplicaciones se realiza mediante el compilador java que está incluido en el Java Development Kit (JDK).

c) Permitir la instalación del JDK. Incorrecta: la instalación del Java Development Kit (JDK) está asociada a los diferentes sistemas operativos soportados.

d) Facilitar al programador la creación del código. Incorrecta: la forma de creación del código se puede facilitar con herramientas externas a la máquina virtual de java (JVM), conocidos como IDEs.

Pregunta 2:

¿Cómo se llama el lenguaje que la máquina virtual de Java (JVM) entiende y puede ejecutar?

a) Ensamblador.
b) Código fuente.
c) Bytecode.
d) Código máquina.

Opción correcta:

c) Bytecode. Correcto: Al código generado después de una compilación de un código fuente se le conoce habitualmente como “Bytecode”.

Retroalimentación de las opciones incorrectas:

a) Ensamblador. Incorrecta: El código ensamblador es un código que su uso se enfoca principalmente como código fuente en microcontroladores.

b) Código fuente. Incorrecta: el código fuente es el código realizado por el programador, este código tiene la característica de ser fácilmente entendible por los humanos y su sintaxis está relacionada con el lenguaje en el cual se está programando (Java, C++, C#, etcétera).

d) Código máquina. Incorrecta: El código máquina es el código entendible a nivel de hardware (0 y 1).

Pregunta 3:

¿Cuál es la diferencia entre un archivo ‘.java’ y un archivo ‘.class’?

a) ‘.java’ es el archivo compilado y ‘.class’ el código fuente.
b) El ‘.class’ es código compilado y el ‘.java’ es el código fuente.
c) El ‘.java’ viene en el JDK y el ‘.class’ no.
d) El ‘.java’ se ejecuta y el ‘.class’ se compila.

Opción correcta:

b) El ‘.class’ es código compilado y el ‘.java’ es el código fuente. Correcto: El archivo .class es el archivo generado después de compilar el archivo .java

Retroalimentación de las opciones incorrectas:

a) ‘.java’ es el archivo compilado y ‘.class’ el código fuente. Incorrecta: el archivo .java es lo que el programador escribe (código fuente).

c) El ‘.java’ viene en el JDK y el ‘.class’ no. Incorrecto: los archivos .class son archivos compilados que se pueden ejecutar con el comando java y el archivo .java que es el que contiene el código fuente, por tanto, este archivo es creado por el desarrollador.

d) El ‘.java’ se ejecuta y el ‘.class’ se compila: Incorrecto: el archivo .java se compila y el .class se ejecuta

Pregunta 4:

¿Cuál es la diferencia entre el comando ‘javac’ y el comando ‘java’?

a) ‘javac’ se utiliza para ejecutar el programa y ‘java’ para compilar el código fuente
b) ‘javac’ es una extensión de un archivo y ‘java’ una herramienta del JDK
c) ‘java’ es un archivo ejecutable y ‘javac’ una herramienta del JDK
d) ‘java’ se utiliza para ejecutar el programa y ‘javac’ para compilar el código fuente

Opción correcta:

d) ‘java’ se utiliza para ejecutar el programa y ‘javac’ para compilar el código fuente**.** Correcto, tanto ‘java’ como ‘javac’ con comandos que están incluidos en el JDK, siendo javac el encargado de realizar la compilación y java el encargado de la ejecución

Retroalimentación de las opciones incorrectas:

a) ‘javac’ se utiliza para ejecutar el programa y ‘java’ para compilar el código fuente. Incorrecto: ‘javac’ es el encargado de compilación y ‘java’ el de ejecución.

b) ‘javac’ es una extensión de un archivo y ‘java’ una herramienta del JDK. Incorrecto: javac no es una extensión de archivo, es una herramienta del JDK para compilar, por otro lado ‘java’ se puede usar para referirse al archivo de código fuente ó la utilidad para ejecutar un programa previamente compilado.

c) ‘java’ es un archivo ejecutable y ‘javac’ una herramienta del JDK. Incorrecto: ‘javac’ sí es una herramienta del JDK que permite compilar con código fuente, pero ‘java’ es un archivo de código fuente ó una herramienta del JDK para ejecutar programas previamente compilados, no un archivo ejecutable.

Pregunta 5:

¿Qué es un entorno de desarrollo integrado (IDE)?

a) Una aplicación que contiene todas las herramientas de desarrollo en una interfaz
b) Un módulo del entorno JAVA.
c) Un editor de texto.
d) Un compilador de archivos java.

Opción correcta:

a) Una aplicación que contiene todas las herramientas de desarrollo en una interfaz**.** Correcto: los entornos de desarrollo integrados (IDE) son una aplicación que permite al desarrollador realizar código de una manera más sencillas en una interfaz que tiene todas las herramientas necesarias para la creación de programas.

Retroalimentación de las opciones incorrectas:

b) Un módulo del entorno JAVA. Incorrecto: los entornos de desarrollo integrados son independientes al lenguaje de programación, éstos pueden ser multilenguaje.

c) Un editor de texto. Incorrecto: si bien un entorno de desarrollo integrado contiene un editor de texto no es lo único, el editor de texto es sólo una herramienta más para apoyar al programador a realizar código.

d) Un compilador de archivos java: Incorrecto, si bien un IDE tiene sus herramientas de compilación internas no significa que sea lo único, un IDE es un conjunto de herramientas para el desarrollo de programas ó sistemas.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno pueda realizar las prácticas posteriores y tenga el conocimiento de cómo configurar un entorno para el desarrollo de programas usando Java.

Instalación del entorno y creación del primer programa

Java es una plataforma que permite el desarrollo de aplicaciones de escritorio, web ó de consola bajo el paradigma de la programación orientada a objetos (POO) que permite simular situaciones cotidianas y programar sus interacciones.

Se deja al alumno la posibilidad de instalar cualquier IDE con el que se sienta cómodo y experimentar con él, sin embargo, las primeras prácticas se tendrán que realizar sin uno, para afianzar las bases de la programación orientada a objetos usando java.

Realizar la instalación del Entorno de Desarrollo de Java (JDK) y realizar un programa “HolaMundo.java” donde al ejecutarse se pueda visualizar el mensaje “HolaMundo: {Nombre}-{Numero_de_cuenta}”.

1.- Ejecutar los siguientes comandos en la terminal/consola de tu sistema operativo y tomar captura de pantalla a la salida de cada comando:

- java -version

- javac

2.- En la consola/terminal colocarse en el directorio que contenga la clase creada “HolaMundo.java” y ejecutar los siguientes comandos, tomar captura de pantalla de la salida del comando:

- javac HolaMundo.java

- java HolaMundo

Crear un documento PDF que contenga:

- Capturas de los comandos ejecutados
- Archivo del código fuente (.java)

Rúbrica de evaluación

Ejercicio 1

Las siguientes salidas de los comandos son las que deberían de aparecer, el número 1.8.0_181 puede variar dependiendo de la versión que haya instalado el alumno.

```
C:\                    java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

La siguiente salida es correcta ya que demuestra que se detecta el comando javac

```
C:\          >javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are used
  -classpath <path>          Specify where to find user class files and annotation processors
  -cp <path>                 Specify where to find user class files and annotation processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>      Specify where to find annotation processors
  -parameters                Generate metadata for reflection on method parameters
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -h <directory>             Specify where to place generated native header files
  -implicit:{none,class}     Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release
  -target <release>          Generate class files for specific VM version
  -profile <profile>         Check that API used is available in the specified profile
  -version                   Version information
  -help                      Print a synopsis of standard options
  -Akey[=value]              Options to pass to annotation processors
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system
  -Werror                    Terminate compilation if warnings occur
  @<filename>                Read options and filenames from file
```

Las siguientes salidas son incorrectas:

```
C:\           >java -version
"java" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
```

Esta salida indica que no se ha instalado el JDK de la manera correcta, esto se puede deber a que no se instaló el JDK ó que no se han modificado las variables de entorno para ser detectado desde consola.

```
C:\           javac
"javac" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
```

Esta salida tiene la misma explicación sobre la falta de la configuración de las variables de entorno ó una falta completa de la instalación del JDK.

```java
public class HolaMundo{
    public static void main(String[] args) {
    System.out.println("Hola Alumno-30804888-1");
    }

}
```

Ejecución en línea de comandos del ejercicio 2

```
C:\Users\Juan\Documents>javac HolaMundo.java
```

En este primer comando no debe imprimir nada en pantalla, cualquier otra salida es errónea.

```
C:\Users\Juan\Documents>java HolaMundo
Hola Alumno-30804888-1
```

En este segundo comando se debe ver una salida similar a la que se muestra en la imagen, cualquier otra salida que no muestre un saludo con nombre del alumno y un número de cuenta es errónea.

Práctica 2

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿A qué se le llaman “tipos de datos”?

a) Al rango de valores que puede tomar una variable
b) Al nombre de las variables
c) Ninguna de las anteriores

Opción correcta:

a) Al rango de valores que puede tomar una variable**.** Correcto: el tipo de dato de una variable es el que define qué valores puede tomar dicha variable.

Retroalimentación de las opciones incorrectas:

b) Al nombre de las variables. Incorrecta: el nombre de las variables puede ser definido por el programador siempre y cuando no se usen palabras reservadas.

c) Ninguna de las anteriores. Incorrecta: el tipo de datos define que valores puede tomar una variable.

Pregunta 2:

¿Qué son las estructuras de control?

a) Son un tipo de dato que permite controlar el flujo de ejecución del programa.
b) Son herramientas del lenguaje que permiten modificar el flujo de ejecución del programa.
c) Son variables que permiten modificar el flujo de ejecución del programa.

Opción correcta:

b) Son herramientas del lenguaje que permiten modificar el flujo de ejecución del programa. Correcto: las estructuras de control son una herramienta del lenguaje de programación que permiten controlar el flujo del programa mediante ciclos ó condiciones.

Retroalimentación de las opciones incorrectas:

a) Son un tipo de dato que permite controlar el flujo de ejecución del programa. Incorrecta: las estructuras de control no son un tipo de dato, los tipos de dato definen los valores aceptados por una variable, las estructuras de control permiten modificar el flujo del programa en tiempo de ejecución.

c) Son variables que permiten controlar el flujo de ejecución del programa. Incorrecta: las variables las declara el programador para almacenar en ellas datos que se utilizan dentro del programa, estas variables pueden ayudar a las estructuras de control a tomar decisiones ó detener ciclos en tiempo de ejecución del programa.

Pregunta 3:

¿Cuál es la diferencia entre variables miembro y variables locales?

a) Las variables miembro se definen dentro de un método y las variables locales se definen a nivel de clase.
b) Las variables locales se definen dentro de un método y las variables miembro se definen a nivel de clase.
c) Las variables locales se definen dentro de un método y las variables miembro son referencias a una variable local.

Opción correcta:

b) Las variables locales se definen dentro de un método y las variables miembro se definen a nivel de clase. Correcto: las variables locales sólo se pueden usar dentro del contexto donde fueron definidas, mientras que las variables miembro pueden ser accedidas por todos los métodos de la clase en cualquier momento.

Retroalimentación de las opciones incorrectas:

a) Los variables miembro se definen dentro de un método y las variables locales se definen a nivel de clase. Incorrecto: las variables miembro se definen a nivel de clase siendo éstas accesibles para todos los métodos de la clase y las variables locales sólo pueden ser usadas en el bloque en el cual fueron creadas.

c) Las variables locales se definen dentro de un método y las variables miembro son referencias a una variable local. Incorrecto: las variables miembro se definen a nivel de clase y son diferentes a las variables locales ya que éstas sólo pueden ser accesibles dentro del método donde se definieron.

Pregunta 4:

¿Cuáles son los tipos principales de variables?

a) Primitivos ó referencia.
b) Referencia ó apuntadores.
c) Primitivos ó apuntadores.

Opción correcta:

a) Primitivos ó referencia. Correcto: las variables pueden ser de tipo primitivo (int, boolean, etcétera) ó pueden ser declaradas como referencias a objetos (Integer, String, etcétera).

Retroalimentación de las opciones incorrectas:

b) Referencia ó apuntadores. Incorrecto: las variables se pueden declarar como tipos primitivos ó referencias, java no permite el uso de apuntadores.

c) Primitivos ó apuntadores. Incorrecto: las variables se pueden declarar como tipos primitivos ó referencias, java no permite el uso de apuntadores.

Pregunta 5:

¿Para qué sirve el operador new?

a) Para declarar una nueva variable.
b) Para crear una variable primitiva.
c) Para crear un Objeto.

Opción correcta:

c) Para crear un Objeto. Correcto: para crear un nuevo objeto en memoria se debe de usar el operador new haciendo referencia a la clase del cual se desea instanciar un nuevo objeto.

Retroalimentación de las opciones incorrectas:

a) Para declarar una nueva variable: Incorrecto, para declarar una nueva variable basta con definir el tipo de dato y el nombre, ya sea como un tipo de dato primitivo ó de referencia.
b) Para crear una variable primitiva. Incorrecto: las variables primitivas sólo requieren ser declaradas y asignarles un valor, dichas variables no pueden referenciar a un objeto que haya sido creado con el operador new.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno haya adquirido los conocimientos del funcionamiento de las estructuras de control y pueda manejar la interacción entre ellas para la realización de una tarea sencilla.

Rellenar los espacios para generar un programa funcional

Las estructuras de control, los tipos de datos y las condiciones permiten modificar el flujo de un programa mediante ciclos ó toma de decisiones para alcanzar el objetivo deseado de la aplicación y que ésta se pueda adaptar a diferentes situaciones que el usuario requiera.

Se tiene el siguiente programa que hace una suma de los primeros n números pares ó impares empezando desde 0 y teniendo como límite superior un número ingresado por el usuario, el programa tiene un menú para seleccionar qué tipo de suma se desea realizar (de números pares ó de números impares). Una vez ingresada la opción del menú el programa pide un número máximo para que éste sea el límite superior de dicha suma.

Completar el siguiente código con las opciones dadas para obtener la funcionalidad descrita anteriormente, ignorar de momento el error que sale en caso de insertar un valor que no sea numérico en línea de comandos, en prácticas posteriores se revisará ese tema.

Opciones:

- in
- out
- sc.nextInt();
- ==
- ¡=
- 3
- 2
- sc.close();

```
import java.util.Scanner;
public class Practica2 {
public static void main(String[] args) {
    int numeroMaximo=0;
    int opcion=0;
    1.- Scanner sc= new Scanner(System. _____ );
    do {
       int suma=0;
       System.out.println("Elige la opción deseada");
       System.out.println("1.- Suma de numeros pares");
       System.out.println("2.- Suma de numeros impares");
       System.out.println("3.- Salir");
       opcion= sc.nextInt();
       if(opcion != 3) {
          System.out.println("Ingrese número máximo: ");
          2.- numeroMaximo= _____
       }
       switch(opcion) {
          case 1:
              System.out.println("Suma de numeros pares con un límite de: "+ numeroMaximo);
            for (int i=0; i<=numeroMaximo;i++) {
              3.- if (i%2 _____ 0) {
                  suma= suma +i;
              }
            }
              System.out.println("El resultado de la suma es: "+ suma);
            break;
          case 2:
   System.out.println("Suma de numeros impares con un límite de: "+     numeroMaximo);
  for (int i=0; i<=numeroMaximo;i++) {
               4.- if (i%2 _____ 0) {
                  suma= suma +i;
               }
            }
              System.out.println("El resultado de la suma es: "+ suma);
            break;
          case 3:
              System.out.println("Saliendo del programa");
    break;
            default:
             System.out.println("Opción incorrecta");
               break;
       }
  5.- }while(opcion!= _____  );
  6.-_____
   }
}
```

Solución y retroalimentación

1.- Scanner sc= new Scanner(System. _____);

- **Respuesta correcta**: "**In**", correcto, Scanner requiere un flujo para recibir información del sistema, en este caso System.in es el que permite que el usuario ingrese datos desde teclado.
- Respuesta incorrecta: "**Out**", incorrecto, si bien System.out existe en un entorno java esta opción provocaría un error de compilación, debido a que Scanner requiere un objeto de entrada de datos "InputStream" y System.out es un objeto de salida de datos (PrintStream).
- Respuesta incorrecta: "**sc.nextInt();**", incorrecto, esta opción provocaría un fallo al compilar por la sintaxis.
- Respuesta incorrecta: "**==**", incorrecto, esta opción carece de sentido en esta línea y provocaría un fallo de sintaxis a la hora de compilar**.**
- Respuesta incorrecta: "**!=**", incorrecto, esta opción carece de sentido en esta línea y provocaría un fallo de sintaxis a la hora de compilar**.**
- Respuesta incorrecta: "**3**", incorrecto, esta opción provocaría un fallo de sintaxis al momento de compilación
- Respuesta incorrecta: "**2**", incorrecto, esta opción provocaría un fallo de sintaxis al momento de compilación
- Respuesta incorrecta: "**sc.close();**", incorrecto, esta opción provocaría un fallo de sintaxis al momento de compilación además de carecer de sentido por usar el mismo objeto que se quiere referenciar en el constructor al momento de referenciarlo.

2.- numeroMaximo= _____

- Respuesta incorrecta: "**In**", incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra in.
- Respuesta incorrecta: "**out**", incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra out.
- **Respuesta correcta**: "**sc.nextInt();**", correcto, con esta instrucción se logra el leer un número ingresado desde teclado al momento de la ejecución.
- Respuesta incorrecta: "**==**", incorrecto, esta opción carece de sentido en esta línea y provocaría un fallo de sintaxis al momento de compilar**.**
- Respuesta incorrecta: "**!=**", incorrecto, esta opción carece de sentido en esta línea y provocaría un fallo de sintaxis al momento de compilar**.**
- Respuesta incorrecta: "**3**", incorrecto, el programa funcionaría, sin embargo, no tendría la funcionalidad de cambiar el número máximo, es decir, siempre estaría el número 3 como límite, lo cual no es la funcionalidad buscada.
- Respuesta incorrecta: "**2**", incorrecto, el programa funcionaría, sin embargo, no tendría la funcionalidad de cambiar el número máximo, es decir, siempre estaría el número 2 como límite, lo cual no es la funcionalidad buscada.

- Respuesta incorrecta: “**sc.close();**”, incorrecto, esta opción provocaría un fallo de sintaxis al momento de compilación debido a que ‘sc.close();’ es una función que no regresa un valor y en esta línea se intenta asignar un valor a “numeroMaximo”, además de no ayudar a completar nuestra funcionalidad deseada.

3.- if (i%2 _____ 0) {

- Respuesta incorrecta: “**In**”, incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra in.
- Respuesta incorrecta: “**out**”, incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra out.
- Respuesta correcta: “**sc.nextInt();**”, incorrecto, esta línea provocaría un fallo al momento de compilar, además de carecer de sentido al no aportar la funcionalidad que se espera del programa
- **Respuesta correcta:** “**==**”, correcto, esta opción permite comparar dos valores primitivos, en este caso se utiliza i%2 para obtener el módulo del número actual (i), si el residuo de la operación i/2 (es decir i%2) es igual a 0 indica que el número es par y éste se deberá sumar al total de números pares**.**
- Respuesta incorrecta: “**!=**”, incorrecto, esta opción no provocaría ningún fallo al compilar, sin embargo no genera la funcionalidad deseada ya que en el bloque de código donde se está usando es la funcionalidad de la suma de números pares, el operador != es para indicar que el módulo sea diferente a 0 lo cual daría la funcionalidad de la suma de números impares**.**
- Respuesta incorrecta: “**3**”, incorrecto, esta opción generaría un fallo de compilación debido a que la instrucción if requiere un valor booleano producto de una evaluación y al colocar un número dicha evaluación no existe.
- Respuesta incorrecta: “**2**”, incorrecto, esta opción generaría un fallo de compilación debido a que la instrucción if requiere un valor booleano producto de una evaluación y al colocar un número dicha evaluación no existe.
- Respuesta incorrecta: “**sc.close();**”, incorrecto, esta opción carece de sentido en este punto y se generaría un error de compilación.

4.- if (i%2 _____ 0) {

- Respuesta incorrecta: “**In**”, incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra in.
- Respuesta incorrecta: “**out**”, incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra out.
- Respuesta incorrecta: “**sc.nextInt();**”, incorrecto, esta línea provocaría un fallo a la hora de compilación, además de carecer de sentido al no aportar la funcionalidad que se espera del programa.
- Respuesta incorrecta: “**==**”, incorrecto, esta opción no provocaría ningún fallo al compilar, sin embargo, no genera la funcionalidad deseada ya que en el bloque

de código donde se está usando es la funcionalidad de la suma de números impares, el operador == es para indicar que el módulo sea igual a 0 lo cual nos daría la funcionalidad de la suma de números pares**.**

- **Respuesta correcta**: "**!=**", correcto, esta opción permite comparar dos valores, en este caso se utiliza i%2 para obtener el módulo del número actual (i), si el residuo de la operación i/2 (es decir i%2) es diferente a 0 indica que el número es impar y éste se deberá sumar al total de números impares.
- Respuesta incorrecta: "**3**", incorrecto, esta opción generaría un fallo de compilación debido a que la instrucción if requiere un valor booleano producto de una evaluación y al colocar un número dicha evaluación no existe.
- Respuesta incorrecta: "**2**", incorrecto, esta opción generaría un fallo de compilación debido a que la instrucción if requiere un valor booleano producto de una evaluación y al colocar un número dicha evaluación no existe.
- Respuesta incorrecta: "**sc.close();**", incorrecto, esta opción carece de sentido debido a que la condición if requiere un valor booleano para evaluar y sc.close() no regresa ningún valor, además de regresar un valor booleano habría un error de sintaxis por ";" que en ese caso no debería escribirse.

5.- }while(opcion!= _____);

- Respuesta incorrecta: "**In**", incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra in.
- Respuesta incorrecta: "**out**", incorrecto, esto provocaría un fallo de sintaxis al momento de compilación debido a no reconocer la palabra out.
- Respuesta correcta: "**sc.nextInt();**", incorrecto, esta línea provocaría un fallo al compilar y no cumple con la intención del programa.
- Respuesta incorrecta: "**==**", incorrecto, esta opción carece de sentido y provocaría un fallo de sintaxis al compilar**.**
- Respuesta incorrecta: "**!=**", incorrecto, esta opción carece de sentido y provocaría un fallo de sintaxis al compilar.
- **Respuesta correcta**: "**3**", correcto, esta opción permite mantener el ciclo de mostrar el menú y recibir la opción del usuario mientras la opción sea diferente de 3, que es la opción de salir del programa.
- Respuesta incorrecta: "**2**", incorrecto, esta opción permite que el programa compile, pero no cumple con lo que se espera del programa al insertar la opción 2 se esperaría poder realizar una suma de números impares, sin embargo, con esta opción el programa terminaría su ejecución en vez de hacer la suma solicitada.
- Respuesta incorrecta: "**sc.close();**", incorrecto, esta opción carece de sentido debido a que sc.close() no regresa un valor con el cual comparar la variable "opción" además de regresar un valor, provocaría un fallo de compilación por un error de sintaxis debido a que el ';' sobraría y en caso de quitarlo tampoco cumpliría con el objetivo del programa.

6.-_____

- Respuesta incorrecta: “**In**”, incorrecto, esto provocaría un fallo de sintaxis al compilar debido a no reconocer la palabra in.
- Respuesta incorrecta: “**out**”, incorrecto, esto provocaría un fallo de sintaxis al compilar debido a no reconocer la palabra out.
- Respuesta incorrecta: “**sc.nextInt();**”, incorrecto, esta línea permitiría la compilación del programa pero antes de salir del programa dejaría un cursor en la consola esperando recibir un valor, después terminaría el programa, lo cual es una funcionalidad no óptima, además de dejar el objeto “sc” en memoria sin liberarla.
- Respuesta incorrecta: “**==**”, incorrecto, esta opción carece de sentido y provocaría un fallo de sintaxis al momento de compilar**.**
- Respuesta incorrecta: “**!=**”, incorrecto, esta opción carece de sentido y provocaría un fallo de sintaxis al momento de compilar.
- Respuesta incorrecta: “**3**”, incorrecto, esta opción provocaría un fallo de compilación debido a que sería un número fuera de contexto que no aportaría nada al programa.
- Respuesta incorrecta: “**2**”, incorrecto, esta opción provocaría un fallo de compilación debido a que sería un número fuera de contexto que no aportaría nada al programa.
- **Respuesta correcta**: “**sc.close();**”, correcto, al terminar de usar un objeto de la clase Scanner se debe de cerrar el flujo de éste, debido a que se debe de liberar la memoria al terminar el proceso para que el sistema operativo lo recupere, además en sistemas más complejos el dejar un flujo de este tipo abierto puede provocar vulnerabilidades de seguridad.

Código completo

```
import java.util.Scanner;

public class Practica2 {

    public static void main(String[] args) {
        int numeroMaximo=0;
        int opcion=0;
        Scanner sc= new Scanner(System.in);
        do {
            int suma=0;
            System.out.println("Elige la opción deseada");
            System.out.println("1.- Suma de numeros pares");
            System.out.println("2.- Suma de numeros impares");
            System.out.println("3.- Salir");
            opcion= sc.nextInt();
            if(opcion != 3) {
                System.out.println("Ingrese número máximo: ");
                numeroMaximo= sc.nextInt();
            }
            switch(opcion) {
            case 1:
                System.out.println("Suma de numeros pares con un límite de : "+ numeroMaximo);
                for (int i=0; i<=numeroMaximo;i++) {
                    if (i%2==0) {
                        suma= suma +i;
                    }
                }
                System.out.println("El resultado de la suma es: "+ suma);
                break;
            case 2:
                System.out.println("Suma de numeros impares con un límite de : "+ numeroMaximo);
                for (int i=0; i<=numeroMaximo;i++) {
                    if (i%2!=0) {
                        suma= suma +i;
                    }
                }
                System.out.println("El resultado de la suma es: "+ suma);
                break;
            case 3:
                System.out.println("Saliendo del programa");
                break;
            default :
                System.out.println("Opción incorrecta");
                break;
            }
        }while(opcion!=3);
        sc.close();
    }

}
```

Salida:

```
Elige la opción deseada
1.- Suma de numeros pares
2.- Suma de numeros impares
3.- Salir
1
Ingrese número máximo:
11
Suma de numeros pares con un límite de : 11
El resultado de la suma es: 30
Elige la opción deseada
1.- Suma de numeros pares
2.- Suma de numeros impares
3.- Salir
2
Ingrese número máximo:
10
Suma de numeros impares con un límite de : 10
El resultado de la suma es: 25
Elige la opción deseada
1.- Suma de numeros pares
2.- Suma de numeros impares
3.- Salir
3
Saliendo del programa
```

Práctica 3

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Qué es un arreglo en programación?

a) Un objeto donde se puede almacenar varios datos de diferente tipo.
b) Un objeto donde se tienen datos en pares clave-valor.
c) Un objeto donde se puede almacenar varios datos del mismo tipo.

Opción correcta:

c) Un objeto donde se puede almacenar varios datos del mismo tipo. Correcto: un arreglo permite almacenar varios datos del mismo tipo bajo un mismo nombre (el nombre del arreglo).

Retroalimentación de las opciones incorrectas:

a) Un objeto donde se puede almacenar varios datos de diferente tipo. Incorrecta: los arreglos pueden almacenar varios datos, pero deben ser de un mismo tipo, el cual se define en la declaración, por ejemplo, un arreglo que almacene enteros se declara: “int[] números;”.

b) Un objeto donde se tienen datos en pares clave-valor. Incorrecta: los arreglos no tienen una clave para acceder al valor, lo que se utiliza para acceder a cada valor es un índice.

Pregunta 2:

¿Cuál de las siguientes declaraciones es incorrecta?

a) int [ ] numeros;
b) char caracteres[ ];
c) string [ ] nombres;

Opción correcta:

c) string [] nombres; Correcto: el orden de los corchetes ([ ]) es correcto, sin embargo, el nombre de la clase es incorrecto, debería ser "String".

Retroalimentación de las opciones incorrectas:

a) int [] numeros; Incorrecta: la declaración es correcta, los corchetes se pueden declarar antes del nombre de la variable.

b) char caracteres[]; Incorrecta: la declaración de los arreglos puede llevar los corchetes después del nombre de la variable.

Pregunta 3:

Cuando se manejan cadenas, ¿cuál es el uso del operador '+'?

a) Incrementar la cantidad de caracteres que puede contener una cadena.
b) Concatenar 2 cadenas para formar otra.
c) Sumar las longitudes de dos ó más cadenas para obtener la longitud total.

Opción correcta:

b) Concatenar 2 cadenas para formar otra. Correcto: Dos ó más cadenas se pueden unir utilizando el operador '+'. A esta operación se le conoce como concatenación.

Retroalimentación de las opciones incorrectas:

a) Incrementar la cantidad de caracteres que puede contener una cadena. Incorrecta: el operador '+' nos permite concatenar dos cadenas para crear una nueva.

c) Sumar las longitudes de dos ó más cadenas para obtener la longitud total: Incorrecta, para obtener la longitud de una cadena se usa el método length() incluido en la clase String, el operador + nos permite el concatenar dos ó más cadenas para obtener una nueva cadena.

Pregunta 4:

¿Qué son los Wrappers?

a) Son clases diseñadas para ser un complemento de los tipos primitivos.
b) Son clases diseñadas para la creación de tipos primitivos.
c) Son clases diseñadas para sustituir a los tipos primitivos.

Opción correcta:

a) Son clases diseñadas para ser un complemento de los tipos primitivos. Correcto: los wrappers son clases diseñadas para complementar a los tipos primitivos y envolver su tipo en una clase ya que los tipos primitivos en java no son objetos, dándoles así más funcionalidad.

Retroalimentación de las opciones incorrectas:

b) Son clases diseñadas para la creación de tipos primitivos. Incorrecta: los wrapper envuelven a un tipo primitivo, sin embargo, no crea un tipo primitivo, ya que éstos existen en java y no son objetos, lo que hace un wrapper es añadir funcionalidad de clases a un tipo primitivo en específico dependiendo del wrapper que se esté utilizando.

c) Son clases diseñadas para sustituir a los tipos primitivos. Incorrecto: los wrappers pueden tener muchas ventajas y se podría pensar que se hicieron para sustituir a los tipos primitivos, sin embargo, usan más memoria, en términos de eficiencia los tipos primitivos son más eficaces que los wrapper, el uso de uno no sustituye al otro, son complementos para uso del programador en caso de necesitarlos.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno use los conocimientos adquiridos sobre la API de java y sus diferentes clases para resolver problemas sencillos.

Rellenar los espacios para generar un programa funcional

Java tiene clases muy útiles a la hora de programar y a la hora de extender la funcionalidad de los tipos de datos primitivos mediante clases especiales. Además permite agrupar información de un mismo tipo en estructuras que pueden ser arreglos ó colecciones, siendo éstas últimas una estructura más compleja con funcionalidades que pueden permitir resolver problemas de una manera óptima.

Objetivo de la evaluación: Que el alumno use los conocimientos adquiridos sobre la API de java y sus diferentes clases para resolver problemas sencillos.

Se debe crear un programa que genera un arreglo de 10 enteros entre 0 y 20 de manera aleatoria, una vez creado dicho arreglo se debe buscar un par de números que sumados den un número que se recibe mediante los argumentos del método main. En la siguiente imagen se puede ver la manera en la que se debe ejecutar el programa, puede verse el arreglo generado, el número enviado mediante los argumentos del método main y los pares encontrados para dicho número:

```
Se revisará el numero: 10
[9,15,17,8,9,3,7,19,15,7]
Realizando algoritmo de busqueda de pares que sumen: 10
Par encontrado: 3+7=10
Par encontrado: 3+7=10
```

Completar el siguiente código con las opciones dadas para obtener la funcionalidad descrita anteriormente

Opciones

- int[10]
- array[i] - array[j]
- array.length
- array[i] + array[j]
- 20
- array.length();
- args[0]

```
public class Practica3 {
    public static void main(String[] args) {
        int array[]= new       ;
        boolean found=false;
        Integer numberToFind=new Integer(       );
        System.out.println("Se revisará el numero: "+ numberToFind);
        System.out.print("[");
        for (int i =0; i<array.length;i++) {
            array[i]=(int) (Math.random()*   );
            System.out.print(array[i]+ (i!=(array.length-1)? ",":""));
        }
        System.out.println("]");

        System.out.println("Realizando algoritmo de busqueda de pares que sumen: "+numberToFind);
        for (int i=0; i<          ;i++) {
            if (i+1!=array.length) { //This is used to avoid overflow
                for (int j=i+1;j<array.length;j++) {
                    if (                ==numberToFind) {
                        System.out.println("Par encontrado: "+array[i]+"+"+array[j]+"="+numberToFind);
                        found=true;
                    }
                }
            }
        }
        if(!found) {
            System.out.println("No se encontraron pares que sumen: " +numberToFind);
        }

    }

}
```

Solución y retroalimentación

1.- int array[]= new _____;

- **Respuesta correcta**: "**int [10]**", correcto, la especificación del programa indica que se debe generar un arreglo de 10 números enteros, ésta es la manera en la que crea el arreglo.
- Respuesta incorrecta: "**array[i] - array[j]**", incorrecto, esta línea provocaría múltiples errores a la hora de compilación ya que las variables i, j no existen en el punto de esta línea, además de carecer de sentido usar el arreglo array en la misma línea que se está creando.
- Respuesta incorrecta: "**array.length**", incorrecto, esta opción genera un error de sintaxis en tiempo de compilación, además de carecer de sentido usar el arreglo array en la misma línea que se está creando.
- Respuesta incorrecta: "**array[i] + array[j]"**, incorrecto, esta opción carece de sentido al usar el arreglo en la misma línea que se está creando, además de que las variables i, j aún no existen en esta parte del código.
- Respuesta incorrecta: "**20"**, incorrecto, esta línea genera un error de sintaxis al compilar debido a que el operador new requiere una clase y la manera correcta de declarar el tamaño del arreglo sería int[10] para cumplir los requerimientos del programa.
- Respuesta incorrecta: "**array.length();**", incorrecto, esta opción generaría un error de sintaxis además de carecer de sentido usar el arreglo array en la misma línea que se está creando.
- Respuesta incorrecta: "**args[0]**", incorrecto, esta línea provoca un error de sintaxis ya que el operador new requiere una clase para poder ser utilizado, mientras args[0] entrega el primer argumento recibido desde línea de comandos, el cual es una cadena.

2.- Integer numberToFind=new Integer(_____);

- Respuesta Incorrecta: "**int [10]**", incorrecto, colocar este pedazo de código en este lugar provoca un error de sintaxis debido a que el constructor requiere un valor numérico.
- Respuesta incorrecta: "**array[i] - array[j]**", incorrecto, esta línea provocaría múltiples errores a la hora de compilación ya que las variables i, j no existen en el punto de esta línea.
- Respuesta incorrecta: "**array.length**", incorrecto, esta opción no genera un error de sintaxis en tiempo de compilación, sin embargo no cumple la funcionalidad deseada que es la de enviar el número a buscar mediante línea de comandos, el número siempre sería equivalente a la longitud del arreglo declarado anteriormente (en este caso 10) .

- Respuesta incorrecta: “**array[i] + array[j]”**, incorrecto, esto provoca un error de sintaxis al momento de compilación debido a que las variables i, j aún no existen en esta parte del código.
- Respuesta incorrecta: “**20”**, incorrecto, esta opción no genera un error de sintaxis, pero no cumple con la funcionalidad deseada debido a que el número a buscar siempre sería 20 y se busca que el número sea ingresado mediante línea de comandos.
- Respuesta incorrecta: “**array.length();**”, incorrecto, esta opción generaría un error de sintaxis debido a que length es un atributo del arreglo, no un método que se pueda llamar y en caso de que se intentase usar array.length el programa funcionaría pero limitaría el número a buscar a la longitud del arreglo, en este caso 10 lo cual no es la funcionalidad deseada.
- **Respuesta correcta**: “**args[0]**”, correcto, este pedazo de código nos permite acceder al primer argumento enviado mediante línea de comandos, que en este caso se busca el número enviado por el usuario y se transforma en un objeto usando la clase wrapper Integer.

3.- array[i]=(int) (Math.random()* _____);

- Respuesta Incorrecta: “**int [10]**”, incorrecto, colocar este pedazo de código en este lugar provoca un error de sintaxis debido a que se está buscando multiplicar un valor aleatorio por otro número.
- Respuesta incorrecta: “**array[i] - array[j]**”, incorrecto, esta línea provocaría error a la hora de compilación ya que la variable j no existe en ese momento.
- Respuesta incorrecta: “**array.length**”, incorrecto, esta opción no genera un error de sintaxis en tiempo de compilación, sin embargo no cumple la funcionalidad deseada que es que el número máximo que exista en el arreglo sea 20, en este caso el número máximo posible sería la longitud del arreglo (10).
- Respuesta incorrecta: “**array[i] + array[j]”**, incorrecto, esta línea provocaría múltiples errores a la hora de compilación ya que la variable j no existe en ese momento.
- **Respuesta correcta**: “**20”**, correcto, esta opción permite llegar al objetivo deseado que es generar un arreglo de números donde el valor máximo sea 20 ya que Math.random() genera un valor entre 0 y 1.
- Respuesta incorrecta: “**array.length();**”, incorrecto, esta opción generaría un error de sintaxis debido a que length es un atributo del arreglo, no un método que se pueda llamar y en caso de que se intentase usar array.length el programa funcionaría pero limitaría el número máximo del arreglo aleatorio a 10 lo cual no es la funcionalidad deseada.

- Respuesta Incorrecta: “**args[0]**”, incorrecto, esto genera un error de compilación debido a que se busca multiplicar un número aleatorio por otro y el arreglo de argumentos “args” es del tipo String el cual no es compatible con el operador *.

4.- for (int i=0; i< _____ ;i++) {

- Respuesta Incorrecta: “**int [10]**”, incorrecto, colocar este pedazo de código en este lugar provoca un error de sintaxis ya que carece de sentido.
- Respuesta incorrecta: “**array[i] - array[j]**”, incorrecto, esta línea provocaría error a la hora de compilación ya que la variable j no existe en ese momento del código.
- **Respuesta correcta**: “**array.length**”, correcto, este bloque de código permite cumplir la funcionalidad deseada que es la de recorrer el arreglo para ir comparando elemento por elemento y usar una estrategia de fuerza bruta para resolver el problema.
- Respuesta incorrecta: “**array[i] + array[j]”**, incorrecto, esta línea provocaría múltiples errores a la hora de compilación ya que la variable j no existe en ese momento.
- Respuesta incorrecta: “**20”**, incorrecto, esta opción no genera errores de sintaxis al momento de compilación sin embargo no cumple con el objetivo del programa ya que se busca recorrer el arreglo de 10 elementos y provocaría un error de desbordamiento al intentar acceder a un elemento del arreglo el cual no existe.
- Respuesta incorrecta: “**array.length();**”, incorrecto, esta opción generaría un error de sintaxis debido a que length es un atributo del arreglo, no un método que se pueda llamar.
- Respuesta Incorrecta: “**args[0]**”, incorrecto, esto genera un error de sintaxis al momento de compilación, ya que se compararía un valor int (i) con un String.

5 .- if (_____ ==numberToFind) {

- Respuesta Incorrecta: “**int [10]**”, incorrecto, colocar este pedazo de código en este lugar provoca un error de sintaxis ya que carece de sentido.
- Respuesta incorrecta: “**array[i] - array[j]**”, incorrecto, esta línea no provoca un error de sintaxis al momento de compilación, sin embargo, no cumple con la funcionalidad buscada que es encontrar dos valores que sumados den el número ingresado en línea de comandos, este bloque nos permitiría encontrar la resta.
- Respuesta incorrecta: “**array.length**”, incorrecto, este bloque de código no provoca un error de sintaxis al momento de compilación, sin embargo no nos lleva al objetivo que es comparar la suma de dos números con el número a buscar y siempre compararía con la cantidad de elementos del arreglo, sin importar los valores que se encuentren en este.

- **Respuesta correcta**: “**array[i] + array[j]”**, correcto, este bloque de código permite que se compare la suma del elemento actual y el elemento siguiente con el número a buscar, en caso de coincidir se accede al bloque if y se imprimen dichos valores.
- Respuesta incorrecta: “**20”**, incorrecto, esta opción no genera errores de sintaxis al momento de compilación sin embargo no cumple con el objetivo del programa ya que compararía el número ingresado a buscar con 20 lo cual no es la funcionalidad que estamos buscando.
- Respuesta incorrecta: “**array.length();**”, incorrecto, esta opción generaría un error de sintaxis debido a que length es un atributo del arreglo, no un método que se pueda llamar.
- Respuesta Incorrecta: “**args[0]**”, incorrecto, esto genera un error de sintaxis al momento de compilación, debido a que se buscaría comparar una cadena contra un valor entero.

Código completo:

```
public class Practica3 {
	public static void main(String[] args) {
		int array[]= new int[10000];
		boolean found=false;
		Integer numberToFind=new Integer(args[0]);
		System.out.println("Se revisará el numero: "+ numberToFind);
		System.out.print("[");
		for (int i =0; i<array.length;i++) {
			array[i]=(int) (Math.random()*20);
			System.out.print(array[i]+ (i!=(array.length-1)? ",":""));
		}
		System.out.println("]");
		System.out.println("Realizando algoritmo de busqueda de pares que sumen:
		"+numberToFind);
		for (int i=0; i<array.length;i++) {
			if (i+1!=array.length) { //This is used to avoid overflow
				for (int j=i+1;j<array.length;j++) {
					if (array[i]+array[j]==numberToFind) {
						System.out.println("Par encontrado:
						"+array[i]+"+"+array[j]+"="+numberToFind);
						found=true;
					}
				}
			}
		}
		if(!found) {
			System.out.println("No se encontraron pares que sumen: "
			+numberToFind);
		}
	}
}
```

Material extra: Diferentes soluciones usando colecciones

Los siguientes códigos de ejemplo son soluciones usando colecciones.

Usando la clase List:

```
public class Practica3List {
	public static void main(String[] args) {
		int array[]= new int[10];
		boolean found=false;
		Integer numberToFind=new Integer(args[0]);
		List<Integer> list= new ArrayList<Integer>();
		System.out.println("Se revisará el numero: "+ numberToFind);
		System.out.print("[");
		for (int i =0; i<array.length;i++) {
			array[i]=(int) (Math.random()*20);
			System.out.print(array[i]+ (i!=(array.length-1)? ",":""));
		}
		System.out.println("]");
		System.out.println("Realizando algoritmo de busqueda de pares que sumen:
		"+numberToFind);
		for (int i=0; i<array.length;i++) {
			list.add(new Integer(array[i]));
		}
		for (int i=0; i<array.length;i++) {
			if(list.contains(numberToFind-array[i])) {
				//it means the list have 1 element where the summ of the
				actual number (array[i]) + the existing in the list is the
				number to find
				System.out.println("Par encontrado:
				"+array[i]+"+"+(numberToFind-array[i])+"="+numberToFind);
				found=true;
			}
		}
		if(!found) {
			System.out.println("No se encontraron pares que sumen: "
			+numberToFind);
		}
	}
}
```

Explicación:

Al igual que el ejemplo original se genera el arreglo aleatorio, pero después se vacía en una lista de enteros llamada “list”, finalmente se recorre esta lista y se utiliza el método *contains* de la clase “*List*” que busca un elemento que exista en la lista, en este caso se busca “numberToFind – array[i]” y en caso de que dicho número exista en la lista se habrá encontrado un par por lo cual se levanta la bandera de “*found*” para evitar mostrar el mensaje de que no se encuentran pares que cumplan con la suma.

Ejecución y prueba de escritorio:

Para el ejemplo se buscan coincidencias que su suma de 10 y un arreglo: [15,11,15,11,16,0,4,9,4,10]

Basado en el algoritmo se recorre la lista que en esencia es un arreglo igual al original y se hacen los siguientes pasos

numberToFind	array[i]	numberToFind-array[i]	Resultado en if
10	15	-5	FALSE
10	11	-1	FALSE
10	15	-5	FALSE
10	11	-1	FALSE
10	16	-6	FALSE
10	0	10	TRUE
10	4	6	FALSE
10	9	1	FALSE
10	4	6	FALSE
10	10	0	TRUE

Ejecución del programa:

```
Se revisará el numero: 10
[15,4,16,8,19,12,0,6,10,17]
Realizando algoritmo de busqueda de pares que sumen: 10
Par encontrado: 4+6=10
Par encontrado: 0+10=10
Par encontrado: 6+4=10
Par encontrado: 10+0=10
```

Usando clase HashTable

```
public class Practica3HashTable {
	public static void main(String[] args) {
		int array[]= new int[10000];
		boolean found=false;
		Integer numberToFind=new Integer(args[0]);
		Hashtable<Integer,Integer> hashTable=new Hashtable<Integer,Integer>();
		System.out.println("Se revisará el numero: "+ numberToFind);
		System.out.print("[");
		for (int i =0; i<array.length;i++) {
			array[i]=(int) (Math.random()*20);
			System.out.print(array[i]+ (i!=(array.length-1)? ",":""));
		}
		System.out.println("]");

		System.out.println("Realizando algoritmo de busqueda de pares que sumen:
		"+numberToFind);
		for (int i=0; i<array.length;i++) {
			hashTable.put(array[i],i); // key is the number and the value is
			the index
		}
		System.out.println("HashTable: "+ hashTable);
		for (int i=0; i<array.length;i++) {
			try {
				int resta= numberToFind- array[i];
				int index= hashTable.get(resta);
				System.out.println("Par encontrado:
				"+array[index]+"+"+(numberToFind-
				array[index])+"="+numberToFind);
				found=true;
			}
			catch(Exception e) {
				continue;
			}
		}
		if(!found) {
			System.out.println("No se encontraron pares que sumen: "
			+numberToFind);
		}
	}
}
```

Explicación:

El código es igual al anterior hasta que se insertan valores en la HashTable, una HashTable tiene la peculiaridad que almacena valores mediante pares clave-valor, de tal manera que se accede al valor mediante la clave, dicha clave es única, es decir no se puede repetir la misma clave en la HashTable, en caso de intentar insertar dos claves iguales se actualizará el valor de la clave pasada sin añadir una clave nueva, para que este algoritmo funcione se almacenará el valor del arreglo como la clave y el índice de donde está ubicado como el valor, ejemplo:

Arreglo: [2,5,7,3,9]

HashTable: [(2,0), (5,1), (7,2), (3,3), (9,4)]

De tal manera que para acceder al valor “0” se usará la llave “2” y el hashTable regresará 0, así como si se usa la llave 5, regresará el valor 1, es decir se obtiene el índice mediante el valor, lo cual puede parecer raro ya que es una manera de invertir el uso común del arreglo.

Una vez llenada la HashTable se procede a realizar un ciclo sobre el arreglo original y se usa la línea “hashTable.get(numberToFind-array[i])!=i” para determinar si dentro del hashTable existe un número que sea la resta del número a buscar menos el número actual del arreglo de existir se puede asegurar que hay un par que números que cumplen la suma (de no existir se controla un error con un bloque try catch que se verá en próximas prácticas).

Visualizar las pruebas de escritorio que se muestran a continuación para que se entienda de una forma más clara cómo funciona el algoritmo.

Ejecución y pruebas de escritorio:

Primero se hará una prueba de escritorio para entender una HashTable, es decir, el ciclo for que tiene la línea *hashTable.put(array[i],i);*

Se generó un array: [7,0,7,6,0,0,15,3,3,13] el cual se insertará en el hashTable:

Índice (i)	*hashTable (llave->valor)*
0	{7->0}
1	{7->0, 0->1}
2	{7->2, 0->1}
3	{7->2, 0->1, 6->3}
4	{7->2, 0->4, 6->3}
5	{7->2, 0->5, 6->3}
6	{7->2, 0->5, 6->3 ,15->6}
7	{7->2, 0->5, 6->3 ,15->6, 3->7}
8	{7->2, 0->5, 6->3 ,15->6, 3->8}
9	{7->2, 0->5, 6->3 ,15->6, 3->8, 13->9}

Como se puede ver arriba, la HashTable termina con 6 elementos ya que no repite llaves, se observa en la tercera iteración donde la llave 7 ya existía y lo que hace la colección es actualizar el valor de 0 a 2, recordar que este valor indica “el índice donde el número 7 es encontrado”, por lo tanto, sólo se necesita almacenar dicha información una vez ya que lo importante es saber que existe un 7 y obtener su índice para poder acceder mediante el arreglo principal a él.

Ahora, se procederá a la segunda parte del algoritmo que es la parte donde se determina si existe ó no un par números que sumados den el número pasado como argumento (de momento ignorar el bloque try-catch, se explicará en futuras prácticas)

```
for (int i=0; i<array.length;i++) {
	try {
		int resta= numberToFind- array[i];
		int index= hashTable.get(resta);
		System.out.println("Par encontrado:
		"+array[index]+"+"+(numberToFind-
		array[index])+"="+numberToFind);
		found=true;
	}
	catch(Exception e) {
		continue;
	}
}
```

Índice (i)	*Resta= numberToFind-array[i]*	*index = hashTable.get(resta)*
0	10-7 = 3	8
1	10-0 = 10	NA
2	10-7 = 3	8
3	10-6 = 4	NA
4	10-0 =10	NA
5	10-0 = 10	NA
6	10-15 = -5	NA
7	10-3 = 7	2
8	10-3 = 7	2
9	10-13 = -3	NA

Como se puede notar en la tabla, los casos donde aparece NA son imposibles debido a que el índice obtenido de la hashTable es inexistente. Por ejemplo, en la iteración 1 se intenta acceder a la llave “10” en la hashTable, dicha llave no existe (revisar la primera tabla de pruebas de escritorio) y, por lo tanto, se descarta que exista un 0 que cumpla con la condición “10-0 = 10”, en caso de que exista un índice se habrá encontrado un par.

```
Se revisará el numero: 10
[7,0,7,6,0,0,15,3,3,13]
Realizando algoritmo de busqueda de pares que sumen: 10
HashTable: {7=2, 6=3, 15=6, 3=8, 13=9, 0=5}
Par encontrado: 3+7=10
Par encontrado: 3+7=10
Par encontrado: 7+3=10
Par encontrado: 7+3=10
```

Mas ejemplos de ejecución del programa:

```
Se revisará el numero: 10
[7,12,10,9,8,5,1,14,15,19]
Realizando algoritmo de busqueda de pares que sumen: 10
HashTable: {19=9, 15=8, 14=7, 12=1, 10=2, 9=3, 8=4, 7=0, 5=5, 1=6}
Par encontrado: 1+9=10
Par encontrado: 5+5=10
Par encontrado: 9+1=10
```

```
Se revisará el numero: 12
[5,12,2,4,7,8,18,13,2,6]
Realizando algoritmo de busqueda de pares que sumen: 12
HashTable: {18=6, 13=7, 12=1, 8=5, 7=4, 6=9, 5=0, 4=3, 2=8}
Par encontrado: 7+5=12
Par encontrado: 8+4=12
Par encontrado: 5+7=12
Par encontrado: 4+8=12
Par encontrado: 6+6=12
```

Si bien este método de solución parece complicar más un problema sencillo se deja al alumno la tarea de comparar los 3 métodos de solución (fuerza bruta, List y HashTable) cambiando la longitud del arreglo principal (int array[]= new int[10];) a valores más grandes como 1000 ó superiores y observar las ventajas de cada método, a su vez se recomienda revisar otras colecciones para intentar dar solución al mismo problema, en prácticas posteriores será requisito usarlas (se indicará en su momento) para distintas tareas.

Práctica 4

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Qué es un objeto?

a) Una variable de tipo primitivo.
b) La representación de un concepto en un programa.
c) Una estructura que almacena solo un tipo de dato.

Opción correcta:

b) La representación de un concepto en un programa. Correcto: un objeto es la idea fundamental de la programación orientada a objetos, siendo ésta la forma en la que el programador puede representar conceptos para desarrollar un programa.

Retroalimentación de las opciones incorrectas:

a) Una variable de tipo primitivo. Incorrecto: ya se definió en prácticas pasadas los tipos primitivos y si bien un objeto puede tener variables primitivas dentro, esto no lo convierte en una variable primitiva.

c) Una estructura que almacena sólo un tipo de datos. Incorrecto: un objeto puede contener variables de diferentes tipos, además de poder contener métodos que definan su comportamiento.

Pregunta 2:

¿Cuál es la definición más acertada de "Clase"?

a) Una plantilla que define la estructura de un concepto de la vida real.
b) Una plantilla que define el comportamiento de un concepto de la vida real.
c) Una plantilla que define la forma y el comportamiento de un concepto de la vida real

Opción correcta:

c) Una plantilla que define la forma y el comportamiento de un concepto de la vida real. Correcto: una clase agrupa una serie de métodos (comportamientos) y atributos (forma) comunes entre una variedad de objetos.

Retroalimentación de las opciones incorrectas:

a) Una plantilla que define la estructura de un concepto de la vida real. Incorrecto: esta respuesta es parcialmente correcta, ya que si bien una clase define la estructura de un concepto, también define sus métodos (el comportamiento del objeto).

b) Una plantilla que define el comportamiento de un concepto de la vida real. Incorrecto: esta respuesta es parcialmente correcta, si bien una clase permite definir el comportamiento de un concepto de la vida real, también define su estructura (atributos).

Pregunta 3:

¿A qué se refiere de instancia?

a) A la creación de un método en una clase.
b) A la creación de un objeto de una clase.
c) Al uso de un método de una clase.

Opción correcta:

b) A la creación de un objeto de una clase. Correcto: a la acción de crear un nuevo objeto de una clase se le "instancia" de esa clase.

Retroalimentación de las opciones incorrectas:

a) A la creación de un método en una clase. Incorrecto: la creación de un método en una clase se define al momento de diseñar una solución para resolver un problema en programación.

c) Al uso de un método de una clase. Incorrecto: no confundir el usar el método de una clase con la "instancia" de una clase, el uso del método se hace mediante una instancia, por instancia se debe entender al objeto creado a partir de una clase.

Pregunta 4:

¿Para qué sirve la sobrecarga de métodos?

a) Para inicializar atributos de una clase.
b) Para crear métodos con el mismo nombre, pero diferentes argumentos.
c) Para el manejo de muchas instancias de una clase.

Opción correcta:

b) Para crear métodos con el mismo nombre, pero diferentes argumentos. Correcto: la sobrecarga permite tener métodos con el mismo nombre, pero diferentes argumentos, esto puede ayudar a añadir versatilidad al programa y reutilizar de código.

Retroalimentación de las opciones incorrectas:

a) Para inicializar atributos de una clase. Incorrecto: la inicialización de los atributos de una clase se puede realizar desde un método constructor el cual puede tener una sobrecarga, sin embargo, la sobrecarga de métodos va más allá de sólo sobrecargar constructores.

c) Para el manejo de muchas instancias de una clase: Incorrecto, la sobrecarga de métodos permite tener un mismo método con diferentes argumentos y así manejar el mismo nombre para realizar tareas distintas dependiendo del tipo ó cantidad de argumentos.

Pregunta 5:

¿Para qué sirve el recolector de basura?

a) Para recuperar la memoria de un objeto sin referencia
b) Para finalizar un ciclo que esté usando recursos.
c) Para destruir un objeto.

Opción correcta:

a) Para recuperar la memoria de un objeto sin referencia. Correcto: la manera en la que java accede a los objetos es mediante una referencia, en caso de que un objeto no tenga referencia no se podrá acceder a él y será tarea del recolector de basura el recuperar este espacio de memoria.

Retroalimentación de las opciones incorrectas:

b) Para finalizar un ciclo que esté usando recursos. Incorrecto: si bien el recolector de basura podría ejecutarse dentro de una estructura de control repetitiva, el ciclo seguirá su flujo normal, el recolector sólo habrá intentado recuperar memoria de un objeto que haya perdido referencia.

c) Para destruir un objeto. Incorrecto: en java no se tiene un método para destruir un objeto, la manera de "destruirlo" se puede conseguir mediante el método *finalize()* ó dependiendo del recolector de basura de java, el cuál intentará recuperar la memoria de un objeto que no tenga referencia (que no se pueda acceder a él).

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno aplique los conocimientos adquiridos hasta el momento para resolver un problema concreto de un sistema mediante la creación de clases y con interacción entre ellas.

Manejo de clases para la creación de un sistema de gestión de un grupo

Un objeto representa la materialización de una clase y la base de todo programa dentro de la programación orientada a objetos, el cual contiene los datos fundamentales y los métodos para la manipulación de éstos.

Una clase equivale a la generalización de un tipo específico de objetos, es decir, es una plantilla ó molde que define las variables y los métodos que son comunes para todos los objetos de un cierto tipo.

Sistema de gestión de alumnos y profesores.

Actividad 1:

Se tienen las clases “Profesor” y “Alumno” como se muestra a continuación:

```
public class Profesor {
    //Atributos
    String nombre;
    String apellidoPaterno;
    String apellidoMaterno;
    String gradoAcademico;
    String numeroEmpleado;
    //Constructor vacio
    public Profesor() {

    }
    //Constructor con argumentos
    public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno, String gradoAcademico,
            String numeroEmpleado) {
        this.nombre = nombre;
        this.apellidoPaterno = apellidoPaterno;
        this.apellidoMaterno = apellidoMaterno;
        this.gradoAcademico = gradoAcademico;
        this.numeroEmpleado = numeroEmpleado;
    }

    //Metodo para mostrar la información del objeto
    @Override
    public String toString() {
        return "Grado Académico: "+gradoAcademico +" nombre: "+nombre +" "+ apellidoPaterno+" "+ apellidoMaterno+ "Numero de empleado: "+ numeroEmpleado;
    }


}
```

```java
public class Alumno {
    String nombre;
    String apellidoPaterno;
    String apellidoMaterno;
    String numeroDeCuenta;

    public Alumno() {
    }

    public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroDeCuenta) {
        this.nombre = nombre;
        this.apellidoPaterno = apellidoPaterno;
        this.apellidoMaterno = apellidoMaterno;
        this.numeroDeCuenta = numeroDeCuenta;
    }

    @Override
    public String toString() {
        return "nombre: " + nombre + " " + apellidoPaterno + " " + apellidoMaterno + "  Numero de cuenta: "
                + numeroDeCuenta;
    }
}
```

Usando las clases dadas realizar un sistema que tenga un menú con las siguientes opciones (El archivo se debe llamar Practica4.java):

- Crear un alumno: se deberá de almacenar el nombre, los apellidos y el número de cuenta del alumno y almacenarlo en un ArrayList de “Alumno”. Antes de regresar al menú se debe mostrar un mensaje de confirmación con la información registrada.
- Crear un profesor: se deberá almacenar el nombre, los apellidos, el grado académico y el número de empleado del profesor y almacenarlo en un ArrayList de “Profesor” . Antes de regresar al menú se debe mostrar un mensaje de confirmación con la información registrada.
- Mostrar Información: Este menú deberá mostrar el estado del sistema actual, es decir, mostrar que profesores y que alumnos se han registrado hasta ese momento (los que se encuentren en los ArrayList correspondientes).
- Salir: Termina la ejecución del programa.

Actividad 2:

Cambiar la estructura de ArrayList para profesores y alumnos por un HashMap ó una HashTable usando como clave el número de empleado ó número de cuenta (según sea el caso) y como valor usar el objeto completo a almacenar de manera que no se pueda almacenar el mismo alumno/profesor dentro de la estructura (revisar material extra de la práctica 3).

Consideraciones:

- En caso de tener problemas con el método “nextInt()” de la clase Scanner usar "nextLine()" y realizar una transformación de datos (parseo) ya sea creando un nuevo objeto a partir de una cadena regresada por nextLine() del tipo necesario, ó usando la clase Wrapper para generar un valor a partir de una cadena:

Ejemplos:

Scanner sc = new Scanner(System.in);
cadena= sc.nextLine();

int numero=new Integer(cadena);
o
int numero = Integer.valueOf(cadena)

Explicación: al usar el método nextInt() (y similares) sólo se ocupa el valor numérico insertado, es decir, al escribir 3 y presionar enter sólo se toma el 3 dejando un salto de línea ('\n') en el scanner, seguido de esto, usar otro método para recibir datos puede resultar en leer este salto de línea que se quedó de la ejecución anterior de manera automática haciendo que sólo se lea dicho salto de línea y seguirá la ejecución del programa, por otro lado nextLine() leerá toda la cadena (incluyendo el salto de línea) dejando limpio el buffer para la siguiente lectura.

- Al usar transformaciones de un tipo de dato a otro, por ejemplo, de String a Integer pueden aparecer errores cuando se intenta transformar una cadena que no representa un número, es decir, la cadena “numero” no tiene un equivalente en el tipo Integer, esto provoca un error que en futuras prácticas se explicará cómo solucionar.
- Para recorrer un HashMap se puede usar el iterador incluido en dichas clases, por ejemplo para un HashMap<String, Object> llamado map:

```
for (Object : map.values()) {
  /*Código a ejecutar, la lógica se lee “para cada objeto del tipo
  Object en los valores del map */
}
```

Entregar en un archivo zip:

- Un documento PDF con las capturas de pantalla de la ejecución del programa en cada opción del menú, se deberán registrar al menos dos alumnos y dos profesores.
- Archivos con extensión java que contenga el código al terminar la actividad 2.

Ejemplo de solución para uso interno del área:

```
public class Practica4B {
    static int opcion=0;
    public static void main(String[] args) {
        HashMap<String, Alumno> alumnos = new HashMap<>();
        HashMap<String, Profesor> profesores = new HashMap<>();
        Scanner sc = new Scanner(System.in);
        do {
            String nombre=null;
            String apellidoP=null;
            String apellidoM= null;
            String cuentaEmpleado=null;
            String gradoAcademico=null;
            System.out.println("=======Sistema de gestión de alumnos para el grupo========== ");
            System.out.println("Selecciona una opción");
            System.out.println("1.- Crear un Alumno");
            System.out.println("2.- Crear un profesor");
            System.out.println("3.- Mostrar información");
            System.out.println("4.- Salir");
            opcion=Integer.valueOf(sc.nextLine());
            System.out.println("opcion seleccionada: "+ opcion);
            switch (opcion) {
            case 1:
                System.out.println("==========Creación de nuevo alumno==========");
                System.out.println("Nombre del alumno:");
                nombre=sc.nextLine();
                System.out.println("Apellido Paterno:");
                apellidoP=sc.nextLine();
                System.out.println("Apellido Materno:");
                apellidoM=sc.nextLine();
                System.out.println("Numero de cuenta:");
                cuentaEmpleado=sc.nextLine();
                Alumno alumnoNuevo= new Alumno(nombre,apellidoP,apellidoM,cuentaEmpleado);
                alumnos.put(alumnoNuevo.numeroDeCuenta, alumnoNuevo);
                System.out.println("Alumno registrado: "+ alumnoNuevo);
                break;
            case 2:
                System.out.println("==========Creación de nuevo Profesor==========");
                System.out.println("Nombre del Profesor:");
                nombre=sc.nextLine();
                System.out.println("Apellido Paterno:");
                apellidoP=sc.nextLine();
                System.out.println("Apellido Materno:");
                apellidoM=sc.nextLine();
                System.out.println("Numero de Empleado:");
                cuentaEmpleado=sc.nextLine();
                System.out.println("Grado Académico:");
                gradoAcademico=sc.nextLine();
                Profesor profesorNuevo= new Profesor(nombre, apellidoP, apellidoM, gradoAcademico, cuentaEmpleado);
                profesores.put(profesorNuevo.numeroEmpleado, profesorNuevo);
                System.out.println("Profesor registrado: "+ profesorNuevo);
                break;
```

```
            case 3:
                System.out.println("Profesores registrados:");
                for (Profesor profesor : profesores.values()) {
                    System.out.println(profesor);
                }
                System.out.println("Alumnos registrados: ");
                for (Alumno alumno : alumnos.values()) {
                    System.out.println(alumno);
                }
                break;
            case 4:
                System.out.println("Fin de la ejecución");
                sc.close();
                break;
            default:
                System.out.println("Opción no valida, intenta de nuevo");
                break;
            }
        }while(opcion!=4);
    }


}
```

Ejecución del sistema:

```
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Mostrar información
4.- Salir
2
opcion seleccionada: 2
==========Creación de nuevo Profesor==========
Nombre del Profesor:
Profesor2
Apellido Paterno:
APP
Apellido Materno:
APM
Numero de Empleado:
987321-456
Grado Académico:
Ingeniero en computación
Profesor registrado: Grado Académico: Ingeniero en computación nombre: Profesor2 APP APMNumero de empleado: 987321-456
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Mostrar información
4.- Salir
3
opcion seleccionada: 3
Profesores registrados:
Grado Académico: Ingeniero en computación nombre: Profesor2 APP APMNumero de empleado: 987321-456
Grado Académico: Maestría en ciencias de la computación nombre: Profesor1 APP APMNumero de empleado: 951753456-2
Alumnos registrados:
nombre: Alumno2 APP APM  Numero de cuenta: 98765432-1
nombre: Alumno1 APP APM  Numero de cuenta: 12345678-9
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Mostrar información
4.- Salir
4
opcion seleccionada: 4
Fin de la ejecución
```

```
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Mostrar información
4.- Salir
1
opcion seleccionada: 1
==========Creación de nuevo alumno==========
Nombre del alumno:
Alumno1
Apellido Paterno:
APP
Apellido Materno:
APM
Numero de cuenta:
12345678-9
Alumno registrado: nombre: Alumno1 APP APM  Numero de cuenta: 12345678-9
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Mostrar información
4.- Salir
1
opcion seleccionada: 1
==========Creación de nuevo alumno==========
Nombre del alumno:
Alumno2
Apellido Paterno:
APP
Apellido Materno:
APM
Numero de cuenta:
98765432-1
Alumno registrado: nombre: Alumno2 APP APM  Numero de cuenta: 98765432-1
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Mostrar información
4.- Salir
2
opcion seleccionada: 2
==========Creación de nuevo Profesor==========
Nombre del Profesor:
Profesor1
Apellido Paterno:
APP
Apellido Materno:
APM
Numero de Empleado:
951753456-2
Grado Académico:
Maestría en ciencias de la computación
Profesor registrado: Grado Académico: Maestría en ciencias de la computación nombre: Profesor1 APP APMNumero de empleado: 951753456-2
```

Rúbrica de evaluación:

Se tendrá que evaluar que el sistema cumpla con lo solicitado que es la posibilidad de almacenar la información de alumnos y profesores y mostrarla después sin posibilidad de que éstos se repitan.

Tomar el código del alumno y compilarlo usando el comando javac *.java en la carpeta entregada por el alumno:

Rúbrica a evaluar	Forma de evaluar	Respuesta esperada	Puntos
Almacenamiento de alumnos	Se deberá usar el menú creado por el alumno para acceder a esta opción y se deberá de crear un alumno conforme se vayan pidiendo los datos, almacenar 2 alumnos	Mensaje de confirmación sobre la creación de dicho alumno para cada registro con sus datos ejemplo: Alumno registrado: nombre: Alumno1 APP APM Número de cuenta: 12345678-9	2
Almacenamiento de profesores	Se deberá usar el menú creado por el alumno para acceder a esta opción y se deberá de crear un profesor conforme se vayan pidiendo los datos, almacenar 2 profesores	Mensaje de confirmación sobre la creación de dicho profesor para cada registro con sus datos ejemplo: Profesor registrado: Grado Académico: Maestría en ciencias de la computación nombre: Profesor1 APP APM Número de empleado: 951753456-2.	2
Mostrar información	Acceder a esta opción mediante el menú del sistema	Se deberá ver la información de los 4 registros, dos profesores y dos alumnos, cualquier otra respuesta es incorrecta.	1
Buen manejo del menú	Insertar una opción no especificada en el menú, por decir si son opciones del 1 al 4, insertar un 5 (no insertar palabras ya que el alumno aún no maneja excepciones)	Un mensaje diciendo que la opción es incorrecta y que permita el seleccionar una opción nuevamente, dar medio punto en caso de que se permita el uso del menú después de una opción no valida pero no aparezca un mensaje de error.	1
Evitar repetición de Alumnos	Registrar un tercer Alumno con el mismo número de cuenta de uno de los dos anteriores pero algún dato diferente (por ejemplo, nombre)	Revisar en el menú de mostrar información que no se haya agregado ningún registro nuevo, se espera que el registro anterior cambie el valor antiguo por el nuevo insertado, en caso de que se cree un registro nuevo se considera un error	2
Evitar repetición de Profesores	Registrar un tercer Profesor con el mismo número de cuenta de uno de los dos anteriores pero algún dato diferente (por ejemplo, nombre)	Revisar en el menú de mostrar información que no se haya agregado ningún registro nuevo, se espera que el registro anterior cambie el valor antiguo por el nuevo insertado, en caso de que se cree un registro nuevo se considera un error	2

Puntos totales: 10

Fórmula para calificar: (aciertos/puntos totales) (10)

Práctica 5

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿A qué se refiere la palabra encapsulamiento?

a) A la acción de ocultar la información de un objeto a los demás.
b) A la acción de modificar la información de un objeto desde otro objeto.
c) A la acción de crear un método dentro de una clase que no regrese valores.

Opción correcta:

a) A la acción de ocultar la información de un objeto a los demás. Correcto: el encapsulamiento permite tener un control de acceso a la información del objeto, ya sea privando su acceso ó mediante métodos de acceso a la misma.

Retroalimentación de las opciones incorrectas:

b) A la acción de modificar la información de un objeto desde otro objeto. Incorrecto: el encapsulamiento es la acción que permite ocultar cierta información de un objeto a los demás objetos, dependiendo del nivel de acceso se pueden modificar ciertos atributos del objeto.

c) A la acción de crear un método dentro de una clase que no regrese valores. Incorrecto: el encapsulamiento tiene que ver con la creación de métodos dentro de una clase para acceder ó manipular la información de éste, sin embargo, no necesariamente tiene (y generalmente no lo es) que ser un método que no regrese valores.

Pregunta 2:

¿A qué se refiere la abstracción?

a) Es un enfoque de solución que parte del nivel de acceso de las variables de un problema.
b) Es un enfoque de solución que parte del nivel más específico de los elementos de un problema y expresa la solución en estos términos.
c) Enfoque de solución que parte del nivel más general de los elementos de un problema y expresar una solución en estos términos.

Opción correcta:

c) Enfoque de solución que parte del nivel más general de los elementos de un problema y expresar una solución en estos términos. Correcto: la abstracción es la acción de captar los elementos más generales de un problema y con base en esto proponer una solución.

Retroalimentación de las opciones incorrectas:

a) Es un enfoque de solución que parte del nivel de acceso de las variables de un problema. Incorrecto: la encapsulación es la encargada de definir el nivel de acceso de las variables de un objeto, mientras la abstracción se enfoca en determinar las partes importantes y generales de un problema.

b) Es un nfoque de solución que parte del nivel más específico de los elementos de un problema y expresa la solución en estos términos. Incorrecto: la abstracción es justo lo contrario, es el concepto encargado de revisar la parte más general de un problema para expresar la solución.

Pregunta 3:

¿Cuántos modificadores de acceso existen en java?

a) 2
b) 3
c) 4

Opción correcta:

b) 3. Correcto: los 3 modificadores de acceso son *public*, *protected* y *private*.

Retroalimentación de las opciones incorrectas:

a) 2. Incorrecto: los modificadores de acceso son 3 (*public*, *protected* y *private)* los cuales permiten proteger ó exponer los datos y métodos de un objeto.

b) 4. Incorrecto: los modificadores de acceso son 3 (*public*, *protected* y *private)* los cuales permiten proteger ó exponer los datos y métodos de un objeto.

Pregunta 4:

¿Cuántos niveles de acceso se tienen en java?

a) 2
b) 3
c) 4

Opción correcta:

c) 4. Correcto: los niveles de acceso son diferentes que los modificadores, es decir los niveles de acceso son *public*, *private*, *protected* y de paquete.

Retroalimentación de las opciones incorrectas:

a) 2. Incorrecto: Java cuenta con 4 niveles de acceso a la información: *public*, *protected*, *private y de paquete*.

b) 3. Incorrecto: Java cuenta con 4 niveles de acceso a la información: *public*, *protected*, *private y de paquete*.

Pregunta 5:

¿Qué opción define el concepto de composición?

a) Objetos con variables primitivas.
b) Objetos con variables privadas.
c) Objetos con referencias a otros objetos.

Opción correcta:

c) Objetos con referencias a otros objetos. Correcto: El concepto de composición está asociado con una idea de "tiene un", lo cual permite diseñar las clases que están compuestas por otros objetos (tiene referencias a otros objetos).

Retroalimentación de las opciones incorrectas:

a) Objetos con variables primitivas. Incorrecto: el concepto de composición implica que una clase tenga como datos miembro referencias a otras clases.

b) Objetos con variables privadas. Incorrecto: el concepto de composición implica que una clase tenga como datos miembro referencias a otras clases, sin importar su nivel de acceso.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno aplique los conocimientos adquiridos sobre encapsulamiento y abstracción para mejorar el diseño del sistema que se tiene hasta el momento.

Sistema de gestión de grupos

El encapsulamiento permite mantener oculta ó controlar el acceso a la información de un objeto.

La composición permite diseñar de clases con estructuras más complejas, se puede considerar una relación “tiene un” como ejemplos: Una casa “tiene un” conjunto de muebles, una casa “tiene un” número de cuartos, etc.

Basándose en el ejercicio de la práctica 4 realizar las siguientes modificaciones:

- Realizar una clase "Grupo", usando el concepto de composición, de tal manera que un objeto de la clase Grupo debe tener un conjunto de alumnos y un profesor asignado, además, se deben incluir métodos getters y setters a esta clase y usarlos para el menú de mostrar información.

- En el menú se deberá crear una opción adicional para “Inscribir alumno al grupo” la cual muestre en pantalla todos los alumnos registrados agregue al grupo seleccionado por número de cuenta del alumno. Dentro de los alumnos registrados en el grupo NO deberán existir duplicados, además, deberá mostrar un mensaje de confirmación al inscribir el alumno. En caso de que el número de cuenta ingresado no exista en los alumnos registrados se deberá mostrar un mensaje indicando que dicho número de cuenta no se ha registrado y no deberá afectar la información del grupo.

- La clase grupo deberá tener un profesor asignado, de manera similar dicho profesor será seleccionable desde los profesores registrados, es decir, se deberá crear un menú adicional con la opción “asignar profesor al grupo” dicha opción deberá mostrar todos los profesores asignados y recibir el número de empleado para poder asignar el profesor al grupo dicho. Al final se debe mostrar un mensaje de confirmación al registrarlo. En caso de intentar asignar un profesor con número de empleado inexistente el sistema deberá mostrar un mensaje indicando que no existe un profesor con ese número de empleado y no deberá afectar la información del grupo.

- Cambiar todos los atributos de la clase Profesor y Alumno para ser privados y sólo se tenga acceso directo a ellos dentro de la clase.

- Usar los métodos getters para recuperar la información necesaria y los métodos setters para asignar la información recibida por la línea de comandos (nombre, apellidos, etcétera) cuando se requiera.

- El menú que mostraba la información registrada deberá mostrar ahora la información almacenada en el grupo.

Considerar:

El orden del menú se deja a consideración del alumno, pero se recomienda el siguiente dado a la naturaleza de las operaciones:

1.- Crear Alumno

2.- Crear Profesor

3.- Inscribir Alumno al grupo

4.- Asignar Profesor al grupo

5.- Mostrar el grupo

6.- Salir

Entregar: un archivo ZIP un documento PDF con capturas de pantalla de la ejecución del programa mostrando el funcionamiento de cada uno de los menús (6 en total), los archivos java: Practica5, Grupo, Alumno y Profesor

Ejemplo para uso interno del área:

```
public class Practica5 {
    static int opcion=0;
    public static void main(String[] args) {
        Grupo grupo = new Grupo();
        HashMap<String, Alumno> alumnos = new HashMap<>();
        HashMap<String, Profesor> profesores = new HashMap<>();
        Scanner sc = new Scanner(System.in);
        do {
            String nombre=null;
            String apellidoP=null;
            String apellidoM= null;
            String cuentaEmpleado=null;
            String gradoAcademico=null;
            System.out.println("=======Sistema de gestión de alumnos para el grupo========== ");
            System.out.println("Selecciona una opción");
            System.out.println("1.- Crear un Alumno");
            System.out.println("2.- Crear un profesor");
            System.out.println("3.- Inscribir Alumno al grupo");
            System.out.println("4.- Asignar profesor al grupo");
            System.out.println("5.- Mostrar el grupo");
            System.out.println("6.- Salir");
            opcion=Integer.valueOf(sc.nextLine());
            System.out.println("opcion seleccionada: "+ opcion);
            switch (opcion) {
            case 1:
                System.out.println("==========Creación de nuevo alumno==========");
                System.out.println("Nombre del alumno:");
                nombre=sc.nextLine();
                System.out.println("Apellido Paterno:");
                apellidoP=sc.nextLine();
                System.out.println("Apellido Materno:");
                apellidoM=sc.nextLine();
                System.out.println("Numero de cuenta:");
                cuentaEmpleado=sc.nextLine();
                Alumno alumnoNuevo= new Alumno(nombre,apellidoP,apellidoM,cuentaEmpleado);
                alumnos.put(alumnoNuevo.getNumeroDeCuenta(), alumnoNuevo);
                System.out.println("Alumno registrado: "+ alumnoNuevo);
                break;
            case 2:
                System.out.println("==========Creación de nuevo Profesor==========");
                System.out.println("Nombre del Profesor:");
                nombre=sc.nextLine();
                System.out.println("Apellido Paterno:");
                apellidoP=sc.nextLine();
                System.out.println("Apellido Materno:");
                apellidoM=sc.nextLine();
                System.out.println("Numero de Empleado:");
                cuentaEmpleado=sc.nextLine();
                System.out.println("Grado Académico:");
                gradoAcademico=sc.nextLine();
                Profesor profesorNuevo= new Profesor(nombre, apellidoP, apellidoM, gradoAcademico, cuentaEmpleado);
                profesores.put(profesorNuevo.getNumeroEmpleado(), profesorNuevo);
                System.out.println("Profesor registrado: "+ profesorNuevo);
                break;
```

```
            case 3:
                String numeroCuenta="";
                System.out.println("==========Inscribir Alumno al Grupo==========");
                System.out.println("Alumnos registrados:");
                for (Alumno alumno : alumnos.values()) {
                    System.out.println(alumno);
                }
                System.out.println("Ingresa el numero de cuenta del alumno que se desea inscribir al grupo");
                numeroCuenta= sc.nextLine();
                if(alumnos.get(numeroCuenta)!=null) {
                    grupo.getAlumnos().put(numeroCuenta, alumnos.get(numeroCuenta));
                    System.out.println("Se ha inscrito al alumno con numero de cuenta: "+ numeroCuenta +" correctamente");
                }
                else {
                    System.out.println("No existe un alumno registrado con el numero de cuenta: "+numeroCuenta);
                }

                break;
            case 4:
                String numeroEmpleado="";
                System.out.println("==========Asignar Profesor al Grupo==========");
                System.out.println("Profesores registrados:");
                for (Profesor profesor : profesores.values()) {
                    System.out.println(profesor);
                }
                System.out.println("Ingresa el numero de empleado del profesor que se desea inscribir al grupo");
                numeroEmpleado= sc.nextLine();
                if (profesores.get(numeroEmpleado)!=null) {
                    grupo.setProfesor(profesores.get(numeroEmpleado));
                    System.out.println("Se ha asignado el profesor con numero de empleado: "+ numeroEmpleado +" correctamente");
                }
                else {
                    System.out.println("No existe un profesor registrado con el numero de empleado: "+numeroEmpleado);
                }

                break;
            case 5:
                System.out.println("Profesores Asignado al grupo:");
                System.out.println(grupo.getProfesor());
                System.out.println("Alumnos inscritos al grupo: ");
                for (Alumno alumno : grupo.getAlumnos().values()) {
                    System.out.println(alumno);
                }
                break;
            case 6:
                System.out.println("Fin de la ejecución");
                sc.close();
                break;
            default:
                System.out.println("Opción no valida, intenta de nuevo");
                break;
            }
        }while(opcion!=6);
    }
}
```

Clases involucradas:

```java
public class Grupo {

    Profesor profesor;
    HashMap<String, Alumno> alumnos = new HashMap<>();

    public void mostrarGrupo() {
        System.out.println("Profesor asignado: " + (profesor!=null?  profesor.getNombre(): "No asignado"));
        System.out.println("Alumnos: ");
        for (Alumno alumno : alumnos.values()) {
            System.out.println(alumno);
        }
    }
    public Profesor getProfesor() {
        return profesor;
    }
    public void setProfesor(Profesor profesor) {
        this.profesor = profesor;
    }
    public HashMap<String, Alumno> getAlumnos() {
        return alumnos;
    }
    public void setAlumnos(HashMap<String, Alumno> alumnos) {
        this.alumnos = alumnos;
    }
}
public class Alumno {
    private String nombre;
    private String apellidoPaterno;
    private String apellidoMaterno;
    private String numeroDeCuenta;
    public Alumno() {
    }
    public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroDeCuenta) {
        this.nombre = nombre;
        this.apellidoPaterno = apellidoPaterno;
        this.apellidoMaterno = apellidoMaterno;
        this.numeroDeCuenta = numeroDeCuenta;
    }
    @Override
    public String toString() {
        return "nombre: " + nombre + " " + apellidoPaterno + " " + apellidoMaterno + "  Numero de cuenta: "
                + numeroDeCuenta;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidoPaterno() {
        return apellidoPaterno;
    }
    public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
    }
    public String getApellidoMaterno() {
        return apellidoMaterno;
    }
    public void setApellidoMaterno(String apellidoMaterno) {
        this.apellidoMaterno = apellidoMaterno;
    }
    public String getNumeroDeCuenta() {
        return numeroDeCuenta;
    }
    public void setNumeroDeCuenta(String numeroDeCuenta) {
        this.numeroDeCuenta = numeroDeCuenta;
    }
}
```

```java
public class Profesor {
    private String nombre;
    private String apellidoPaterno;
    private String apellidoMaterno;
    private String gradoAcademico;
    private String numeroEmpleado;
    public Profesor() {
    }
    public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno, String gradoAcademico,
            String numeroEmpleado) {
        this.nombre = nombre;
        this.apellidoPaterno = apellidoPaterno;
        this.apellidoMaterno = apellidoMaterno;
        this.gradoAcademico = gradoAcademico;
        this.numeroEmpleado = numeroEmpleado;
    }
    @Override
    public String toString() {
        return "Grado Académico: "+gradoAcademico +" nombre: "+nombre +" "+
                apellidoPaterno+" "+ apellidoMaterno+ " Numero de empleado: "+ numeroEmpleado;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidoPaterno() {
        return apellidoPaterno;
    }
    public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
    }
    public String getApellidoMaterno() {
        return apellidoMaterno;
    }
    public void setApellidoMaterno(String apellidoMaterno) {
        this.apellidoMaterno = apellidoMaterno;
    }
    public String getGradoAcademico() {
        return gradoAcademico;
    }
    public void setGradoAcademico(String gradoAcademico) {
        this.gradoAcademico = gradoAcademico;
    }
    public String getNumeroEmpleado() {
        return numeroEmpleado;
    }
    public void setNumeroEmpleado(String numeroEmpleado) {
        this.numeroEmpleado = numeroEmpleado;
    }
}
```

Ejecución:

```
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
1
opcion seleccionada: 1
==========Creación de nuevo alumno==========
Nombre del alumno:
Alumno1
Apellido Paterno:
App
Apellido Materno:
Apm
Numero de cuenta:
12345678-9
Alumno registrado: nombre: Alumno1 App Apm  Numero de cuenta: 12345678-9


=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
1
opcion seleccionada: 1
==========Creación de nuevo alumno==========
Nombre del alumno:
Alumno2
Apellido Paterno:
App
Apellido Materno:
Apm
Numero de cuenta:
98765432-1
Alumno registrado: nombre: Alumno2 App Apm  Numero de cuenta: 98765432-1
```

```
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
2
opcion seleccionada: 2
==========Creación de nuevo Profesor==========
Nombre del Profesor:
Profesor1
Apellido Paterno:
app
Apellido Materno:
apm
Numero de Empleado:
951753-2
Grado Académico:
Ingeniero
Profesor registrado: Grado Académico: Ingeniero nombre: Profesor1 app apmNumero de empleado: 951753-2

opcion seleccionada: 2
==========Creación de nuevo Profesor==========
Nombre del Profesor:
Profesor2
Apellido Paterno:
app
aApellido Materno:
pm
Numero de Empleado:
987zds-4
Grado Académico:
Maestría
Profesor registrado: Grado Académico: Maestría nombre: Profesor2 app apmNumero de empleado: 987zds-4
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
3
opcion seleccionada: 3
==========Inscribir Alumno al Grupo==========
Alumnos registrados:
nombre: Alumno2 App Apm  Numero de cuenta: 98765432-1
nombre: Alumno1 App Apm  Numero de cuenta: 12345678-9
Ingresa el numero de cuenta del alumno que se desea inscribir al grupo
12345678-9
Se ha inscrito al alumno con numero de cuenta: 12345678-9 correctamente
```

```
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
4
opcion seleccionada: 4
==========Asignar Profesor al Grupo==========
Profesores registrados:
Grado Académico: Maestría nombre: Profesor2 app apmNumero de empleado: 987zds-4
Grado Académico: Ingeniero nombre: Profesor1 app apmNumero de empleado: 951753-2
Ingresa el numero de empleado del profesor que se desea inscribir al grupo
987zds-4
Se ha asignado el profesor con numero de empleado: 987zds-4 correctamente
```

```
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
5
opcion seleccionada: 5
Profesores Asignado al grupo:
Grado Académico: Maestría nombre: Profesor2 app apmNumero de empleado: 987zds-4
Alumnos inscritos al grupo:
nombre: Alumno1 App Apm  Numero de cuenta: 12345678-9
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
6
opcion seleccionada: 6
Fin de la ejecución
```

Rúbrica de evaluación:

Se deberá evaluar que el sistema cumpla con lo solicitado que es la posibilidad de mantener almacenado dos tipos diferentes de información, los alumnos y profesores registrados al sistema y los registrados al grupo, evitar repeticiones en ambos registros, evitar el almacenar registros nulos a los datos del grupo, que se muestre la información del grupo y que se usen los métodos get y set al cambiar los atributos a una visibilidad privada.

Rúbrica a evaluar	Forma de evaluar	Respuesta esperada	Puntos
Creación de la clase Grupo	Abrir en un editor de texto la clase “Grupo” entregada por el alumno	La clase debe de tener un atributo de la clase Profesor y un HashMap/HashTable de alumnos con la estructura Key-Value donde Key sea un String y Value “Alumno” y ambos atributos deberán ser privados, además deberá tener los métodos get y set para cada atributo.	1 punto.
Modificación a clase Alumno	Abrir en un editor de texto la clase “Alumno” entregada por el alumno	La clase debe contener todos sus atributos con el modificador “private” y tener los métodos get y set para cada uno	1 punto.
Modificación a clase Profesor	Abrir en un editor de texto la clase “Profesor” entregada por el alumno	La clase debe contener todos sus atributos con el modificador “private” y tener los métodos get y set para cada uno	1 punto.
Funcionamiento del menú Inscribir un Alumno al grupo	Registrar 2 alumnos usando el menú de crear alumno, después usar el menú de inscribir alumno al grupo e inscribir únicamente 1 de los dos, repetir el proceso e intentar inscribir un alumno inexistente.	Se espera poder registrar el alumno que existe y recibir un mensaje de aviso al intentar registrar al alumno que no existe en el sistema	2 puntos (1 punto por registro correcto y 1 punto por el mensaje de aviso al intentar registrar un alumno inexistente)
Funcionamiento del menú Asignar Profesor al Grupo	Registrar al menos 1 profesor en el menú de crear profesor después usar el menú de asignar profesor al grupo y usar dicha opción para asignar un profesor registrado y uno que no exista	Se espera poder asignar un profesor existente mediante su número de empleado y recibir un mensaje de error al intentar registrar un profesor inexistente	2 puntos (un punto por registrar al profesor y uno por el mensaje al ingresar un número de empleado inexistente)
Funcionamiento del menú mostrar grupo	Usar el menú de mostrar información del grupo.	deberá visualizarse la información que se haya asignado hasta el momento (un alumno y un profesor)	5 puntos (2 puntos por la impresión de la información del profesor mediante el objeto “grupo”, ver ejemplo de solución, 2 puntos por la impresión de la información de los alumnos usando el objeto “grupo ver ejemplo de solución, 1 punto extra por el mensaje mostrado en pantalla

Puntos totales: 12
Fórmula para calificar: (aciertos/puntos totales)*(10)

Práctica 6

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Qué es un paquete?

a) Una agrupación de objetos.
b) Una agrupación de clases.
c) Una agrupación de variables.

Opción correcta:

b) Una agrupación de clases. Correcto: para mantener una organización adecuada de un programa se utilizan los paquetes para agrupar clases que tengan un funcionamiento similar ó alguna relación y así mantener un orden fácilmente entendible y escalable.

Retroalimentación de las opciones incorrectas:

a) Una agrupación de objetos. Incorrecto: una colección (como List ó HashMap) podrían considerarse una agrupación de objetos, sin embargo, los paquetes permiten agrupar un conjunto de clases para mantener un orden en el programa.

c) Una agrupación de variables. Incorrecto: una agrupación de variables sería lo equivalente a un arreglo lo cual no tiene que ver con la definición de paquete, éstos últimos permiten mantener un orden ó clasificación en el programa al agrupar clases en un solo lugar.

Pregunta 2:

¿Dónde deben ser escritas las líneas para importar clases de un paquete?

a) Dentro de la declaración de la clase.
b) Después de la declaración de la clase.
c) Antes de la declaración de la clase.

Opción correcta:

c) Antes de la declaración de la clase. Correcto: la importación de clases desde otro paquete debe realizarse antes de la declaración de la clase, es decir tienen que ser las primeras líneas del archivo con el código fuente.

Retroalimentación de las opciones incorrectas:

a) Dentro de la declaración de la clase. Incorrecto: dentro de la declaración de la clase no se pueden importar los paquetes, esto genera un fallo de compilación, las importaciones deben realizarse en las primeras líneas del archivo java, fuera de cualquier clase.

b) Después de la declaración de la clase. Incorrecto: después de la declaración de la clase no se pueden importar los paquetes, esto genera un fallo de compilación, las importaciones deben realizarse en las primeras líneas del archivo java, fuera de cualquier clase.

Pregunta 3:

¿Cuál de los siguientes paquetes se importa automáticamente?

a) java.lang.
b) java.awt.
c) java.math.

Opción correcta:

b) java.lang. Correcto: este paquete incluye clases basicas del lenguaje, como lo son la clase Object ó String. Por ello, no es necesario realizar un import de esta clase ya que se importa automáticamente.

Retroalimentación de las opciones incorrectas:

b) java.awt. Incorrecto: este es un paquete que contiene clases útiles para crear elementos visuales como ventanas y botones en una aplicación java, en caso de ocuparla se tiene que declarar su importación explícitamente.

b) java.math. Incorrecto, este es un paquete que contiene clases que permiten utilizar funciones matemáticas como la potencia (pow), generar números aleatorios (random) entre otros métodos relacionados con el uso de operaciones ó constantes matemáticas, en caso de ocuparla se tiene que declarar su importación explícitamente.

Pregunta 4:

¿Qué argumento se debe usar para indicar al compilador el directorio de salida?

a) -d
b) -a
c) -r

Opción correcta:

a) -d. Correcto: si se desea que el compilador de Java (javac) cree la estructura de las clases (paquetes y clases) en una carpeta específica la ruta se indica mediante el argumento -d.

Retroalimentación de las opciones incorrectas:

b) -a. Incorrecto: si se requiere que el compilador genere la estructura necesaria para el uso de paquetes se debe utilizar el argumento -d.

c) -r. Incorrecto: si se requiere que el compilador genere la estructura necesaria para el uso de paquetes se debe utilizar el argumento -d.

Pregunta 5:

¿Para qué sirve un archivo JAR?

a) Para ordenar las clases de un programa en una aplicación.
b) Para almacenar en un comprimido las clases y archivos necesarios para la ejecución de una aplicación.
c) Para crear las carpetas según la estructura de los paquetes de la aplicación.

Opción correcta:

b) Para almacenar en un comprimido las clases y archivos necesarios para la ejecución de una aplicación. Correcto, un archivo jar empaqueta todos los archivos necesarios (class, imágenes, etc.) en una aplicación java para que ésta pueda ser ejecutada en cualquier equipo que tenga instalada la JVM.

Retroalimentación de las opciones incorrectas:

a) Para ordenar las clases de un programa en una aplicación. Incorrecto: para ordenar las clases de una aplicación utilizan los paquetes, los archivos Jar almacenan dichos paquetes junto con otros archivos necesarios para poder ejecutar la aplicación.

c) Para crear las carpetas según la estructura de los paquetes de la aplicación. Incorrecto: la creación de las carpetas según la estructura de los paquetes es trabajo del compilador, el objetivo del archivo jar es usar la estructura de carpetas y almacenarla en un solo archivo comprimido con el cuál se pueda ejecutar la aplicación en diferentes equipos siempre y cuando se cuente con la JVM.

Pregunta 6:

¿Para qué sirve el archivo MANIFEST?

a) Para declarar los paquetes de la aplicación.
b) Para compilar las clases dentro de la aplicación.
c) Para describir las configuraciones de la aplicación.

Opción correcta:

c) Para describir las configuraciones de la aplicación. Correcto: en el archivo manifest se pueden describir algunas configuraciones de la aplicación, por ejemplo, se puede indicar cuál es la clase que contiene el método main.

Retroalimentación de las opciones incorrectas:

a) Para declarar los paquetes de la aplicación. Incorrecto, el archivo manifest permite escribir las configuraciones de la aplicación, como por ejemplo la clase que contenga el método main.

b) Para compilar las clases dentro de la aplicación. Incorrecto: la compilación de las clases es realizada por javac para clases simples, el archivo manifest permite escribir las configuraciones de la aplicación.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno aplique los conocimientos adquiridos hasta el momento para generar una aplicación con documentación y a su vez haga portable dicha aplicación usando el JDK de java.

Ordenar el sistema en paquetes

La organización en paquetes es una herramienta importante para el desarrollo de aplicaciones ya que permite agrupar clases con cierta estructura ó ó idea común. Además, permite tener una organización que ayuda al programador a encontrar las clases necesarias de manera sencilla e intuitiva.

La documentación de un sistema siempre es importante ya que permite leer de manera concreta la funcionalidad de una clase y de los métodos contenidos en ella, para esto se puede utilizar la herramienta de documentación brindada por java (javadoc) que genera archivos HTML similares al API de Java, además de brindar la posibilidad a los entornos de desarrollo integrado (IDE) de mostrar información relevante al programador al momento de estar codificando.

Reutilzando el sistema que se ha desarrollado hasta ahora, realizar una copia utilizando la versión de la práctica 5 con el nombre Practica6.java :

- Generar la documentación de las clases agregando el autor y las descripciones de las clases junto con los parámetros usados para los métodos que realicen tareas importantes (no es necesario realizar comentarios de getter y setters, el nombre ya es una convección que describe su funcionamiento, pero se pueden realizar si así se desea).

- Ordenar en paquetes las clases del sistema creado en la práctica 5. La jerarquía base de los paquetes será ***'mx.unam.fi.poo'***. En este paquete estará la clase principal (la que contiene al método main). El resto de las clases (Profesor,Alumno y Grupo) deberán estar en un paquete llamado ***mx.unam.fi.poo.*****pojo** esto es debido a que las clases se agrupan por un propósito similar, todos estos paquetes deberán estar almacenados en una carpeta llamada src, es decir la ruta completa de la clase principal será ***src/mx/unam/fi/poo/Practica6.Java***, adicionalmente crear dentro de la carpeta src una carpeta llamada "***clases***" la cual se utilizará para almacenar los archivos generados por la compilación, esto se hace para mantener un mejor orden y distinción entre código fuente y bytecode.

Cuestionario:

Los siguientes comandos se deben ejecutar estando dentro de la carpeta src.

1.- Ejecutar "*javac -d clases mx/unam/fi/poo/Practica6.java*"

¿Qué sucedió?

2.- Cambiar de directorio a la carpeta clases (cd clases) y ejecutar

"*jar -cvfe practica6.jar mx.unam.fi.poo.Practica6 mx/unam/fi/poo/*.class mx/unam/fi/poo/{nombreDefinidoPorElAlumno}}/*.class"*

Seguido de *"java -jar pratica6.jar*"

Explicar que hace el primer comando a detalle y probar que el segundo funcione correctamente.

3.- ¿Cuáles son las ventajas de usar un empaquetado JAR a comparación de usar el compilado de paquetes usando javac?

4.- ¿De qué manera consideras que ayuda el manejo de paquetes al desarrollar una aplicación?

5.- Con lo aprendido hasta el momento, da tus opiniones y conclusiones sobre el desarrollo de programas usando java y la programación orientada a objetos.

Entregar: un archivo ZIP que contenga un PDF con las respuestas del cuestionario, las preguntas que involucren la ejecución de comandos deberán llevar una captura de pantalla de la consola por cada comando ejecutado, añadir al zip el archivo jar generado, los archivos .java y la documentación javadoc, en las siguientes prácticas se le permite al alumno usar un entorno de desarrollo integrado (IDE), si así lo cree conveniente.

Ejemplo para uso interno del área:

Documentación javadoc en código

```
package mx.unam.fi.poo;

import java.util.HashMap;
import java.util.Scanner;
import mx.unam.fi.poo.utils.Alumno;
import mx.unam.fi.poo.utils.Profesor;
import mx.unam.fi.poo.utils.Grupo;
/**
 * Clase que simula un sistema de registro y creación de usuarios para un grupo único
 * @author Juan Carlos Del Camino Rojas
 * @version 1.0
 *
 */
public class Practica5 {
    static int opcion=0;
    /**
     * Metodo Principal de la aplicación
     * @param args se pueden enviar argumentos, sin embaro se ignoran en la aplicación
     */
    public static void main(String[] args) {
        Grupo grupo = new Grupo();
        HashMap<String, Alumno> alumnos = new HashMap<>();
        HashMap<String, Profesor> profesores = new HashMap<>();
        Scanner sc = new Scanner(System.in);

/**
 * Clase que describe la estructura de un Alumno : nombre, apellidos y numero de
cuenta
 * @author Juan Carlos Del Camino Rojas
 * @version 1.0
 *
 */
public class Alumno {
       private String nombre;
       private String apellidoPaterno;
       private String apellidoMaterno;
       private String numeroDeCuenta;

       public Alumno() {
       }
       /**
        * Constructor con argumentos para la creación de un alumno
        * @param nombre - El nombre del Alumno
        * @param apellidoPaterno - El apellido paterno del alumno
        * @param apellidoMaterno - el apellido materno del alumno
        * @param numeroDeCuenta - el numero de cuenta del alumno
        * */
       public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno,
String numeroDeCuenta) {
             this.nombre = nombre;
             this.apellidoPaterno = apellidoPaterno;
             this.apellidoMaterno = apellidoMaterno;
```

```
		this.numeroDeCuenta = numeroDeCuenta;
	}

	/**
	 * Sobreescritura del metodo to string para mostrar la información del alumno
	 * @return la información del alumno en el formato "nombre:
	{nombre+apellidoPaterno+ApellidoMaterno} Numero de cuenta: {numeroDeCuenta}
	 * */
	@Override
	public String toString() {
		return "nombre: "+nombre +" "+ apellidoPaterno+" "+ apellidoMaterno+ "
		Numero de cuenta: "+ numeroDeCuenta;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public void setApellidoPaterno(String apellidoPaterno) {
		this.apellidoPaterno = apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public void setApellidoMaterno(String apellidoMaterno) {
		this.apellidoMaterno = apellidoMaterno;
	}
	public String getNumeroDeCuenta() {
		return numeroDeCuenta;
	}
	public void setNumeroDeCuenta(String numeroDeCuenta) {
		if(numeroDeCuenta.length()>9) {
			System.out.println("El numero de cuenta no puede ser mayor a 9
			digitos, se trunca la cadena");
			this.numeroDeCuenta = numeroDeCuenta.substring(0, 9);

		}
		else {
			this.numeroDeCuenta = numeroDeCuenta;
		}
	}
}
```

```java
/**
 * Clase que describe la estructura de un Alumno : nombre, apellidos y numero de cuenta
 * @author Juan Carlos Del Camino Rojas
 * @version 1.0
 *
 */
public class Grupo {
    private Profesor profesor;
    HashMap<String, Alumno> alumnos = new HashMap<>();
    /**
     * Metodo para mostrar el grupo en la consola, muestra el profesor asignado y la lista de alumnos del grupo
     * */
    public void mostrarGrupo() {
        System.out.println("Profesor asignado: " + (profesor!=null?  profesor.getNombre(): "No asignado"));
        System.out.println("Alumnos: ");
        for (Alumno alumno : alumnos.values()) {
            System.out.println(alumno);
        }
    }
```

```java
public class Profesor {

	private String nombre;
	private String apellidoPaterno;
	private String apellidoMaterno;
	private String gradoAcademico;
	private String numeroEmpleado;

	public Profesor() {
	}

	/**
	 * Constructor con argumentos para la creación de un profesor
	 * @param nombre - El nombre del Alumno
	 * @param apellidoPaterno - El apellido paterno del profesor
	 * @param apellidoMaterno - el apellido materno del profesor
	 * @param gradoAcademico - el maximo grado de estudios de este profesor
	 * @param numeroEmpleado - el numero de cuenta del profesor
	 * */
	public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno,
String gradoAcademico,
			String numeroEmpleado) {
		this.nombre = nombre;
		this.apellidoPaterno = apellidoPaterno;
		this.apellidoMaterno = apellidoMaterno;
		this.gradoAcademico = gradoAcademico;
		this.numeroEmpleado = numeroEmpleado;
	}

	/**
	 * Sobreescritura del metodo to string para mostrar la información del profesor
	 * @return la información del alumno en el formato:
	 * "Grado Academico:{gradoAcademico}
	 * nombre:{nombre+apellidoPaterno+ApellidoMaterno}
	 * Numero de empleado: {numeroDeEmpleado}"
	 * */
```

```
	@Override
	public String toString() {
		return "Grado Académico: "+gradoAcademico +" nombre: "+nombre +" "+
		apellidoPaterno +" "+ apellidoMaterno+ "Numero de empleado: "+
		numeroEmpleado;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public void setApellidoPaterno(String apellidoPaterno) {
		this.apellidoPaterno = apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public void setApellidoMaterno(String apellidoMaterno) {
		this.apellidoMaterno = apellidoMaterno;
	}
	public String getGradoAcademico() {
		return gradoAcademico;
	}
	public void setGradoAcademico(String gradoAcademico) {
		this.gradoAcademico = gradoAcademico;
	}
	public String getNumeroEmpleado() {
		return numeroEmpleado;
	}
	public void setNumeroEmpleado(String numeroEmpleado) {
		if(numeroEmpleado.length()>9) {
			System.out.println("El numero de empleado no puede ser mayor a 9
caracteres, se trunca la cadena ");
			this.numeroEmpleado = numeroEmpleado.substring(0, 9);

		}
		else {
			this.numeroEmpleado = numeroEmpleado;
		}
	}
}
```

Documentación javadoc en navegador web

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES ALL CLASSES

A G M P S T

A

Alumno - Class in mx.unam.fi.poo.utils
Clase que describe la estructura de un Alumno : nombre, apellidos y numero de cuenta
Alumno() - Constructor for class mx.unam.fi.poo.utils.Alumno

Alumno(String, String, String, String) - Constructor for class mx.unam.fi.poo.utils.Alumno
Constructor con argumentos para la creación de un alumno

G

getAlumnos() - Method in class mx.unam.fi.poo.utils.Grupo

getApellidoMaterno() - Method in class mx.unam.fi.poo.utils.Alumno

getApellidoMaterno() - Method in class mx.unam.fi.poo.utils.Profesor

getApellidoPaterno() - Method in class mx.unam.fi.poo.utils.Alumno

getApellidoPaterno() - Method in class mx.unam.fi.poo.utils.Profesor

getGradoAcademico() - Method in class mx.unam.fi.poo.utils.Profesor

getNombre() - Method in class mx.unam.fi.poo.utils.Alumno

getNombre() - Method in class mx.unam.fi.poo.utils.Profesor

getNumeroDeCuenta() - Method in class mx.unam.fi.poo.utils.Alumno

getNumeroEmpleado() - Method in class mx.unam.fi.poo.utils.Profesor

getProfesor() - Method in class mx.unam.fi.poo.utils.Grupo

Grupo - Class in mx.unam.fi.poo.utils
Clase que describe la estructura de un Alumno : nombre, apellidos y numero de cuenta
Grupo() - Constructor for class mx.unam.fi.poo.utils.Grupo

M

main(String[]) - Static method in class mx.unam.fi.poo.Practica5
Metodo Principal de la aplicación
mostrarGrupo() - Method in class mx.unam.fi.poo.utils.Grupo
Metodo para mostrar el grupo en la consola, muestra el profesor asignado y la lista de alumnos del grupo
mx.unam.fi.poo - package mx.unam.fi.poo

Rúbrica de evaluación

- Se evaluará que las 4 clases (Practica6, Alumno, Profesor y Grupo) tengan su documentación javadoc agregando autor, descripción de la clase y al menos un método documentado, se dará 1 punto por clase con un total de 4 y un punto más si es que hay al menos un método documentado en alguna de las 4 para un total de 5

Cuestionario:

Numero de pregunta	Respuesta esperada	Puntos
1	"Se crearon las clases compiladas dentro del directorio clases" ó ó una frase que signifique lo mismo que la anterior y deberá llevar su captura de pantalla	1 punto (cero puntos en caso de que no anexe la captura de pantalla)
2	"El primer comando genera el archivo jar indicando el nombre de practica6 y dando como argumento la clase que contiene el método principal (practica5) además de dar las rutas de donde se encuentran los archivos .class compilados anteriormente" ó ó una respuesta similar, deberá incluirse una captura de pantalla	1 punto (cero puntos en caso de que no anexe las capturas de pantalla)
3	"Que se genera un archivo único que se puede ejecutar ya que contiene toda la información necesaria para su correcto funcionamiento" ó ó una respuesta similar	1 punto
4	"Permite el tener una estructura al proyecto para mantener un orden y permite la reutilización de código de manera organizada" ó ó una respuesta similar	1 punto
5	Respuesta abierta ya que son las conclusiones del alumno, deberá tener una descripción de lo que piensa del desarrollo de java usando lo aprendido hasta el momento	1 punto

Capturas de pantalla esperadas

Pregunta 1:

```
A:\Desarrollo\WorkSpace\PracticasPoo\src>javac -d clases mx/unam/fi/poo/Practica5.java

A:\Desarrollo\WorkSpace\PracticasPoo\src>
```

Pregunta 2

```
A:\Desarrollo\WorkSpace\PracticasPoo\src\clases>jar -cvfe practica6.jar mx.unam.fi.poo.Practica5 mx/unam/fi/poo/*.cla
ss mx/unam/fi/poo/utils/*.class
manifiesto agregado
agregando: mx/unam/fi/poo/Practica5.class(entrada = 4883) (salida = 2266)(desinflado 53%)
agregando: mx/unam/fi/poo/utils/Alumno.class(entrada = 1700) (salida = 815)(desinflado 52%)
agregando: mx/unam/fi/poo/utils/Grupo.class(entrada = 1907) (salida = 909)(desinflado 52%)

A:\Desarrollo\WorkSpace\PracticasPoo\src\clases>
```

```
A:\Desarrollo\WorkSpace\PracticasPoo\src\clases>java -jar practica6.jar
=======Sistema de gestión de alumnos para el grupo==========
Selecciona una opción
1.- Crear un Alumno
2.- Crear un profesor
3.- Inscribir Alumno al grupo
4.- Asignar profesor al grupo
5.- Mostrar el grupo
6.- Salir
```

Puntos totales:

Revisión de comentarios javadoc en código =5
Cuestionario= 5

Calificación: ((puntos javadoc + puntos cuestionario) / 10) *10

Práctica 7

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿De qué clase heredan todas las clases creadas en java?

a) Object.
b) String.
c) Main.

Opción correcta:

a) Object. Correcto: todas las clases creadas en java heredan implícitamente de la clase Object y se pueden comportar como objetos, que es la base del paradigma.

Retroalimentación de las opciones incorrectas:

b) String. Incorrecto: la clase String es una clase que, implícitamente, hereda de Object por lo cual se puede comportar como un objeto, siendo la clase Object principal para que todas las clases se comporten como objeto.

c) Main. Incorrecto: no confundir el método main con una clase, el método main es el primero en ejecutarse en una aplicación java, sin embargo, todas las clases que utilice una aplicación son una subclase de la clase Object.

Pregunta 2:

¿Qué es la herencia?

a) Es el proceso de creación de métodos a partir de clases.

b) Es el proceso de creación de clases a partir de métodos

c) Es el proceso de creación de clases a partir de otras clases.

Opción correcta:

c) Es el proceso de creación de clases a partir de otras clases. Correcto: la herencia es un proceso que permite generar subclases de una clase base, esto permite crear una nueva clase sin necesidad de reescribir todo el código de su superclase, además de permitir el extender la funcionalidad de ésta.

Retroalimentación de las opciones incorrectas:

a) Es el proceso de creación de métodos a partir de clases. Incorrecto: este proceso no puede ser realizado debido a que un método pertenece a una clase, es decir, se puede crear un método dentro de una clase, pero no a partir de la misma. La herencia permite la creación de una nueva clase a partir de una ya existente, la nueva clase podrá utilizar los atributos y métodos de la clase que hereda.

b) Es el proceso de creación de clases a partir de métodos. Incorrecto: un método pertenece a una clase, pero no permite definir una clase. La herencia permite la creación de una nueva clase a partir de una ya existente donde la nueva clase pueda tener acceso a métodos y atributos de su superclase.

Pregunta 3:

De los siguientes métodos, ¿Cuáles se heredan de la clase Object ?

a) equals, toString, hashCode.
b) get, set, delete.
c) create, clone, has.

Opción correcta:

a) equals, toString, hashCode. Correcto: equals, toString y hashCode son métodos que se heredan de la clase Object y sirven para comparar un objeto con otro, convertir la información del objeto en una cadena y para obtener un identificador único de objeto, respectivamente.

Retroalimentación de las opciones incorrectas:

a) get, set, delete. Incorrecto: estos métodos pueden llegar a ser comunes en la programación pero no se heredan de la clase Object, debido a que su implementación es opcional. Los métodos equals, toString y hashCode son métodos que toda clase posee ya que son heredados de la clase Object.

c) create, clone, has: Incorrecto, clone es un método que se hereda de la clase Object, sin embargo, has y créate son métodos específicos de una clase y pueden ó ó no existir. Los métodos equals, toString y hashCode que toda clase posee ya que son heredados de la clase Object.

Pregunta 4:

¿Cuál es una de las condiciones que se debe de cumplir para sobre escribir un método heredado?

a) El valor de retorno siempre debe ser nulo.
b) Debe tener el mismo tipo de dato pero diferente número de parámetros.
c) Debe tener el mismo nombre.

Opción correcta:

c) Debe tener el mismo nombre. Correcto: para sobre escribir un método heredado se debe cumplir con ciertas condiciones: el método debe tener el mismo nombre, debe tener el mismo tipo y número de parámetros, su nivel de acceso debe ser igual ó ó más accesible y debe tener el mismo tipo de retorno ó ó algún subtipo.

Retroalimentación de las opciones incorrectas:

a) El valor de retorno siempre debe ser nulo: Incorrecto, el valor de retorno puede ser nulo **únicamente** cuando el método heredado tenga este tipo de retorno. Para sobre escribir un método heredado: el método debe tener el mismo nombre, debe tener el mismo tipo y número de parámetros, su nivel de acceso debe ser igual ó ó más accesible y debe tener el mismo tipo de retorno ó ó algún subtipo .

b) Debe tener el mismo tipo de dato pero diferente número de parámetros. Incorrecto: los requisitos que se deben de cumplir para sobrescribir un método son: el método debe tener el mismo nombre, debe tener el mismo tipo y número de parámetros, su nivel de acceso debe ser igual ó más accesible y debe tener el mismo tipo de retorno ó algún subtipo.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno haya entendido el concepto de herencia y sobre escritura de métodos.

Llenar los espacios en blanco para finalizar el código

La herencia permite la reutilización de código y la extensión de funcionalidad de un código previamente realizado, la herencia permite que haya métodos que se puedan sobrescribir ya que se heredan de la superclase, esto es útil cuando se desea que la funcionalidad de la subclase sea diferente (más específica).

Se tiene el siguiente código con espacios en blanco, rellenar dichos espacios con la opción correcta del siguiente listado para generar una compilación y funcionamiento correcto además de seleccionar la correcta salida al ejecutar la clase main.

Opciones de código:

- @Override
- extends
- implements
- MiembroUnam
- Profesor

Opciones de salida de ejecución:

- Profesor: MiembroUnam [nombre=Profesor, apellidoPaterno=APP, apellidoMaterno=APM, numeroIdentificacion=Profesor-1] Grado Academico: Ingeniería en computación
  Alumno: MiembroUnam [nombre=Alumno, apellidoPaterno=APM, apellidoMaterno=APM, numeroIdentificacion=Alumno-1] carrera Ingeniería en computación
- Profesor: Grado Academico: Ingeniería en computación
  Alumno: carrera Ingeniería en computación

Código principal:

```
public class Practica7 {
	public static void main(String[] args) {
		Profesor profesor= new Profesor("Profesor", "APP", "APM", "Profesor-1",
"Ingeniería en computación");
		Alumno alumno = new Alumno("Alumno", "APM", "APM", "Alumno-1",
"Ingeniería en computación");
		System.out.println(profesor);
		System.out.println(alumno);
	}
}
```

Clase MiembroUnam

```java
public class MiembroUnam {

    private String nombre;
    private String apellidoPaterno;
    private String apellidoMaterno;
    private String numeroIdentificacion;
    public MiembroUnam() {
    }
    public MiembroUnam(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroIdentificacion) {
        super();
        this.nombre = nombre;
        this.apellidoPaterno = apellidoPaterno;
        this.apellidoMaterno = apellidoMaterno;
        this.numeroIdentificacion = numeroIdentificacion;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidoPaterno() {
        return apellidoPaterno;
    }
    public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
    }
    public String getApellidoMaterno() {
        return apellidoMaterno;
    }
    public void setApellidoMaterno(String apellidoMaterno) {
        this.apellidoMaterno = apellidoMaterno;
    }
    public String getNumeroIdentificacion() {
        return numeroIdentificacion;
    }
    public void setNumeroIdentificacion(String numeroIdentificacion) {
        this.numeroIdentificacion = numeroIdentificacion;
    }

    public String toString() {
        return "MiembroUnam [nombre=" + nombre + ", apellidoPaterno=" + apellidoPaterno + ", apellidoMaterno="
                + apellidoMaterno + ", numeroIdentificacion=" + numeroIdentificacion + "]";
    }

}
```

Clase Alumno

```
public class Alumno extends [redacted] {
    private String carrera;

    public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroIdentificacion,String carrera) {
        super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
        this.carrera = carrera;
    }
    public String getCarrera() {
        return carrera;
    }
    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }

    @Override
    public String toString() {
        String datosMiembroUnam=super.toString();
        return "Alumno: " + datosMiembroUnam +" carrera "+ carrera;

    }
}
```

Clase Profesor

```
public class Profesor [redacted]MiembroUnam {
    private String gradoAcademico;
    public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroIdentificacion,String gradoAcademico) {
        super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
        this.gradoAcademico = gradoAcademico;
    }
    public String getGradoAcademico() {
        return gradoAcademico;
    }

    public void setGradoAcademico(String gradoAcademico) {
        this.gradoAcademico = gradoAcademico;
    }
    @Override
    public String toString() {
        String datosMiembroUnam=super.toString();
        return "Profesor: " + datosMiembroUnam +" Grado Academico: "+ gradoAcademico;
    }
}
```

Solución y retroalimentación

1.- En la clase “MiembroUnam”

- **Respuesta correcta**: “**@Override**”. Correcto: la etiqueta override permite identificar la sobre escritura de métodos heredados de la superclase, en este caso se sobre escribe el método toString para personalizar la conversión del objeto a una cadena.
- Respuesta incorrecta: “**extends**”. Incorrecto: la palabra reservada extends se utiliza para indicar herencia, utilizarla en este contexto provocaría un error de sintaxis ya que extends se debe utilizar en la declaración de la clase, la respuesta correcta es @Override que permite sobre escribir el comportamiento del método *toString()* heredado de la clase Object.

- Respuesta incorrecta: “**implements**”. Incorrecto: la palabra reservada implements se utiliza al momento de la declaración de clases para utilizar interfaces (este tema se abordará más adelante). La respuesta correcta es @Override que permite sobre escribir el comportamiento del método *toString()* heredado de la clase Object.
- Respuesta incorrecta: “**MiembroUnam**”, incorrecto, el usar esta opción en el espacio vacío provocará un error de compilación, la respuesta correcta es @Override que permite sobre escribir el comportamiento del método *toString()* heredado de la clase Object.
- Respuesta incorrecta: “**Profesor**”, incorrecto, el usar esta opción en el espacio vacío provocará un error de compilación, la respuesta correcta es @Override que permite sobre escribir el comportamiento del método *toString()* heredado de la clase Object.

2.- En la clase “Alumno”

- **Respuesta correcta**: “**MiembroUnam**”. Correcto: la clase Alumno será una subclase de la clase MiembroUnam, esto se indica con la palabra reservada extends seguida de la clase de la cual se desea heredar atributos y métodos.
- Respuesta Incorrecta: “**@Override**”, incorrecto, la etiqueta override permite identificar métodos sobre escritos en un código. Esta instrucción en esta ubicación provocaría un error de sintaxis, la respuesta correcta es “MiembroUnam” ya que es la clase que va a ser la superclase de Alumno.
- Respuesta incorrecta: “**extends**”. Incorrecto: la palabra reservada extends se utiliza para indicar herencia, si se utiliza en este contexto se genera un error de sintaxis. La respuesta correcta es “MiembroUnam” ya que es la clase que va a ser la superclase de Alumno.
- Respuesta incorrecta: “**implements**”. Incorrecto: la palabra reservada implements se utiliza al momento de la declaración de clases para utilizar interfaces (este tema se abordará más adelante). La respuesta correcta es “MiembroUnam” ya que es la clase que va a ser la superclase de Alumno.
- Respuesta incorrecta: “**Profesor**”. Incorrecto: esto provocaría un error de sintaxis, en el constructor ya que se está utilizando el constructor de la clase base para generar un constructor en la subclase. La respuesta correcta es “MiembroUnam” ya que es la clase que va a ser la superclase de Alumno.

3.- En la clase “Profesor”

- Respuesta correcta: “**extends**”. Correcto: la palabra reservada extends se utiliza para indicar que la clase actual va a heredar de otra clase. Esto se puede leer como que un profesor “es un” MiemboUnam.

- Respuesta incorrecta: “**MiembroUnam**”. Incorrecto: esto provocaría un error de sintaxis, la respuesta correcta es extends debido a que se pretende cumplir con la lógica de que “un Profesor es un MiemboUnam”.
- Respuesta Incorrecta: “**@Override**”. Incorrecto, la etiqueta override sirve para indicar la sobre escritura de métodos. En esta línea generaría un error de sintaxis, la respuesta correcta es extends debido a que se pretende cumplir con la lógica de que “un Profesor es un MiemboUnam”.
- Respuesta incorrecta: “**implements**”. Incorrecto: la palabra reservada implements se utiliza al momento de la declaración de clases para utilizar interfaces (este tema se abordará más adelante). La respuesta correcta es extends debido a que se pretende cumplir con la lógica de que “un Profesor es un MiemboUnam”.
- Respuesta incorrecta: “**Profesor**”, incorrecto, esto provocaría un error de sintaxis, La respuesta correcta es extends debido a que se pretende cumplir con la lógica de que “un Profesor es un MiemboUnam”.

4.- Salida de la ejecución:

- Respuesta correcta: “**Profesor: MiembroUnam [nombre=Profesor, apellidoPaterno=APP, apellidoMaterno=APM, numeroIdentificacion=Profesor-1] Grado Academico: Ingeniería en computación**
  Alumno: MiembroUnam [nombre=Alumno, apellidoPaterno=APM, apellidoMaterno=APM, numeroIdentificacion=Alumno-1] carrera Ingeniería en computación”, Correcto, esto se logra debido a que el método que muestra la información concatena los datos provenientes de la superclase y los de la subclase
- Respuesta incorrecta: “**Profesor: Grado Academico: Ingeniería en computación Alumno: carrera Ingeniería en computación**”: Incorrecto, los datos de la superclase también se imprimen en el método toString ya que se accede a dicha información usando el método “super()” y se concatenan ambas informaciones.

Códigos completos:

```
public class Practica7 {
	public static void main(String[] args) {
		Profesor profesor= new Profesor("Profesor", "APP", "APM", "Profesor-1",
		"Ingeniería en computación");
		Alumno alumno = new Alumno("Alumno", "APM", "APM", "Alumno-1",
		"Ingeniería en computación");
		System.out.println(profesor);
		System.out.println(alumno);
	}
}

public class Alumno extends MiembroUnam{
	private String carrera;
	public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno,
	String numeroIdentificacion,String carrera) {
		super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
		this.carrera = carrera;
	}
	public String getCarrera() {
		return carrera;
	}
	public void setCarrera(String carrera) {
		this.carrera = carrera;
	}
	@Override
	public String toString() {
		String datosMiembroUnam=super.toString();
		return "Alumno: " + datosMiembroUnam +" carrera "+ carrera;
	}
}

public class Profesor extends MiembroUnam {
	private String gradoAcademico;
	public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno,
	String numeroIdentificacion,String gradoAcademico) {
		super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
		this.gradoAcademico = gradoAcademico;
	}
	public String getGradoAcademico() {
		return gradoAcademico;
	}
	public void setGradoAcademico(String gradoAcademico) {
		this.gradoAcademico = gradoAcademico;
	}
	@Override
	public String toString() {
		String datosMiembroUnam=super.toString();
		return "Profesor: " + datosMiembroUnam +" Grado Academico: "+
		gradoAcademico;
	}
}
```

```java
public class MiembroUnam {
	private String nombre;
	private String apellidoPaterno;
	private String apellidoMaterno;
	private String numeroIdentificacion;
	public MiembroUnam() {
	}
	public MiembroUnam(String nombre, String apellidoPaterno, String
	apellidoMaterno, String numeroIdentificacion) {
		super();
		this.nombre = nombre;
		this.apellidoPaterno = apellidoPaterno;
		this.apellidoMaterno = apellidoMaterno;
		this.numeroIdentificacion = numeroIdentificacion;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public void setApellidoPaterno(String apellidoPaterno) {
		this.apellidoPaterno = apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public void setApellidoMaterno(String apellidoMaterno) {
		this.apellidoMaterno = apellidoMaterno;
	}
	public String getNumeroIdentificacion() {
		return numeroIdentificacion;
	}
	public void setNumeroIdentificacion(String numeroIdentificacion) {
		this.numeroIdentificacion = numeroIdentificacion;
	}
	@Override
	public String toString() {
		return "MiembroUnam [nombre=" + nombre + ", apellidoPaterno=" +
		apellidoPaterno + ", apellidoMaterno="+ apellidoMaterno + ",
		numeroIdentificacion=" + numeroIdentificacion + "]";
	}

}
```

Práctica 8

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿En qué consiste el polimorfismo?

a) Propiedad por la cual un objeto de una clase base se puede comportar como una de sus subclases.
b) Propiedad por la cual 2 ó más objetos pueden pertenecer a una misma clase.
c) Propiedad por la cual un objeto de una superclase se comporta diferente a una de sus subclases.

Opción correcta:

a) Propiedad por la cual un objeto de una clase base se puede comportar como una de sus subclases. Correcto: el polimorfismo es una capacidad del lenguaje para permitir que un objeto se comporte (cambie de forma) como un objeto de una de sus subclases.

Retroalimentación de las opciones incorrectas:

b) Propiedad por la cual 2 ó más objetos pueden pertenecer a una misma clase. Incorrecto: que dos objetos pertenezcan a una misma clase no tiene que ver con polimorfismo, el polimorfismo no iguala objetos, si no que permite que un objeto se comporte igual que una de sus subclases

c) Propiedad por la cual un objeto de una superclase se comporta diferente a una de sus subclases. Incorrecto: el polimorfismo es justo lo contrario, un objeto de una superclase se puede comportar como una de sus subclases.

Pregunta 2:

De las siguientes, ¿cuál es una peculiaridad de las clases abstractas?

a) Que no se pueden crear objetos de ellas.
b) Que no pueden regresan un valor.
c) Que implementan únicamente métodos que no regresan valores.

Opción correcta:

a) Que no se pueden crear objetos de ellas. Correcto: Una clase abstracta no se puede instanciar debido a que no tiene una implementación completa.

Retroalimentación de las opciones incorrectas:

b) Que no pueden regresan un valor. Incorrecto: una clase no regresa valores, esa es tarea de los métodos, pero un método abstracto puede ó no regresar un valor específico, la respuesta correcta es que no se pueden crear objetos de dichas clases debido a que su implementación no está completa.

c) Que implementan únicamente métodos que no regresan valores. Incorrecto: los métodos abstractos pueden definir valores de retorno, la peculiaridad de una clase abstracta es que no se pueden crear objetos de ella debido a que su implementación no está definida en su totalidad.

Pregunta 3:

¿Cuál es una característica de una interfaz?

a) Todos los métodos están implementados.
b) Todos los métodos regresan un tipo de dato.
c) Todos los métodos son abstractos.

Opción correcta:

c) Todos los métodos son abstractos. Correcto: una interfaz permite definir los métodos de una clase sin decir cómo, la clase que implemente dicha interfaz debe definir el comportamiento de los métodos.

Retroalimentación de las opciones incorrectas:

a) Todos los métodos están implementados. Incorrecto: justamente una interfaz se caracteriza, por lo contrario, ningún método está implementado en una interfaz debido a que sólo es un contrato sobre el qué se puede hacer, mas no el cómo.

b) Todos los métodos regresan un tipo de dato. Incorrecto: una interfaz puede declarar métodos que pueden ó no regresar algún valor, lo que no puede hacer una interfaz es implementar el bloque de código de los métodos que defina.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno comprenda los conceptos de polimorfismo, interfaz y métodos abstractos para la reutilización y que pueda ampliar la funcionalidad del código fuente ya existente.

Rellenar los espacios en blanco para generar un programa funcional

El polimorfismo, las interfaces y los métodos abstractos permiten crear un programa que sea fácilmente reutilizable ya que se puede implementar de diferentes maneras el mismo código sin necesidad de hacer todo desde cero, esto además permite la fácil integración de una ó más clases en diferentes aplicaciones.

La clase main se muestra a continuación

```
public class Practica8 {
	public static void main(String[] args) {
		Profesor profesor= new Profesor("Profesor", "APP", "APM", "Profesor-1",
		"Ingeniería en computación");
		Alumno alumno = new Alumno("Alumno", "APM", "APM", "Alumno-1",
		"Ingeniería en computación");
		System.out.println("====Información del alumno====");
		System.out.println(alumno);
		System.out.println("información del objeto" + alumno.informacion());
		System.out.println("Actividades del alumno:");
		System.out.println(alumno.temporadaInscripciones());
		System.out.println(alumno.semestre());
		System.out.println(alumno.temporadaExamenes());
		System.out.println(alumno.intersemestral());
		System.out.println("====Información del Profesor====");
		System.out.println(profesor);
		System.out.println("infromación del objeto "+ profesor.informacion());
		System.out.println("Actividades del profesor:");
		System.out.println(profesor.temporadaInscripciones());
		System.out.println(profesor.semestre());
		System.out.println(profesor.temporadaExamenes());
		System.out.println(profesor.intersemestral());
	}
}
```

Colocar las líneas de código necesarias dentro del siguiente código para obtener la funcionalidad observada en la salida de la ejecución del programa usando las siguientes opciones opciones dadas:

- interface
- public String informacion();
- extends MiembroUnam implements Actividades
- implements Actividades
- extends Actividades
- abstract class
- public abstract String informacion();

Se muestra la respuesta de la ejecución del programa.

```
====Información del alumno====
Alumno: MiembroUnam [nombre=Alumno, apellidoPaterno=APM, apellidoMaterno=APM,
numeroIdentificacion=Alumno-1] carrera Ingeniería en computación
información del objetoAlumno
Actividades del alumno:
pago de inscripción, revisión de grupos y planeación de horario
Tomar clases y entregar tareas
Estudio constante y realización de examenes
Tomar cursos, descansar y prepararse para el siguiente semestre
====Información del Profesor====
Profesor: MiembroUnam [nombre=Profesor, apellidoPaterno=APP, apellidoMaterno=APM,
numeroIdentificacion=Profesor-1] Grado Academico: Ingeniería en computación
infromación del objeto Profesor
Actividades del profesor:
Preparación del curso que impartirá y obtención de listas de alumnos
Impartir clases y calificar tareas
creación de examenes y calificar los mismos
entregar califcaciones al sistema, dar y tomar cursos
```

```
public           Actividades {
    public String temporadaInscripciones();
    public String temporadaExamenes();
    public String semestre();
    public String intersemestral();
}
```

```
public class Alumno                                    {
    private String carrera;

    public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroIdentificacion,String carrera) {
        super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
        this.carrera = carrera;
    }
    public String getCarrera() {
        return carrera;
    }
    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }
    @Override
    public String toString() {
        String datosMiembroUnam=super.toString();
        return "Alumno: " + datosMiembroUnam +" carrera "+ carrera;
    }
    public String informacion() {
        return "Alumno";
    }
    public String temporadaInscripciones() {
        return "pago de inscripción, revisión de grupos y planeación de horario";
    }
    public String temporadaExamenes() {
        return "Estudio constante y realización de examenes";
    }
    public String semestre() {
        return "Tomar clases y entregar tareas";
    }
    public String intersemestral() {
        return "Tomar cursos, descansar y prepararse para el siguiente semestre";
    }
}
```

```java
public class Profesor extends MiembroUnam {
    private String gradoAcademico;
    public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroIdentificacion,String gradoAcademico) {
        super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
        this.gradoAcademico = gradoAcademico;
    }
    public String getGradoAcademico() {
        return gradoAcademico;
    }

    public void setGradoAcademico(String gradoAcademico) {
        this.gradoAcademico = gradoAcademico;
    }
    @Override
    public String toString() {
        String datosMiembroUnam=super.toString();
        return "Profesor: " + datosMiembroUnam +" Grado Academico: "+ gradoAcademico;
    }
    public String informacion() {
        return "Profesor";
    }
    public String temporadaInscripciones() {
        return "Preparación del curso que impartirá y obtención de listas de alumnos";
    }
    public String temporadaExamenes() {
        return "creación de examenes y calificar los mismos";
    }
    public String semestre() {
        return "Impartir clases y calificar tareas";
    }
    public String intersemestral() {
        return "entregar califcaciones al sistema, dar y tomar cursos";
    }
}
```

```java
public MiembroUnam {
    private String nombre;
    private String apellidoPaterno;
    private String apellidoMaterno;
    private String numeroIdentificacion;
    public MiembroUnam() {
    }
    public MiembroUnam(String nombre, String apellidoPaterno, String apellidoMaterno, String numeroIdentificacion) {
        super();
        this.nombre = nombre;
        this.apellidoPaterno = apellidoPaterno;
        this.apellidoMaterno = apellidoMaterno;
        this.numeroIdentificacion = numeroIdentificacion;
    }

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidoPaterno() {
        return apellidoPaterno;
    }
    public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
    }
    public String getApellidoMaterno() {
        return apellidoMaterno;
    }
    public void setApellidoMaterno(String apellidoMaterno) {
        this.apellidoMaterno = apellidoMaterno;
    }
    public String getNumeroIdentificacion() {
        return numeroIdentificacion;
    }
    public void setNumeroIdentificacion(String numeroIdentificacion) {
        this.numeroIdentificacion = numeroIdentificacion;
    }
    @Override
    public String toString() {
        return "MiembroUnam [nombre=" + nombre + ", apellidoPaterno=" + apellidoPaterno + ", apellidoMaterno="
                + apellidoMaterno + ", numeroIdentificacion=" + numeroIdentificacion + "]";
    }

}
```

Solución y retroalimentación

1.- Clase “Actividades”:

- Respuesta correcta: “**interface**”. Correcto: una interfaz permite la declaración de métodos sin definir su implementación. Las interfaces se declaran con la palabra reservada interface antes del nombre de la interfaz.
- Respuesta incorrecta: “**public String informacion();**”, incorrecto, esta línea generaría un error de sintaxis ya que indica la declaración de un método y éste estaría fuera del cuerpo de la clase. La respuesta correcta es “interface” ya que esta palabra reservada permite declarar un archivo como una interfaz cuyos métodos sean abstractos y sin implementación.
- Respuesta incorrecta: “**extends MiembroUnam implements Actividades**”. Incorrecto: esta línea provocaría un error de sintaxis, la respuesta correcta es “interface”, el uso de esta palabra reservada permite declarar una interfaz cuyos métodos sean abstractos y sin implementación.
- Respuesta incorrecta: “**implements Actividades”**. Incorrecto: esta línea carece de sentido y provocaría un error de sintaxis, la respuesta correcta es “interface”, el uso de esta palabra reservada permite declarar una interfaz cuyos métodos sean abstractos y sin implementación.
- Respuesta incorrecta: “**extends Actividades**”. Incorrecto: esta línea provocaría un error de sintaxis además de carecer de sentido, la palabra reservada “interface” permite declarar una interfaz cuya peculiaridad es que todos sus métodos son abstractos y sin implementación.
- Respuesta incorrecta: “**abstract class**”: Incorrecto, esta línea provoca un error de sintaxis dentro del cuerpo entre llaves debido a que se convertiría en una clase abstracta y los métodos definidos dentro de ella deberán de ser abstractos (de manera explícita) para que no estén definidos, la respuesta correcta es “interface” que permite definir los métodos de manera abstracta sin necesidad de la palabra reservada *abstract*, además, una interfaz no permite definir la implementación los métodos declarados en ella.
- Respuesta incorrecta: “**public abstract String informacion();**”, Incorrecto, esta línea provocaría un error de sintaxis, , la respuesta correcta es “interface”, el uso de esta palabra reservada permite declarar una interfaz cuyos métodos sean abstractos y sin implementación.

2.- Clase “Alumno”:

- Respuesta incorrecta: “**interface**”. Incorrecto: esto provocaría un fallo de sintaxis además de carecer de lógica, la respuesta correcta es “extends MiembroUnam implements Actividades” para que la clase alumno herede de “MiembroUnam” y además implemente la interfaz “Actividades”.
- Respuesta incorrecta: “**public String informacion();**”. Incorrecto: esta línea generaría un error de sintaxis ya que se estaría definiendo un método fuera del cuerpo de la clase, la respuesta correcta es “extends MiembroUnam implements Actividades” que permite que la clase alumno herede de “MiembroUnam” e implemente la interfaz “Actividades”.
- **Respuesta correcta**: “**extends MiembroUnam implements Actividades**”. Correcto: esta línea permite conseguir el funcionamiento esperado, para la clase Alumno herede los métodos y atributos de MiembroUnam y que implemente los métodos de la interfaz “Actividades”.
- Respuesta incorrecta: “**implements Actividades”**, incorrecto, esta línea tiene sentido, pero provocaría un error de sintaxis al invocar al constructor con argumentos ya que no estaría disponible por no heredarlo de “MiembroUnam”

- Respuesta incorrecta: “**extends Actividades**”. Incorrecto: esta línea provoca un error de sintaxis debido a que no se puede heredar una interfaz, la respuesta correcta es “extends MiembroUnam implements Actividades”, esta línea permite conseguir heredar de MiembroUnam e implementar la interfaz Actividades.
- Respuesta incorrecta: “**abstract class**”: Incorrecto, esta línea provoca un error de sintaxis, la respuesta correcta es “extends MiembroUnam implements Actividades” que permite obtener la funcionalidad deseada.
- Respuesta incorrecta: “**public abstract String informacion();**”, Incorrecto, esta línea provocaría un error de sintaxis, la respuesta correcta es “extends MiembroUnam implements Actividades” que permite obtener la funcionalidad deseada mostrada en la ejecución.

3.- Clase Profesor

- Respuesta incorrecta: “**interface**”. Incorrecto: esto provocaría un fallo de sintaxis, la respuesta correcta es “implements Actividades” ya que se busca que un profesor implemente los métodos definidos en la interfaz Actividades.
- Respuesta incorrecta: “**public String informacion();**”, incorrecto, esta línea generaría un error de sintaxis ya que un método no se puede declarar en ese lugar, la respuesta correcta es “implements Actividades” que permite alcanzar la funcionalidad buscada en la ejecución del programa.
- Respuesta incorrecta: “**extends MiembroUnam implements Actividades**”, Incorrecto: esta línea provocaría un error de sintaxis. Por otro lado “implements

Actividades" permite alcanzar la funcionalidad deseada, que es implementar las actividades que el profesor realiza mediante la interfaz Actividades.

- **Respuesta correcta**: "**implements Actividades"**. Correcto: esta línea permite realizar la funcionalidad deseada que es implementar los métodos declarados en la interfaz actividades dentro del cuerpo de la clase Profesor.
- Respuesta incorrecta: "**extends Actividades**". Incorrecto: Java no permite la herencia múltiple entre clases, además, no se puede heredar una interfaz, en este caso la respuesta correcta es "implements Actividades" que permite definir la implementación de los métodos declarados en la interfaz en el cuerpo de la clase Profesor.
- Respuesta incorrecta: "**abstract class**". Incorrecto: esta línea provoca un error de sintaxis, la respuesta correcta es "implements Actividades" que permite obtener la funcionalidad mostrada.
- Respuesta incorrecta: "**public abstract String informacion();**". Incorrecto: esta línea provocaría un error de sintaxis, la respuesta correcta es "implements Actividades" que permite obtener la funcionalidad mostrada en la ejecución.

4.- Clase MiembroUnam primer recuadro

- Respuesta incorrecta: "**interface**". Incorrecto: esta línea tiene sentido por sí sola, pero provocaría un fallo de sintaxis ya que la clase tiene métodos definidos y una interfaz no puede implementar métodos, la respuesta correcta es "abstract class" que permite tener métodos abstractos y métodos concretos en la clase.
- Respuesta incorrecta: "**public String informacion();**", incorrecto, esta línea generaría un error de sintaxis, la respuesta correcta es "abstract class" que permite tener métodos abstractos y métodos concretos declarados en la.
- Respuesta incorrecta: "**extends MiembroUnam implements Actividades**", incorrecto, esta línea provocaría un error de sintaxis ya que carece de sentido, la respuesta correcta es "abstract class" que permite tener métodos abstractos y métodos concretos declarados en la clase.
- Respuesta incorrecta: "**implements Actividades"**, Incorrecto, esta línea provocaría un error de sintaxis debido a que no es el lugar correcto para implementar interfaces, además de no cumplir con el objetivo del resultado de la ejecución del programa, la respuesta correcta es "abstract class" que permite tener métodos abstractos y métodos concretos declarados en la clase.
- Respuesta incorrecta: "**extends Actividades**": Incorrecto, esta línea provocaría un error de sintaxis debido a que no es el lugar indicado para heredar una clase, además de que Actividades es una interfaz y ésta no puede ser heredada debido a la falta de implementación de sus métodos, la respuesta correcta es "abstract class" que permite tener métodos abstractos y métodos concretos declarados en la clase.

- **Respuesta correcta**: “**abstract class**”. Correcto: esta línea permite declarar la clase como abstracta lo cual permite tener métodos abstractos y métodos concretos. Los métodos abstractos deberán implementarse mediante la subclase que herede a esta clase.
- Respuesta incorrecta: “**public abstract String informacion();**”, Incorrecto, esta línea provocaría un error de sintaxis ya que es una declaración de un método abstracto y debería ir dentro del cuerpo de la clase, la respuesta correcta es “abstract class” que permite tener métodos abstractos y métodos concretos declarados en la clase.

5.- Clase MiembroUnam segundo recuadro

- Respuesta incorrecta: “**interface**”, incorrecto, esta palabra reservada no puede utilizarse en este contexto, la respuesta correcta es “public abstract String informacion();” que declara un método abstracto que se usa en el sistema para mostrar la información de los objetos que heredan de esta clase, siendo diferente para cada una de las subclases.
- Respuesta incorrecta: “**public String informacion();**”, Incorrecto, para que un método pueda carecer de definición requiere que sea abstracto, la respuesta correcta es “public abstract String información()” que permite dejar la implementación de dicho método en responsabilidad de las subclases de MiembroUnam.
- Respuesta incorrecta: “**extends MiembroUnam implements Actividades**”, incorrecto, esta línea provocaría un error de sintaxis además de carecer de sentido, la respuesta correcta es “public abstract String informacion();” que declara un método abstracto que se usa en el sistema para mostrar la información de los objetos. Su implementación es diferente para cada una de las subclases.
- Respuesta incorrecta: “**implements Actividades”**, incorrecto, esta línea genera un error de sintaxis ya que no es el contexto indicado para implementar interfaces, la respuesta correcta es “public abstract String información ();” que permite declarar un método abstracto que se usa en el programa para mostrar la información de los objetos. Su implementación es diferente para cada una de las subclases.
- Respuesta incorrecta: “**extends Actividades**”, incorrecto, esta opción genera un error de sintaxis debido a que no es el contexto indicado para heredar clases, además de que Actividades es una interfaz y no se puede heredar se debe implementar, la respuesta correcta es “public abstract String informacion();” que declara un método abstracto que se usa en el sistema para mostrar la información de los objetos que heredan de la clase MiembroUnam, siendo diferente para cada una de las subclases.
- Respuesta incorrecta: “**abstract class**”, incorrecto, esta línea provocaría un error de sintaxis debido a que sólo se puede usar en la declaración de la clase y no en el cuerpo de ésta. La respuesta correcta es “public abstract String información ();”

que permite declarar un método abstracto que se usa en el programa para mostrar la información de los objetos que heredan de la clase MiembroUnam, siendo diferente para cada una de las subclases .

- **Respuesta correcta**: "**public abstract String informacion();**" correcto, esta línea declara un método abstracto que se usa en el programa para mostrar la información de los objetos que heredan el método de la clase MiembroUnam, siendo diferente para cada una de las subclases.

Códigos completos:

Clase main

```
public class Practica8 {
	public static void main(String[] args) {
		Profesor profesor= new Profesor("Profesor", "APP", "APM", "Profesor-1",
		"Ingeniería en computación");
		Alumno alumno = new Alumno("Alumno", "APM", "APM", "Alumno-1",
		"Ingeniería en computación");
		System.out.println("====Información del alumno====");
		System.out.println(alumno);
		System.out.println("información del objeto" + alumno.informacion());
		System.out.println("Actividades del alumno:");
		System.out.println(alumno.temporadaInscripciones());
		System.out.println(alumno.semestre());
		System.out.println(alumno.temporadaExamenes());
		System.out.println(alumno.intersemestral());
		System.out.println("====Información del Profesor====");
		System.out.println(profesor);
		System.out.println("infromación del objeto "+ profesor.informacion());
		System.out.println("Actividades del profesor:");
		System.out.println(profesor.temporadaInscripciones());
		System.out.println(profesor.semestre());
		System.out.println(profesor.temporadaExamenes());
		System.out.println(profesor.intersemestral());
	}
}
```

Interfaz Actividades:

```
 public interface Actividades {
	public String temporadaInscripciones();
	public String temporadaExamenes();
	public String semestre();
	public String intersemestral();
}
```

Clase Alumno

```java
public class Alumno extends MiembroUnam implements Actividades{
	private String carrera;
	public Alumno(String nombre, String apellidoPaterno, String apellidoMaterno,
	String numeroIdentificacion,String carrera) {
		super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
		this.carrera = carrera;
	}
	public String getCarrera() {
		return carrera;
	}
	public void setCarrera(String carrera) {
		this.carrera = carrera;
	}
	@Override
	public String toString() {
		String datosMiembroUnam=super.toString();
		return "Alumno: " + datosMiembroUnam +" carrera "+ carrera;
	}
	public String informacion() {
		return "Alumno";
	}
	public String temporadaInscripciones() {
		return "pago de inscripción, revisión de grupos y planeación de horario";
	}
	public String temporadaExamenes() {
		return "Estudio constante y realización de examenes";
	}
	public String semestre() {
		return "Tomar clases y entregar tareas";
	}
	public String intersemestral() {
		return "Tomar cursos, descansar y prepararse para el siguiente semestre";
	}
}
```

Clase Profesor:

```java
public class Profesor extends MiembroUnam implements Actividades {
	private String gradoAcademico;
	public Profesor(String nombre, String apellidoPaterno, String apellidoMaterno,
	String numeroIdentificacion,String gradoAcademico) {
		super(nombre, apellidoPaterno, apellidoMaterno, numeroIdentificacion);
		this.gradoAcademico = gradoAcademico;
	}
	public String getGradoAcademico() {
		return gradoAcademico;
	}
	public void setGradoAcademico(String gradoAcademico) {
		this.gradoAcademico = gradoAcademico;
	}
	@Override
	public String toString() {
		String datosMiembroUnam=super.toString();
		return "Profesor: " + datosMiembroUnam +" Grado Academico: "+
		gradoAcademico;
	}
	public String informacion() {
		return "Profesor";
	}
	public String temporadaInscripciones() {
		return "Preparación del curso que impartirá y obtención de listas de
		alumnos";
	}
	public String temporadaExamenes() {
		return "creación de examenes y calificar los mismos";
	}
	public String semestre() {
		return "Impartir clases y calificar tareas";
	}
	public String intersemestral() {
		return "entregar califcaciones al sistema, dar y tomar cursos";
	}
}
```

Clase MiembroUnam

```java
public abstract class MiembroUnam {
	private String nombre;
	private String apellidoPaterno;
	private String apellidoMaterno;
	private String numeroIdentificacion;
	public MiembroUnam() {
	}
	public MiembroUnam(String nombre, String apellidoPaterno, String
	apellidoMaterno, String numeroIdentificacion) {
		super();
		this.nombre = nombre;
		this.apellidoPaterno = apellidoPaterno;
		this.apellidoMaterno = apellidoMaterno;
		this.numeroIdentificacion = numeroIdentificacion;
	}
	public abstract String informacion();
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public void setApellidoPaterno(String apellidoPaterno) {
		this.apellidoPaterno = apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public void setApellidoMaterno(String apellidoMaterno) {
		this.apellidoMaterno = apellidoMaterno;
	}
	public String getNumeroIdentificacion() {
		return numeroIdentificacion;
	}
	public void setNumeroIdentificacion(String numeroIdentificacion) {
		this.numeroIdentificacion = numeroIdentificacion;
	}
	@Override
	public String toString() {
		return "MiembroUnam [nombre=" + nombre + ", apellidoPaterno=" +
		apellidoPaterno + ", apellidoMaterno=" + apellidoMaterno + ",
		numeroIdentificacion=" + numeroIdentificacion + "]";
	}

}
```

Práctica 9

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Para qué sirve un diagrama de caso de uso?

a) Define la manera en la que el usuario interactúa con el sistema.
b) Muestra la interacción entre varios objetos y el orden entre ellos.
c) Muestra el flujo de acciones y los objetos involucrados.

Opción correcta:

a) Define la manera en la que el usuario interactúa con el sistema. Correcto: el diagrama de casos de uso sirve para visualizar la manera en la que el usuario del interactúa con el sistema, mostrando las posibles actividades según el rol que se tenga.

Retroalimentación de las opciones incorrectas:

b) Muestra la interacción entre varios objetos y el orden entre ellos: Incorrecto, este uso es propio del diagrama de comunicación, la respuesta correcta es “Define la manera en la que el usuario del sistema interactúa con éste”.

c) Muestra el flujo de acciones y los objetos involucrados: Incorrecto, esta definición le pertenece al diagrama de actividades, la respuesta correcta es “Define la manera en la que el usuario del sistema interactúa con éste”.

Pregunta 2:

¿Para qué sirve un diagrama de actividades?

a) Define la manera en la que el usuario del sistema interactúa con éste.
b) Permite modelar las características de las clases que componen al sistema.
c) Muestra el flujo de acciones y los objetos involucrados.

Opción correcta:

c) Muestra el flujo de acciones y los objetos involucrados. Correcto: el diagrama de actividades sirve para mostrar el flujo de acciones y objetos involucrados en éstas.

Retroalimentación de las opciones incorrectas:

a) Define la manera en la que el usuario del sistema interactúa con éste: Incorrecto, esto es la definición del diagrama de caso de uso, la respuesta correcta es “muestra el flujo de acciones y los objetos involucrados”.

b) Permite modelar las características de las clases que componen al sistema: Incorrecto, esta es la definición del diagrama de clases, la respuesta correcta es “muestra el flujo de acciones y los objetos involucrados”.

Pregunta 3:

¿Para qué sirve un diagrama de secuencia?

a) Muestra la interacción entre objetos y el orden entre ellos.
b) Muestra la interacción ordenada de eventos.
c) Describe las transiciones por las que puede transitar un objeto.

Opción correcta:

b) Muestra la interacción ordenada de eventos. Correcto, el diagrama de secuencia permite visualizar la interacción ordenada de eventos en un sistema ó solución.

Retroalimentación de las opciones incorrectas:

a) Muestra la interacción entre objetos y el orden entre ellos: Incorrecto, esta es la definición de un diagrama de comunicación, la respuesta correcta es “Muestra la interacción ordenada de eventos”.

c) Describe las transiciones por las que puede transitar un objeto: Incorrecto, esta es la definición de un diagrama de estado, la respuesta correcta es “Muestra la interacción ordenada de eventos”.

Pregunta 4:

¿Para qué sirve un diagrama de objetos?

a) Describen las transiciones por las que puede pasar un objeto.
b) Modelan las instancias generadas en un momento de la ejecución.
c) Describe las transiciones por las que puede pasar un objeto.

Opción correcta:

b) Modelan las instancias generadas en un momento de la ejecución. Correcto, el diagrama de objetos modela las instancias generadas en un momento de la ejecución para describir al sistema en una acción en particular.

Retroalimentación de las opciones incorrectas:

a) Describen las transiciones por las que puede pasar un objeto: Incorrecto, esta es la definición de un diagrama de estados, la respuesta correcta es “Modelan las instancias generadas en un momento de la ejecución”.

c) Describe las transiciones por las que puede pasar un objeto: Incorrecto, esta es la definición de un diagrama de estado, la respuesta correcta es “Modelan las instancias generadas en un momento de la ejecución”.

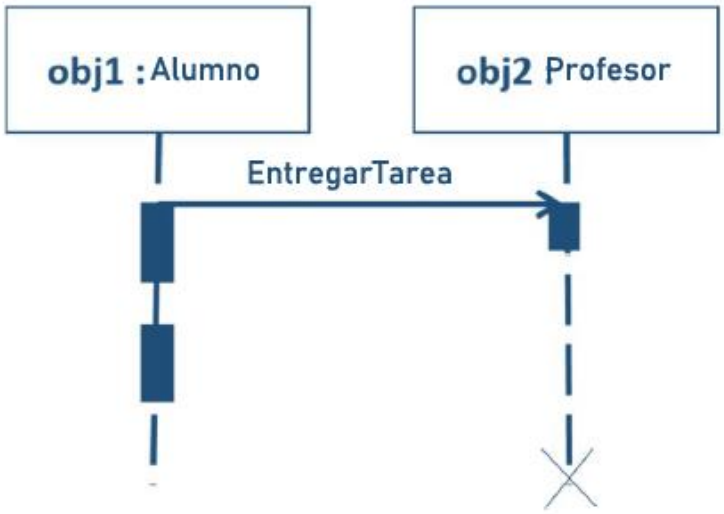
Actividad que genera calificación:

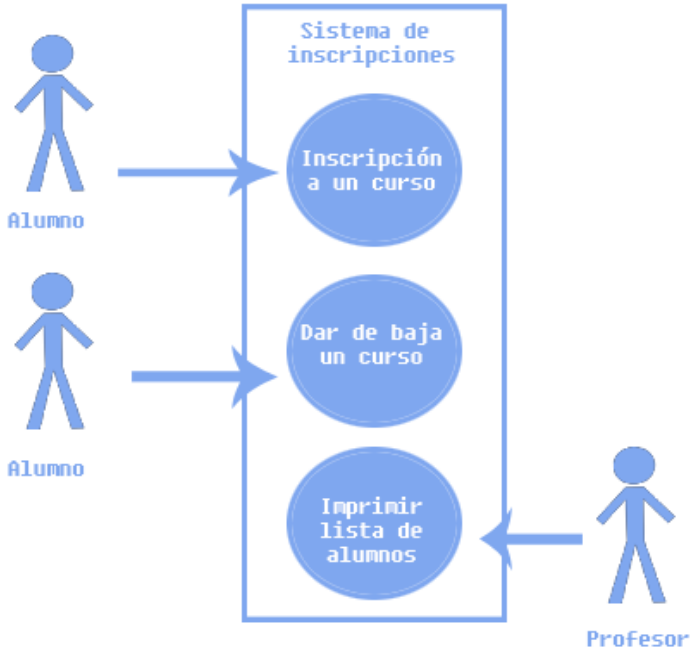
Objetivo de la evaluación: Que el alumno identifique satisfactoriamente como se realiza cada uno de los diagramas UML, para que pueda modelar un sistema utilizando los diagramas que se requieran.

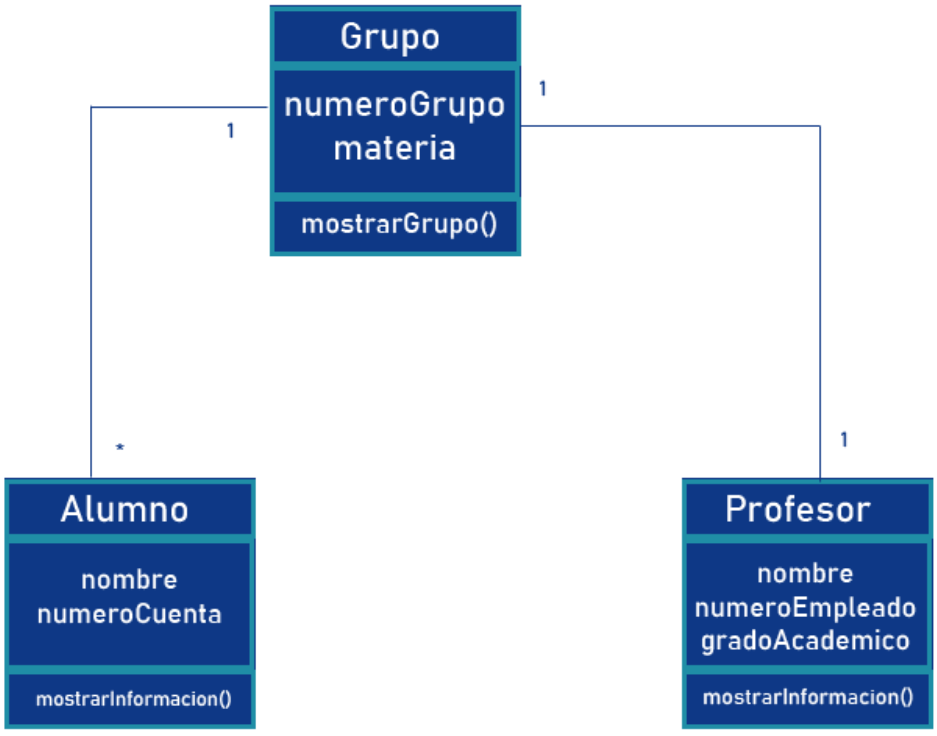
Identificar los diagramas UML

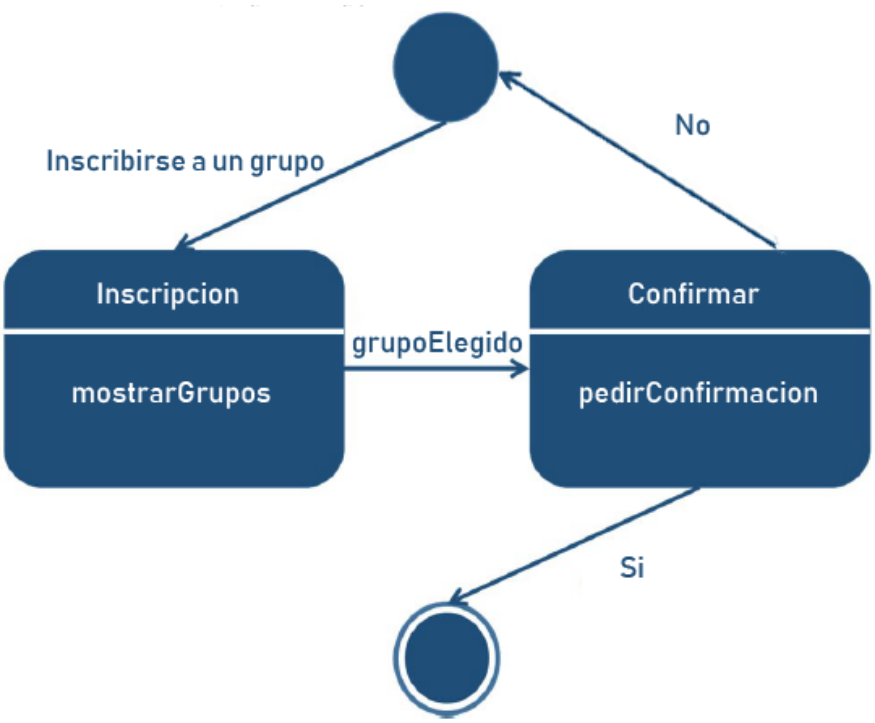
UML es un lenguaje que permite documentar sistemas de manera gráfica, esto sirve para que cualquiera con persona pueda entender las actividades del sistema y, después, sea capaz de ampliar su funcionamiento ó darle mantenimiento.

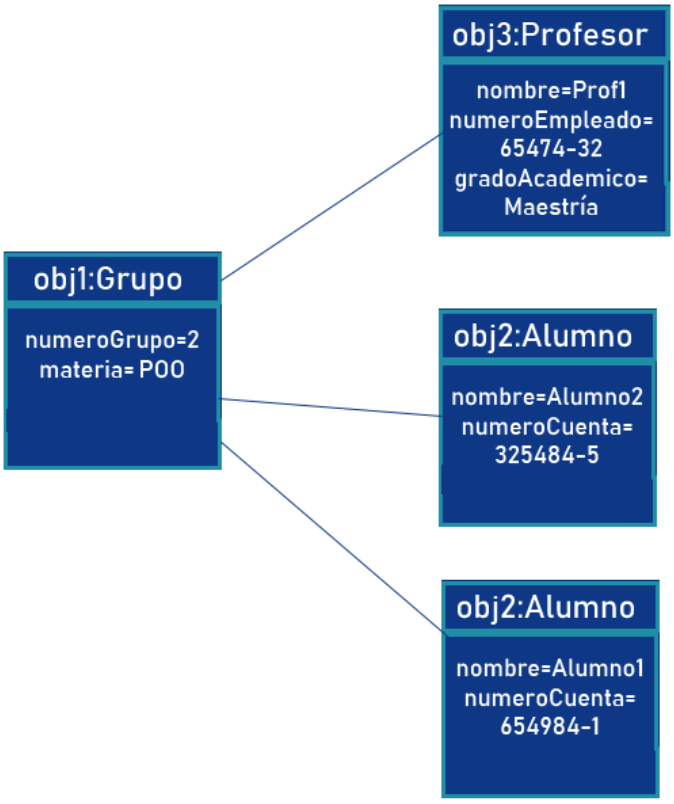
Dados los siguientes diagramas UML, arrastrar a su recuadro correspondiente cada uno de los diagramas UML mostrados a continuación



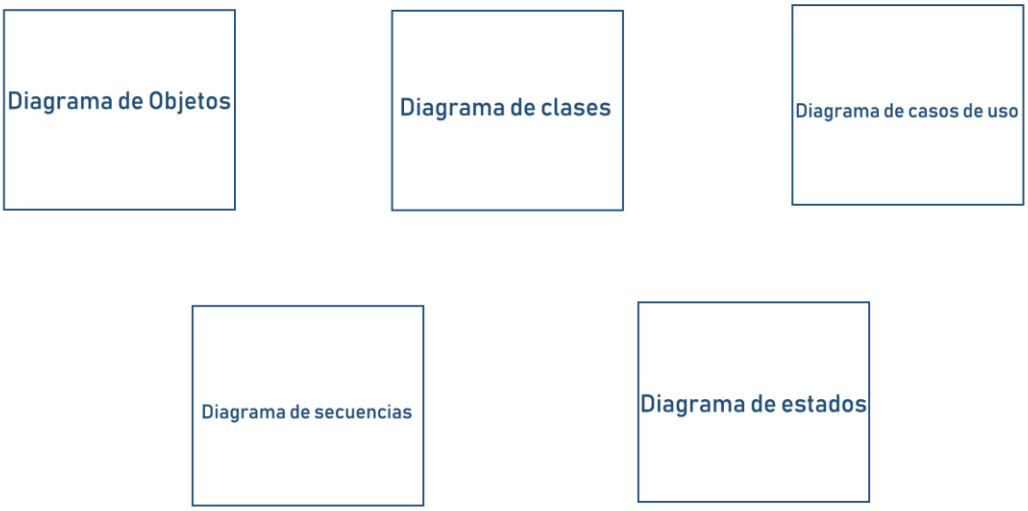

Sistema de inscripciones
Inscripción a un curso
Dar de baja un curso
Imprimir lista de alumnos
Alumno
Alumno
Profesor


Grupo
numeroGrupo
materia
mostrarGrupo()
1
1
*
1
Alumno
nombre
numeroCuenta
mostrarInformacion()
Profesor
nombre
numeroEmpleado
gradoAcademico
mostrarInformacion()


Inscribirse a un grupo
No
Inscripcion
mostrarGrupos
grupoElegido
Confirmar
pedirConfirmacion
Si


obj3:Profesor
nombre=Prof1
numeroEmpleado=
65474-32
gradoAcademico=
Maestría
obj1:Grupo
numeroGrupo=2
materia= POO
obj2:Alumno
nombre=Alumno2
numeroCuenta=
325484-5
obj2:Alumno
nombre=Alumno1
numeroCuenta=
654984-1

El alumno tendrá un panel donde podrá arrastrar dichos diagramas, cada panel tendrá el nombre de un diagrama, dicho nombre deberá coincidir con el diagrama en si para ser calificado mediante el sistema, el panel se muestra a continuación:



Solución y retroalimentación

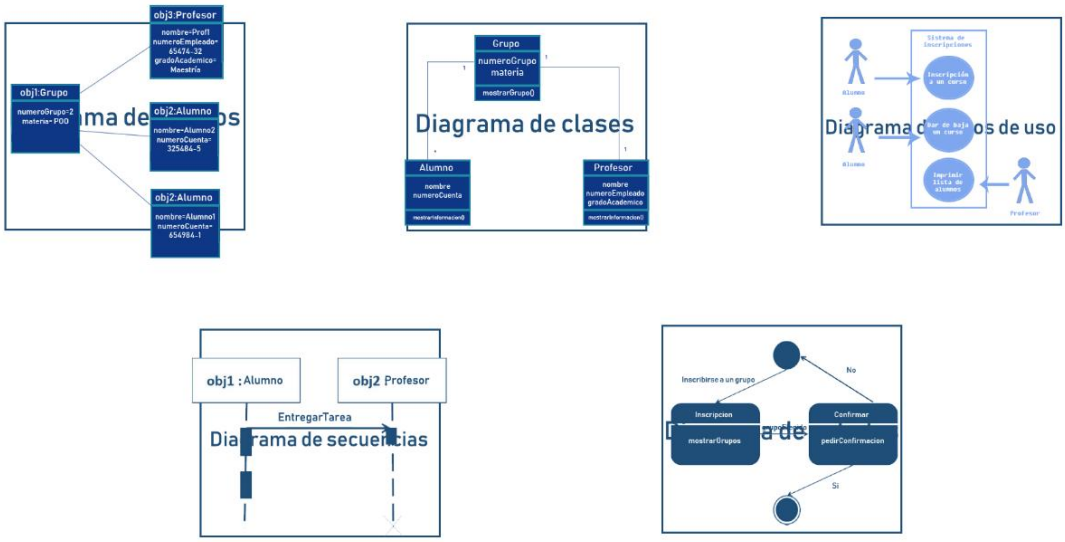


Diagrama de objetos:

Correcto: Los diagramas de objetos modelan las instancias generadas a través de las clases y se utilizan para describir al sistema en un instante de tiempo (o acción) en particular

Incorrecto: Los diagramas de objetos deben de modelar las instancias que se crean en el sistema a través de las clases y se utilizan para describir un instante de tiempo en particular.

Diagrama de clases:

Correcto: Los diagramas de clases permiten modelar las características de las clases que componen al sistema.

Incorrecto: Los diagramas de clases deben de modelar las características de las clases que componen al sistema.

Diagrama de casos de uso:

Correcto: Los diagramas de casos de uso definen la manera en la que el usuario interactúa con el sistema.

Incorrecto: Los diagramas caso de uso deben definir la manera en la que un usuario puede interactuar con el sistema.

Diagrama de secuencia:

Correcto: Los diagramas secuencia muestran una interacción ordenada de eventos.

Incorrecto: Los diagramas de secuencia deben de mostrar una interacción ordenada de eventos visualizando los objetos principales en cada interacción.

Diagrama de estados:

Correcto: Los diagramas de estados permiten describir las transiciones por las cuales puede pasar un objeto durante su tiempo de vida.

Incorrecto: Los diagramas de estados deben describir las transiciones por las cuales puede pasar un objeto durante su tiempo de vida.

Práctica 10

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Qué es un error sintáctico?

a) Errores que suceden cuando la semántica ó el significado no es el que se pretendía.
b) Errores que se generan por infringir las normas de escritura de un lenguaje.
c) Errores que suceden cuando se está ejecutando un programa.

Opción correcta:

b) Errores que se generan por infringir las normas de escritura de un lenguaje. Correcto, los errores sintácticos suceden cuando no se utilizan las reglas del lenguaje de programación de manera correcta, ejemplos de éstos pueden ser falta de cierres de llave (‘}’), falta de punto y coma, palabras reservadas mal escritas, entre otros.

Retroalimentación de las opciones incorrectas:

a) Errores que suceden cuando la semántica ó el significado no es el que se pretendía. Incorrecto: este tipo de errores se presenta cuando la lógica del programa no corresponde con la esperada, sin embargo, los errores sintácticos son “Errores que se generan por infringir las normas de escritura de un lenguaje”.

c) Errores que suceden cuando se está ejecutando un programa: Incorrecto, a estos errores se les llama errores de ejecución, suceden por errores de programación como el acceso a localidades de memoria no permitidas, entre otros. Los errores sintácticos son “Errores que se generan por infringir las normas de escritura de un lenguaje”.

Pregunta 2:

¿Qué es un error semántico?

a) Errores que suceden cuando la lógica ó el significado no es el que se pretendía.
b) Errores que se generan por infringir las normas de escritura de un lenguaje.
c) Errores que suceden cuando se está ejecutando un programa.

Opción correcta:

a) Errores que suceden cuando la lógica ó el significado no es el que se pretendía. Correcto, los errores semánticos se presentan cuando un programa no consigue el resultado esperado.

Retroalimentación de las opciones incorrectas:

a) Errores que se generan por infringir las normas de escritura de un lenguaje: Incorrecto, esta es la definición de error sintáctico, la respuesta correcta es “Errores que suceden cuando la semántica ó el significado no es el que se pretendía”.

c) Errores que suceden cuando se está ejecutando un programa: Incorrecto, a estos errores se les llama errores de ejecución, suceden por errores de programación como el acceso a localidades de memoria no permitidas, entre otros, la respuesta correcta es “Errores que se generan por infringir las normas de escritura de un lenguaje”.

Pregunta 3:

¿Qué es un error de ejecución?

a) Errores que suceden cuando la semántica ó el significado no es el que se pretendía.
b) Errores que se generan por infringir las normas de escritura de un lenguaje.
c) Errores que suceden cuando se está ejecutando un programa.

Opción correcta:

c) Errores que suceden cuando se está ejecutando un programa. Correcto, los errores de ejecución provocan que la aplicación termine abruptamente, esto sucede por algún error no controlado dentro del código.

Retroalimentación de las opciones incorrectas:

a) Errores que suceden cuando la semántica ó el significado no es el que se pretendía: Incorrecto, a estos errores se les llama errores semánticos ó lógicos, la respuesta correcta es “Errores que suceden cuando se está ejecutando un programa”.

b) Errores que se generan por infringir las normas de escritura de un lenguaje: Incorrecto, esta es la definición de error sintáctico, la respuesta correcta es Errores que suceden cuando se está ejecutando un programa”.

Pregunta 4:

¿Para qué sirve un bloque de código “finally”?

a) Para definir un bloque de código donde una excepción puede existir.
b) Para definir un bloque de código donde se controla una excepción.
c) Para definir un bloque de código a ejecutar sin importar si se genera ó no una excepción.

Opción correcta:

c) Para definir un bloque de código a ejecutar sin importar si se genera ó no una excepción. Correcto, el código dentro de un bloque finally se ejecutará al final de del bloque try-catch, ya sea que no se arroje una excepción en el bloque try ó que se controle en el bloque catch, el bloque finally siempre se ejecuta.

Retroalimentación de las opciones incorrectas:

a) Para definir un bloque de código donde una excepción puede existir: Incorrecto, esta definición le corresponde al bloque try, el bloque finally permite definir una acción a realizar sin importar lo sucedido en el bloque try-catch.

b) Para definir un bloque de código donde se controla una excepción: Incorrecto, esta es la definición del bloque catch, el bloque finally permite definir una acción a realizar sin importar lo sucedido en el bloque try-catch.

Actividad que genera calificación:

Objetivo de la evaluación: que el alumno pueda controlar errores mediante el manejo de excepciones con los bloques try-catch-finally.

Identificar los bloques try-catch-finally que correspondan al código

Los bloques try catch pueden ayudar a tener un control de errores eficiente, pero se requiere saber que se puede provocar una excepción en el flujo del programa, además el bloque finally permite realizar una acción al finalizar ambos bloques en caso de ser necesario.

Realizar la actividad en la plataforma, se muestran 4 códigos con diferentes errores sin ser controlados y 4 recuadros con código try-catch que controlan dichos errores, arrastrar el código sin controlar al recuadro que controle dichas excepciones provocadas:

```
public class Excepcion1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("favor de insertar tu edad:");
        int edad= Integer.valueOf(sc.nextLine());
        System.out.println("tu edad es:"+ edad);
    }
}
```

```
public class Excepcion2 {
    public static void main(String[] args) {
        Profesor profesor = null;
        Alumno alumno=null;
        Scanner sc = new Scanner(System.in);
        System.out.println("Insertar nombre:");
        String nombre=sc.nextLine();
        System.out.println("es Alumno o Profesor?");
        String miembro=sc.nextLine();
        switch (miembro) {
        case "Alumno":
            alumno.setNombre(nombre);
            break;
        case "Profesor":
            profesor.setNombre(nombre);
        default:
            break;
        }

    }
}
```

```java
public class Excepcion3 {
    public static void main(String[] args) {
        int number= 50;
        for (int i=10; i>=0;i--) {
            System.out.println("resultado: "+ number/i);
        }
    }
}
```

```java
public class Excepcion4 {
    public static void main(String[] args) {
        for (int i=0; i<=args.length;i++) {
            System.out.println(args[i]);
        }
    }
}
```

Recuadros de excepciones:

```java
public static void main(String[] args) {
    try {
        /*
         * CÓDIGO A CONTROLAR
         * */
    }catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Excepción controlada: "+ e.getMessage());
    }

}
```

```java
public static void main(String[] args) {
    try {
        /*
         * CÓDIGO A CONTROLAR
         * */
    } catch(NullPointerException e ) {
        System.out.println("Excepcion controlada: " + e.getMessage());
    }
}
```

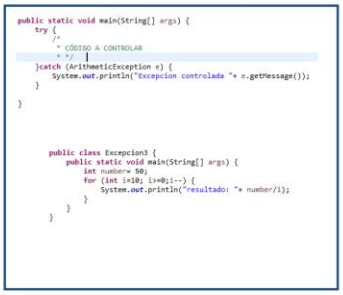
```java
public static void main(String[] args) {
    try {
        /*
         * CÓDIGO A CONTROLAR
         * */
    }catch (ArithmeticException e) {
        System.out.println("Excepcion controlada "+ e.getMessage());
    }

}
```

```java
public static void main(String[] args) {
    try {
        /*
         * CÓDIGO A CONTROLAR
         * */
    }catch (NumberFormatException e) {
        System.out.println("Excepción controlada:"+ e.getMessage());
    }

}
```

Solución y retroalimentación



Recuadro 1:

Correcto: La excepción mostrada es un ArrayIndexOutOfBoundsException esto sucede cuando se desea acceder mediante un índice inexistente a un arreglo lo cual sucede al realizar un ciclo iterativo donde se manejen mal la variable que permite acceder al arreglo.
Incorrecto: La excepción mostrada es un ArrayIndexOutOfBoundsException por lo tanto el código debe intentar acceder a un elemento de un arreglo inexistente.

Recuadro 2:

Correcto: La excepción mostrada es un NullPointerException esto sucede cuando se intenta utilizar un objeto el cual no tiene una referencia, es decir no ha sido instanciado.
Incorrecto: La excepción mostrada es un NullPointerException por lo tanto el código debe intentar acceder a un elemento sin referencia ó que no ha sido instanciado.

Recuadro 3:

Correcto: La excepción mostrada es un NumberFormatException lo cual indica que el dato utilizado requiere tener un formato numerico pero no lo cumple, esto es común cuando se intenta realizar la transformación de una cadena de texto a un número.
Incorrecto: La excepción mostrada es un NumberFormatException por lo tanto el código debe intentar acceder a un elemento de un arreglo inexistente.

Recuadro4:

Correcto: La excepción mostrada es un ArithmeticException esta excepción sirve para controlar situaciones aritméticas no definidas como una división entre cero.
Incorrecto: La excepción mostrada es un ArithmeticException por lo tanto el código debe provocar una operación aritmética no definida como una división entre cero.

Códigos completos para uso interno del área

```
public class Excepcion1 {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            System.out.println("favor de insertar tu edad:");
            int edad= Integer.valueOf(sc.nextLine());
            System.out.println("tu edad es:"+ edad);
        }catch (NumberFormatException e) {
            System.out.println("Excepción controlada:"+ e.getMessage());
        }
    }
}
```

```
public class Excepcion2 {
    public static void main(String[] args) {
        try {
            Profesor profesor = null;
            Alumno alumno=null;
            Scanner sc = new Scanner(System.in);
            System.out.println("Insertar nombre:");
            String nombre=sc.nextLine();
            System.out.println("es Alumno o Profesor?");
            String miembro=sc.nextLine();
            switch (miembro) {
            case "Alumno":
                alumno.setNombre(nombre);
                break;
            case "Profesor":
                profesor.setNombre(nombre);
            default:
                break;
            }
        } catch(NullPointerException e ) {
            System.out.println("Excepcion controlada: " + e.getMessage());
        }
    }
}
```

```java
public class Excepcion3 {
    public static void main(String[] args) {
        try {
            int number= 50;
            for (int i=10; i>=0;i--) {
                System.out.println("resultado: "+ number/i);
            }
        }catch (ArithmeticException e) {
            System.out.println("Excepcion controlada "+ e.getMessage());
        }
    }
}

public class Excepcion4 {
    public static void main(String[] args) {
        try {
            for (int i=0; i<=args.length;i++) {
                System.out.println(args[i]);
            }
        }catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Excepción controlada: "+ e.getMessage());
        }

    }
}
```

Práctica 11

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

La clase FileOutputStream convierte los streams de bytes a streams de caracteres.

a) Verdadero.
b) Falso.

b) Falso. Correcto: la clase FileOutputStream es encargada de crear y escribir un flujo de bytes en un archivo de texto plano, no de convertir flujos de bytes a flujos de caracteres.

a) Verdadero: Incorrecto, la clase encargada de convertir los streams de bytes a streams de caracteres es la clase InputStreamReader, mientras que la clase FileOutputStream es encargada de crear y escribir un flujo de bytes en un archivo de texto plano.

Pregunta 2:

La clase FileInputStream permite crear y escribir un flujo de bytes en un archivo de texto plano.

a) Verdadero.
b) Falso.

b) Falso. Correcto, la clase FileInputStream es encargada de leer un flujo de bytes en un archivo de texto plano.

a) Verdadero: Incorrecto, la clase encargada de crear y escribir un flujo de bytes en un archivo de texto plano es FileOutputStream, la clase FileInputStream es la encargada de leer un flujo desde un archivo de texto.

Pregunta 3:

La clase FileWriter permite escribir un flujo de caracteres en un archivo de texto plano.

a) Verdadero.
b) Falso.

a) Verdadero. Correcto, la clase FileWriter permite escribir un flujo de caracteres en un archivo de texto plano de manera sencilla.

b) Falso: Incorrecto la clase FileWriter sí permite escribir un flujo de caracteres en un archivo de texto plano.

Pregunta 4:

La clase FileReader permite obtener los caracteres desde una fuente externa.

a) Verdadero.
b) Falso.

a) Verdadero. Correcto, la clase FileReader permite leer los caracteres desde una fuente externa, comúnmente un archivo de texto plano.

b) Falso: Incorrecto la clase FileReader tiene como función obtener los caracteres ingresados desde una fuente externa, como lo es un archivo de texto plano.

Pregunta 5:

La clase BufferedReader permite convertir los streams de bytes a streams de caracteres.

a) Verdadero.
b) Falso.

b) Falso. Correcto, la clase BufferedReader crea un buffer para realizar una lectura eficiente de caracteres, la clase que convierte los streams de bytes a streams de caracteres es la clase InputStreamReader.

a) Verdadero: Incorrecto la clase BufferedReader crea un buffer para realizar una lectura eficiente de caracteres, la clase que convierte los streams de bytes a streams de caracteres es la clase InputStreamReader.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno use los conocimientos adquiridos para identificar las diferentes clases relacionadas con el manejo de archivos y pueda usarlas en aplicaciones de consola.

Rellenar los espacios en blanco para generar un programa funcional

El manejo de archivos es una tarea importante en todo lenguaje de programación ya que permite guardar información importante generada en la ejecución del programa en un medio no volátil como lo es un disco duro, ya que la mayoría de las veces se necesita que la información generada persista en alguna base de datos ó archivo de texto para su reutilización en un futuro ó para fines de auditoría.

Se tienen los siguientes códigos que crean un archivo de texto en donde se lee y se escribe información, rellenar los espacios en blanco con una de las opciones:

- File archivo= new File ("grupo"+grupo+".txt");
- fos= new FileOutputStream("grupo"+grupo+".txt");
- PrintWriter salida= new PrintWriter(bw);
- linea=br.readLine();
- br= new BufferedReader(fr);
- FileWriter fw= new FileWriter(archivo);
- FileReader archivo= new FileReader;

Código 1:

```
public class Practica11 {
        public static void main(String[] args) {
        List<Alumno> alumnos= new ArrayList<Alumno>();
        Profesor profesor = new Profesor("Profesor", "APP", "APM", "Profesor-1", "Maestría en ciencias de la computación");
        Scanner sc= new Scanner(System.in);
        for (int i=1; i<11;i++){
            Alumno alumnoNuevo= new Alumno("Alumno"+i, "APP", "APM", "Alumno-"+i, "Ing. en computación");
            alumnos.add(alumnoNuevo);
        }
        try {
        System.out.println("Inserte numero de grupo a crear/editar");
        Integer grupo= Integer.valueOf(sc.nextLine());
        if (!archivo.exists()){
            System.out.println("Archivo no encontrado, se crea uno nuevo");
            archivo.createNewFile();
        }
        BufferedWriter bw= new BufferedWriter(fw);
        salida.println("========Profesor========");
        salida.println(profesor);
        salida.println("========Alumnos========");
        for (Alumno alumno: alumnos){
            salida.println(alumno);
        }
        salida.close();
        }
        catch (IOException e) {
            System.out.println("Excepcion controlada, hubo un error al leer o escribir en un archivo "+e.getMessage());
        }
        catch( NumberFormatException e){
            System.out.println("ExcepciÃ³n controlada, no se puede convertir el valor ingresado a un numero"+ e.getMessage());
        }
    }
}
```

Código 2:

```
public class Practica11B {
        public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        try {
            System.out.println("Inserte numero de grupo a visualizar");
            Integer grupo= Integer.valueOf(sc.nextLine());
                                        ("grupo"+grupo+".txt");
            BufferedReader br= new BufferedReader(archivo);
            System.out.println("Grupo "+ grupo);
            String linea= br.readLine();
            while (linea!=null){
                System.out.println(linea);
                                    
            }
            br.close();
        }
        catch (IOException e) {
            System.out.println("Excepcion controlada, hubo un error al leer o escribir en un archivo "+e.getMessage());
        }
        catch( NumberFormatException e){
            System.out.println("EXCEPCIÃ³N controlada, no se puede convertir el valor ingresado a un numero"+ e.getMessage());
        }
    }
}
```

Solución y retroalimentación

Código 1

1.- Primer recuadro:

- **Respuesta correcta: "File archivo= new File ("grupo"+grupo+".txt");**", correcto, esta línea genera una referencia a un objeto del tipo File con el nombre del archivo personalizado para el grupo ingresado mediante línea de comandos.
- Respuesta incorrecta: **fos=new FileOutputStream("grupo"+grupo+".txt");**", incorrecto esta línea provocaría un error de sintaxis debido a que el objeto fos no existe en este punto del programa, la respuesta correcta es "File archivo= new File ("grupo"+grupo+".txt");**".**
- Respuesta incorrecta: "**PrintWriter salida= new PrintWriter(bw);**", incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto bw no existe hasta este punto del programa, la respuesta correcta es "File archivo= new File ("grupo"+grupo+".txt");**".**
- Respuesta incorrecta: "**linea=br.readLine();**", incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto línea no existe en el contexto actual, la respuesta correcta es "File archivo= new File ("grupo"+grupo+".txt");**".**
- Respuesta incorrecta: "**br= new BufferedReader(fr);**", incorrecto, esto provocaría un error de sintaxis debido a que el objeto br no existe en este contexto, la respuesta correcta es "File archivo= new File ("grupo"+grupo+".txt");**".**
- Respuesta incorrecta: "**FileWriter fw= new FileWriter(archivo);**", incorrecto, esta línea provocaría un error de sintaxis debido a que no existe el objeto archivo en ese momento, la respuesta correcta es "File archivo= new File ("grupo"+grupo+".txt");**".**
- Respuesta incorrecta: "**FileReader archivo= new FileReader**", incorrecto, esta línea provocaría un error de sintaxis en el código subsecuente debido a que el objeto archivo que se usa en el código es una instancia de un objeto File y no de uno

FileReader y este último no contiene el método “exists()”, la respuesta correcta es “File archivo= new File ("grupo"+grupo+".txt");**”.**

2.- Segundo recuadro:

- Respuesta Incorrecta: “**File archivo= new File ("grupo"+grupo+".txt");**”, incorrecto, esta línea provocaría un error de sintaxis debido a que crea un objeto archivo que para este punto ya se utilizó anteriormente, la respuesta correcta es “FileWriter fw= new FileWriter(archivo);”.
- Respuesta incorrecta: **fos=new FileOutputStream("grupo"+grupo+".txt");**”, incorrecto esta línea provocaría un error de sintaxis debido a que el objeto fos no existe en este contexto, la respuesta correcta es “FileWriter fw= new FileWriter(archivo);**”.**
- Respuesta incorrecta: “**PrintWriter salida= new PrintWriter(bw);**”, incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto bw no existe hasta este punto del programa, la respuesta correcta es “FileWriter fw= new FileWriter(archivo);**”.**
- Respuesta incorrecta: “**linea=br.readLine();**”, incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto línea no existe en el contexto actual, la respuesta correcta es “File archivo= new File ("grupo"+grupo+".txt");**”.**
- Respuesta incorrecta: “**br= new BufferedReader(fr);**”, incorrecto, esto provocaría un error de sintaxis debido a que el objeto br no existe en este contexto, la respuesta correcta es “FileWriter fw= new FileWriter(archivo);”**.**
- **Respuesta correcta:** “**FileWriter fw= new FileWriter(archivo);**”, correcto, esta línea crea un objeto FileWriter que ayuda a escribir en un archivo de texto de una manera simple**.**
- Respuesta incorrecta: “**FileReader archivo= new FileReader”** incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto archivo ya existe, la respuesta correcta es “FileWriter fw= new FileWriter(archivo);”**.**

3.- Tercer recuadro:

- Respuesta Incorrecta: “**File archivo= new File ("grupo"+grupo+".txt");**”, incorrecto, esta línea provocaría un error de sintaxis debido a que crea un objeto archivo que ya existía anteriormente por lo tanto lo considera como un nombre repetido, la respuesta correcta es “PrintWriter salida= new PrintWriter(bw);”.
- Respuesta incorrecta: **fos=new FileOutputStream("grupo"+grupo+".txt");**”, incorrecto esta línea provocaría un error de sintaxis debido a que el objeto fos no existe en este contexto, la respuesta correcta es “PrintWriter salida= new PrintWriter(bw);**”.**
- **Respuesta correcta:** “**PrintWriter salida= new PrintWriter(bw);**”, correcto, esta línea crea un objeto que permite imprimir en un objeto BufferedWriter que a su vez

está asociado a un archivo, para escribir en éste la información generada en el programa**.**

- Respuesta incorrecta: "**linea=br.readLine();**", incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto línea no existe en el contexto actual, la respuesta correcta es "PrintWriter salida= new PrintWriter(bw);**".**
- Respuesta incorrecta: "**br= new BufferedReader(fr);**", incorrecto, esto provocaría un error de sintaxis debido a que el objeto br no existe en este contexto, la respuesta correcta es "FileWriter fw= new FileWriter(archivo);"**.**
- Respuesta incorrecta: "**FileWriter fw= new FileWriter(archivo);**", incorrecto, esta línea crea un objeto fw que es usado justamente una linea arriba de este recuadro, lo cual indica que el objeto ya había sido creado anteriormente, provocaría un error de sintaxis por nombrar dos objetos iguales, la respuesta correcta es "PrintWriter salida= new PrintWriter(bw);"**.**
- Respuesta incorrecta: "**FileReader archivo= new FileReader**", incorrecto, esta línea provocaría un error de sintaxis debido a que el objeto archivo ya existe, la respuesta correcta es "PrintWriter salida= new PrintWriter(bw);"**.**

Código 2

4.- Primer recuadro:

- Respuesta Incorrecta: "**File archivo= new File ("grupo"+grupo+".txt");**", incorrecto, esta línea provocaría un error de sintaxis ya que rompe con las reglas del lenguaje, la respuesta correcta es "FileReader archivo= new FileReader".
- Respuesta incorrecta: **fos=new FileOutputStream("grupo"+grupo+".txt");**", incorrecto esta línea provocaría un error de sintaxis debido a que el objeto fos no existe en este contexto, la respuesta correcta es "FileReader archivo= new FileReader**".**
- Respuesta incorrecta: "**PrintWriter salida= new PrintWriter(bw);**", incorrecto, esta linea provocaría un error de sintaxis debido a que rompe con las reglas del lenguaje, la respuesta correcta es "FileReader archivo= new FileReader" que permite terminar la instrucción mostrada**.**
- Respuesta incorrecta: "**linea=br.readLine();**", incorrecto, esta línea provoca un error de sintaxis al romper con las reglas del lenguaje, la respuesta correcta es "FileReader archivo= new FileReader**".**
- Respuesta incorrecta: "**br= new BufferedReader(fr);**", incorrecto, esto provocaría un error de sintaxis debido a que el objeto br no existe en este contexto, la respuesta correcta es "FileWriter fw= new FileWriter(archivo);"**.**
- Respuesta incorrecta: "**FileWriter fw= new FileWriter(archivo);**", incorrecto, esta línea no cumple con las reglas del lenguaje, lo cual provoca un error de sintaxis, la respuesta correcta es "FileReader archivo= new FileReader" que permite completar la instrucción mostrada**.**

- **Respuesta correcta:** “**FileReader archivo= new FileReader”**, correcto, esta línea completa la instrucción creando un objeto llamado archivo que permite, más adelante, crear un objeto BufferedReader con el cual se puede leer la información de un archivo de texto**.**

5.- Quinto recuadro:

- Respuesta Incorrecta: “**File archivo= new File ("grupo"+grupo+".txt");**”, incorrecto, esta línea provoca un error de sintaxis debido a que el objeto archivo se usa antes de este contexto, por lo cual ya existe y se provocaría una variable duplicada, la respuesta correcta es “linea=br.readLine();” que permite leer línea a línea un archivo.
- Respuesta incorrecta: **fos=new FileOutputStream("grupo"+grupo+".txt");**”, incorrecto esta línea provocaría un error de sintaxis debido a que el objeto fos no existe en este contexto, la respuesta correcta es “linea=br.readLine();”**.**
- Respuesta incorrecta: “**PrintWriter salida= new PrintWriter(bw);**”, incorrecto, esta línea provocaría un error de sintaxis debido a que rompe con las reglas del lenguaje, la respuesta correcta es “linea=br.readLine();” que permite leer una línea del archivo**.**
- **Respuesta correcta**: “**linea=br.readLine();**”, correcto, esta línea permite leer línea a línea el archivo de texto que esté relacionado con el objeto BufferedReader br, el método readLine() retornará nulo cuando ya no existan más líneas en el archivo**.**
- Respuesta incorrecta: “**br= new BufferedReader(fr);**”, incorrecto, esto provocaría un error de sintaxis debido a que el objeto estaría duplicado, la respuesta correcta es “linea=br.readLine();”**.**
- Respuesta incorrecta: “**FileWriter fw= new FileWriter(archivo);**”, incorrecto, esta línea provoca un error debido a que el constructor FileWriter no usa un objeto del tipo FileReader, la respuesta correcta es “linea= br.readLine();” que permite leer una línea del archivo usado para crear el objeto br**.**
- Respuesta incorrecta: “**FileReader archivo= new FileReader”**, incorrecto, esta línea generaría un error al duplicar nombre de variables (debido a que ya se declaró antes). La respuesta correcta es “linea= br.readLine();” que permite leer una línea del archivo usado para crear el objeto br**.**

Códigos completos

```
public class Practica11B {
		public static void main(String[] args) {
		Scanner sc= new Scanner(System.in);
		try {
			System.out.println("Inserte numero de grupo a visualizar");
			Integer grupo= Integer.valueOf(sc.nextLine());
			FileReader archivo= new FileReader ("grupo"+grupo+".txt");
			BufferedReader br= new BufferedReader(archivo);
			System.out.println("Grupo "+ grupo);
			String linea= br.readLine();
			while (linea!=null){
				System.out.println(linea);
				linea=br.readLine();
			}
			br.close();
			sc.close();
		}
		catch (IOException e) {
			System.out.println("Excepción controlada, hubo un error al leer ó
			escribir en un archivo "+e.getMessage());
		}
		catch( NumberFormatException e){
			System.out.println("Excepción controlada, no se puede convertir el
			valor ingresado a un numero"+ e.getMessage());
		}
	}
}
```

```
public class Practica11 {
		public static void main(String[] args) {
		List<Alumno> alumnos= new ArrayList<Alumno>();
		Profesor profesor = new Profesor("Profesor", "APP", "APM", "Profesor-1",
		"Maestría en ciencias de la computación");
		Scanner sc= new Scanner(System.in);
		for (int i=1; i<11;i++){
			Alumno alumnoNuevo= new Alumno("Alumno"+i, "APP", "APM", "Alumno-
			"+i, "Ing. en computación");
			alumnos.add(alumnoNuevo);
		}
		try {
		System.out.println("Inserte numero de grupo a crear/editar");
		Integer grupo= Integer.valueOf(sc.nextLine());
		File archivo= new File ("grupo"+grupo+".txt");
		if (!archivo.exists()){
			System.out.println("Archivo no encontrado, se crea uno nuevo");
			archivo.createNewFile();
		}
		FileWriter fw= new FileWriter(archivo);
		BufferedWriter bw= new BufferedWriter(fw);
		PrintWriter salida= new PrintWriter(bw);
```

```
                salida.println("========Profesor========");
                salida.println(profesor);
                salida.println("========Alumnos========");
                for (Alumno alumno: alumnos){
                      salida.println(alumno);
                }
                salida.close();
                }
                catch (IOException e) {
                      System.out.println("Excepcion controlada, hubo un error al leer ó
                      escribir en un archivo "+e.getMessage());
                }
                catch( NumberFormatException e){
                      System.out.println("Excepción controlada, no se puede convertir el
                      valor ingresado a un numero"+ e.getMessage());
                }
          }
}
```

Práctica 12

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Qué es un hilo?

a) Un único flujo de ejecución dentro de un proceso.
b) Un programa en ejecución dentro de su propio espacio de direcciones.
c) Una aplicación que funciona de manera secuencial.

Opción correcta:

a) Un único flujo de ejecución dentro de un proceso. Correcto, un hilo requiere de un proceso que lo controle, siendo el hijo un flujo de ejecución dentro de éste que se ejecuta de manera asíncrona.

Retroalimentación de las opciones incorrectas:

b) Un programa en ejecución dentro de su propio espacio de direcciones: Incorrecto, esta es la definición de un proceso el cual es necesario para el control de un hilo, la respuesta correcta es “Un único flujo de ejecución dentro de un proceso”

c) Una aplicación que funciona de manera secuencial: Incorrecto, la idea de un hilo es que se ejecute de manera asíncrona, es decir que no se necesite seguir una secuencia de instrucciones específica, la respuesta correcta es “Un único flujo de ejecución dentro de un proceso”.

Pregunta 2:

¿Cómo se llama el estado en el cual está un hilo al ser creado?

a) new.
b) runnable.
c) dead.

Opción correcta:

a) new. Correcto, el estado en el cual se encuentra un hilo la primera vez que es creado es el estado new y se mantiene así hasta que el método start es llamado.

Retroalimentación de las opciones incorrectas:

b) runable, Incorrecto, este estado es el que se genera al llamar el método start de un hilo que está en el estado new, que es el estado en el que se encuentra un hilo al ser creado.

c) dead, Incorrecto, un hilo se encuentra en este estado cuando se ejecutó el método stop ó termina su ejecución, la respuesta correcta es el estado “new”.

Pregunta 3:

¿Cómo se llama el estado en el cual está un hilo cuando se invoca el método run?

a) new.
b) runnable.
c) not running.

Opción correcta:

b) runnable. Correcto, el estado en el cual se encuentra un cuando el método run es invocado es el estado runnable.

Retroalimentación de las opciones incorrectas:

a) new, Incorrecto, el estado new es el estado de un hilo recién creado, que cambia hasta que se invoca al método run para tener un estado runnable.

c) not running, Incorrecto, este estado significa que un hilo está detenido por que se invocó el método suspend, sleep ó wait, la respuesta correcta es runnable que se asigna una vez el método run es invocado.

Pregunta 4:

¿Cómo se llama el estado en el cual está un hilo cuando se ejecuta el método sleep?

a) dead.
b) not running.
c) new.

Opción correcta:

b) not running. Correcto, el estado en el cual se encuentra un hilo un cuando se ejecuta el método sleep es el estado not running. A este estado también se puede llegar por los métodos suspend y wait.

Retroalimentación de las opciones incorrectas:

a) dead, Incorrecto, el estado dead lo adquiere un hilo una vez que termina su ejecución ó se ejecuta el método stop, el estado alcanzado por ejecutar el método sleep es el estado not running.

c) new, Incorrecto, el estado new se alcanza cuando se crea un nuevo hilo y no se ha ejecutado ningún otro método, el estado alcanzado al ejecutar el método sleep es el estado not running.

Pregunta 5:

¿Cómo se llama el estado en el cual está un hilo cuando el método run termina su ejecución?

a) dead.
b) not running.
c) stop.

Opción correcta:

a) dead. Correcto, dead es el estado que se le asigna a un hilo cuando termina su ejecución ó se ejecuta el método stop.

Retroalimentación de las opciones incorrectas:

b) not running, Incorrecto, este estado se asigna cuando se llama al método sleep, wait ó suspend, la respuesta correcta es dead que se adquiere al terminar la ejecución del hilo ó al invocar al método stop.

c) stop, Incorrecto, esto es un método no un estado, al ejecutar este método se consigue el estado *dead* que es la respuesta correcta.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno comprenda la utilidad de los hilos y logre implementarlos en un sistema básico que simula cajas de un super mercado.

Rellenar los espacios en blanco para generar un programa funcional

Los hilos (Threads) permiten ejecutar un bloque de código de manera asíncrona a la aplicación principal, mejorando la evidencia de esta para procesos que pueden ejecutarse de una manera no secuencial ya que se pueden lanzar múltiples hilos que se ejecuten de manera concurrente.

Se tiene el siguiente programa que simula una serie de cajas de supermercado que atienden a una fila de clientes, dichas filas se generan de manera automática en el programa, usando hilos llenar los espacios en blanco con las opciones siguientes:

- implements Runnable
- Thread.currentThread().getName()
- atenderCliente(fila, velocidad);
- extends Runnable
- implements Thread
- Thread.sleep

```
public class Practica12{
	public static void main(String[] args) {
		List<Cliente> fila= new ArrayList<Cliente>();
		List<Cliente> fila2= new ArrayList<Cliente>();
		List<Cliente> fila3= new ArrayList<Cliente>();
		List<Cliente> fila4= new ArrayList<Cliente>();
		Cliente cliente=null;
		for (int i=0; i<2;i++) {
			cliente= new Cliente();
			fila.add(cliente);
			cliente= new Cliente();
			fila2.add(cliente);
			cliente= new Cliente();
			fila3.add(cliente);
			cliente= new Cliente();
			fila4.add(cliente);
		}
		new Thread(new Caja(fila,1),"caja rapida").start();
		new Thread(new Caja(fila2,2),"caja 1").start();
		new Thread(new Caja(fila3,2),"caja 2").start();
		new Thread(new Caja(fila4,3),"caja lenta").start();
	}
}
```

Clase: Producto

```
 public class Producto {
	private String id;
	private float precio;
	private String descripcion;
	public Producto() {
	}
	public Producto(String id, float precio, String descripcion) {
		super();
		this.id = id;
		this.precio = precio;
		this.descripcion = descripcion;
	}
```

```
	@Override
	public String toString() {
		return "Producto [id=" + id + ", precio=" + precio + ", descripcion=" +
		descripcion + "]";
	}
	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public float getPrecio() {
		return precio;
	}
	public void setPrecio(float precio) {
		this.precio = precio;
	}
	public String getDescripcion() {
		return descripcion;
	}
	public void setDescripcion(String descripcion) {
		this.descripcion = descripcion;
	}
}
```

Clase: Tienda

```
 public class Tienda {
	private List <Producto> productos= new ArrayList<Producto>();
	public Tienda() {
		productos.add(new  Producto("2541",3.5f,"manzana roja"));
		productos.add(new Producto("1452",5.5f,"mandarina fresca"));
		productos.add(new Producto("1423",120.5f,"Carne de res (1 Kilo)"));
		productos.add(new Producto("851",11.5f,"Refresco sabor uva"));
		productos.add(new Producto("951",10.0f,"Yogurth 355ml"));
	}
	public List<Producto> getProductos() {
		return productos;
	}
	public void setProductos(List<Producto> productos) {
		this.productos = productos;
	}
}
```

Clase: Cliente

```java
public class Cliente {
	private List<Producto> productos= new ArrayList<Producto>();
	public Cliente() {
		seleccionarProductos();
	}
	public List<Producto> getProductos() {
		return productos;
	}
	public void setProductos(List<Producto> productos) {
		this.productos = productos;
	}
	public void seleccionarProductos() {
		Tienda tienda= new Tienda();
		int numeroProductos= (int) (Math.random()*10);
		for (int i=0; i<numeroProductos;i++) {
			int indexProducto= (int)
			(Math.random()*(tienda.getProductos().size()-1));
			productos.add(tienda.getProductos().get(indexProducto));
		}
	}
}
```

Clase: Caja

```java
public class Caja {
    private List<Cliente> fila;
    private long velocidad;

    public Caja(List<Cliente> fila, long velocidad) {
        super();
        this.fila = fila;
        this.velocidad = velocidad;
    }
    public void run() {

    }
    public void atenderCliente(List<Cliente> fila, long velocidad){
        for (Cliente cliente : fila) {
            for(Producto producto: cliente.getProductos()) {
                try {
                    System.out.println("Caja:"+ +" atendiendo a cliente, pasando el producto: "+producto);
                    (velocidad * 1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                finally {
                    System.out.println("Se termina de atender al cliente");
                }
            }
        }
    }
    public List<Cliente> getFila() {
        return fila;
    }
    public void setFila(List<Cliente> fila) {
        this.fila = fila;
    }
    public long getVelocidad() {
        return velocidad;
    }
    public void setVelocidad(long velocidad) {
        this.velocidad = velocidad;
    }
}
```

Solución y retroalimentación

1.- Primer recuadro

- **Respuesta correcta**: "**implements Runnable**", correcto, esta línea permite que la clase implemente la interfaz Runnable lo cual permite crear hilos de esta clase.
- Respuesta incorrecta: "**Thread.currentThread().getName()**", incorrecto, esta línea permite obtener el nombre de identificador del hilo, debe de usarse dentro de algún método, usarlo en este contexto generaría un error de sintaxis, la respuesta correcta es "implements Runnable" que permite utilizar esta clase para crear Threads en un proceso.
- Respuesta incorrecta: "**atenderCliente(fila, velocidad);**", incorrecto, esta línea provocaría un error de sintaxis, la respuesta correcta es "implements Runnable".
- Respuesta incorrecta: "**extends Runnable**", incorrecto, esta línea genera un error de sintaxis debido a que Runnable es una interfaz y no se puede heredar de ella, la respuesta correcta es "implements Runnable".
- Respuesta incorrecta: "**implements Thread**", incorrecto, esta línea provocaría un error debido a que la clase Thread no se implementa, se hereda, la respuesta correcta es "implements Runnable".
- Respuesta incorrecta: "**Thread.sleep**", incorrecto, esta línea provocaría un error de sintaxis al romper las reglas del lenguaje, la respuesta correcta es "implements Runnable".

2.- Segundo recuadro

- Respuesta incorrecta: "**implements Runnable**", incorrecto, esta línea provoca un error de sintaxis debido a que el contexto donde se está intentando usar no es el correcto para la implementación de interfaces, la respuesta correcta es "atenderCliente(fila,velocidad);".
- Respuesta incorrecta: "**Thread.currentThread().getName()**", incorrecto, esta línea es sintácticamente correcta, sin embargo no le da ningún significado al método run, la respuesta correcta es "atenderCliente(fila,velocidad);".
- **Respuesta correcta**: "**atenderCliente(fila, velocidad);**", correcto esta línea hace el llamado al método encargado de atender al cliente recibe una fila y una velocidad en segundos por producto para diferenciar la velocidad en la que se ejecuta cada caja.
- Respuesta incorrecta: "**extends Runnable**", incorrecto, esta línea genera un error de sintaxis debido a que Runnable es una interfaz y no se puede heredar de ella, además de no ser el contexto indicado para heredar de una clase, la respuesta correcta es "atenderCliente(fila, velocidad);".
- Respuesta incorrecta: "**implements Thread**", incorrecto, esta línea provocaría un error debido a que la clase Thread no se implementa ya que no es una interfaz, además de que el contexto para implementar interfaces no es el adecuado, la respuesta correcta es "atenderCliente(fila,velocidad);".

- Respuesta incorrecta: “**Thread.sleep**”, incorrecto, esta línea provocaría un error de sintaxis al romper las reglas del lenguaje, la respuesta correcta es “atenderCliente(fila,velocidad);”.

3.- Tercer recuadro

- Respuesta incorrecta: “**implements Runnable**”, incorrecto, esta línea generaría un error de sintaxis debido a que el contexto para implementar interfaces no es el adecuado, la respuesta correcta es “Thread.currentThread().getName()”.
- **Respuesta correcta**: “**Thread.currentThread().getName()**”, correcto, esta línea permite obtener el nombre del hilo y poderla usar para los fines que se requieran, en este caso se utiliza para imprimir en consola un mensaje el cual indica que hilo es el que está mandando dicho mensaje.
- Respuesta incorrecta: “**atenderCliente(fila, velocidad);**”, incorrecto, esta línea provocaría un error de sintaxis debido a que contiene un punto y coma, además provocaría un ciclo sin fin donde el método se llama a si mismo sin un manejo de recursividad, la respuesta correcta es “Thread.currentThread().getName()”.
- Respuesta incorrecta: “**extends Runnable**”, incorrecto, esta línea genera un error de sintaxis debido a que Runnable es una interfaz y no se puede heredar de ella, la respuesta correcta que funciona en este código es “Thread.currentThread().getName()”.
- Respuesta incorrecta: “**implements Thread**”, incorrecto, esta línea provocaría un error debido a que la clase Thread no se implementa, se hereda, la respuesta correcta es “Thread.currentThread().getName()”.
- Respuesta incorrecta: “**Thread.sleep**”, incorrecto, esta línea provocaría un error de sintaxis al romper las reglas del lenguaje, la respuesta correcta es “Thread.currentThread().getName()”.

4.- Cuarto Recuadro

- Respuesta incorrecta: “**implements Runnable**”, incorrecto, esta línea provoca un error de sintaxis ya que no permite completar la instrucción mostrada, la respuesta correcta es “Thread.sleep”.
- Respuesta incorrecta: “**Thread.currentThread().getName()**”, incorrecto, esta línea no completa la instrucción buscada, la solución correcta es “Thread.sleep” que permite detener la ejecución del hilo por unos instantes.
- Respuesta incorrecta: “**atenderCliente(fila, velocidad);**”, incorrecto, esta línea provocaría un error de sintaxis debido a que no permite terminar la ejecución buscada que es detener la ejecución del hilo por un momento, para lograr esto la respuesta correcta “Thread.sleep” que permite realizar dicha acción.
- Respuesta incorrecta: “**extends Runnable**”, incorrecto, esta línea genera un error de sintaxis debido a que Runnable es una interfaz y no se puede heredar de ella además de no lograr el objetivo que es ejecutar una pausa en la ejecución del hilo, la respuesta correcta es “Thread.sleep”.

- Respuesta incorrecta: “**implements Thread**”, incorrecto, esta línea provocaría un error debido a que la clase Thread no se implementa, se hereda, la respuesta correcta es “Thread.sleep” que permite completar la funcionalidad deseada.
- **Respuesta correcta**: “**Thread.sleep**”, correcto, esta línea permite terminar la instrucción mostrada la cual permite poner en un estado de espera la ejecución del hilo.

Código completo:

```
public class Caja implements Runnable  {
	private List<Cliente> fila;
	private long velocidad;

	public Caja(List<Cliente> fila, long velocidad) {
		super();
		this.fila = fila;
		this.velocidad = velocidad;
	}
	public void run() {
		atenderCliente(fila, velocidad);
	}
	public void atenderCliente(List<Cliente> fila, long velocidad){
		for (Cliente cliente : fila) {
			for(Producto producto: cliente.getProductos()) {
				try {
				System.out.println("Caja:"+Thread.currentThread().getName()
				+" atendiendo a cliente, pasando el producto: "+producto);
				Thread.sleep(velocidad * 1000);
				} catch (InterruptedException e) {
					e.printStackTrace();
				}
				finally {
					System.out.println("Se termina de atender al
					cliente");
				}
			}
		}
	}
	public List<Cliente> getFila() {
		return fila;
	}
	public void setFila(List<Cliente> fila) {
		this.fila = fila;
	}
	public long getVelocidad() {
		return velocidad;
	}
	public void setVelocidad(long velocidad) {
		this.velocidad = velocidad;
	}
}
```

Parte de la ejecución:

```
Caja:caja rapida atendiendo a cliente, pasando el producto: Producto [id=851,
precio=11.5, descripcion=Refresco sabor uva]
Caja:caja lenta atendiendo a cliente, pasando el producto: Producto [id=1423,
precio=120.5, descripcion=Carne de res (1 Kilo)]
Caja:caja 2 atendiendo a cliente, pasando el producto: Producto [id=1423,
precio=120.5, descripcion=Carne de res (1 Kilo)]
Caja:caja 1 atendiendo a cliente, pasando el producto: Producto [id=1423,
precio=120.5, descripcion=Carne de res (1 Kilo)]
Se termina de atender al cliente
Caja:caja rapida atendiendo a cliente, pasando el producto: Producto [id=851,
precio=11.5, descripcion=Refresco sabor uva]
Se termina de atender al cliente
Se termina de atender al cliente
```

Práctica 13

Actividad que no genera calificación: Cuestionario de conocimientos

Pregunta 1:

¿Para qué sirven los patrones de diseño?

a) Para indicar qué código se debe desarrollar en la solución.
b) Para agilizar el desarrollo de una solución.
c) Para proveer un lenguaje de programación que desarrolle una solución.

Opción correcta:

b) Para agilizar el desarrollo de una solución. Correcto, un patrón de diseño provee un paradigma probado para diseñar la solución especifica de un problema y así agilizar el desarrollo de esta.

Retroalimentación de las opciones incorrectas:

a) Para indicar que código se debe desarrollar en la solución: Incorrecto, un patrón de diseño no especifica código a realizar, ya que brinda un paradigma general de cómo implementar una solución, la respuesta correcta es “Para agilizar el desarrollo de una solución”.

c) Para proveer un lenguaje de programación que desarrolle una solución: Incorrecto, un patrón de diseño no especifica el lenguaje de programación en donde se debe de realizar una solución, el patrón de diseño únicamente provee un paradigma para agilizar el desarrollo de ésta (la solución).

Pregunta 2:

Menciona el patrón de diseño que define 3 capas, cada una con un rol especifico

a) Modelo-Vista-Controlador.
b) Creacional.
c) Estructural.

Opción correcta:

a) Modelo-Vista-Controlador. Correcto, el patrón Modelo-Vista-Controlador (MVC) desacopla vistas y modelos que son comunicadas mediante un controlador que avisa de cualquier cambio en el modelo a la vista.

Retroalimentación de las opciones incorrectas:

b) Creacional: Incorrecto, este patrón se basa en la herencia para variar la clase que crea una instancia, no necesariamente tiene tres capas definidas como el patrón Modelo-Vista-Controlador.

c) Estructural: Incorrecto, en un patrón de diseño estructural es importante el cómo están compuestos clases y objetos para formar estructuras grandes sin la necesidad estricta de realizar 3 capas.

Pregunta 3:

¿Cuál es el patrón de diseño que requiere de una instancia única de una clase y controlar el acceso a ésta?

a) Modelo-Vista-Controlador.
b) Singleton.
c) Estructural.

Opción correcta:

b) Singleton. Correcto, el patrón Singleton requiere de una sola instancia de una clase a la cual los clientes accedan de manera controlada para mantener un control estricto de la información.

Retroalimentación de las opciones incorrectas:

a) Modelo-Vista-Controlador: Incorrecto, el patrón MVC requiere de desacoplar vistas y modelos que son comunicados mediante un controlador ante cualquier cambio en el modelo, la respuesta correcta es “Singleton”.

c) Estructural: Incorrecto, en un patrón de diseño estructural es importante el cómo están compuestos clases y objetos para formar estructuras grandes sin necesidad estricta de mantener una única instancia de una clase.

Pregunta 4:

¿Cuál es el patrón de diseño que sugiere usar la herencia?

a) Modelo-Vista-Controlador.
b) Singleton.
c) Estructural.

Opción correcta:

c) Estructural. Correcto, el patrón estructural tiene como característica principal el cómo estén compuestas sus clases, basado en herencia para generar estructuras grandes.

Retroalimentación de las opciones incorrectas:

a) Modelo-Vista-Controlador: Incorrecto, el patrón MVC requiere de desacoplar vistas y modelos que son comunicados mediante un controlador ante cualquier cambio en el modelo, la respuesta correcta es “Estructural”.

b) Singleton: Incorrecto, al patrón singleton sólo le interesa mantener una clase con una instancia única a la cual se tenga un acceso controlado, no la herencia de las clases, lo que si le interesa a los patrones estructurales.

Actividad que genera calificación:

Objetivo de la evaluación: Que el alumno tenga un acercamiento a los patrones de diseño y entienda ejemplos de sistemas sencillos donde se utilicen los mismos.

Cuestionario

En la ingeniería de software se utilizan comúnmente patrones de diseño que permiten tener un panorama de soluciones a problemas cotidianos.

Los patrones de diseño permiten dar un paradigma completo de como diseñar la solución de un problema común, lo cual agiliza el desarrollo de la misma (la solución) ya que se tiene un patrón a seguir que ya ha sido probado anteriormente.

Contestar lo solicitado en las siguientes preguntas:

1.- Se tiene el siguiente sistema para registrar usuarios.

Clase principal

```
public class Patron1 {
	private static UsuarioService servicios= new Registro();
	static Scanner sc= new Scanner(System.in);
	public static void mostrarMenu() {
		System.out.println("Registro de usuarios");
		System.out.println("1.- Registrar usuario");
		System.out.println("2.- Borrar usuario");
		System.out.println("3.- Actualizar contraseña de usuario");
		System.out.println("4.- Ver Usuario");
		System.out.println("5.- Ver todos los usuarios");
		System.out.println("6.- Salir");
	}
	public static void registroUsuario() {
		String username=null;
		String password=null;
		System.out.println("Ingresa tu nombre de usuario:");
		username=sc.nextLine();
		System.out.println("Ingresa tu contraseña:");
		password=sc.nextLine();
		if(servicios.registrarUsuario(username, password)) {
			System.out.println("Usuario Registrado correctamente");

		}
		else {
			System.out.println("Error al registrar este usuario: nombre de
			usuario ya existente");
		}
	}
	public static void borrarUsuario() {
		System.out.println("Ingresa el nombre de usuario a eliminar");
		servicios.borrarUsuario(sc.nextLine());
	}
	public static void actualizarUsuario() {
		String username=null;
		String password=null;
		System.out.println("Ingresa el nombre de usuario a actualizar:");
		username=sc.nextLine();
		System.out.println("Ingresa nueva contraseña:");
		password=sc.nextLine();
		if(servicios.actualizarUsuario(username, password)) {
			System.out.println("Usuario actualizado correctamente");
		}
		else {
			System.out.println("Error al actualizar este usuario: nombre de
			usuario no existente");
		}
	}
```

```
	public static void verUsuario() {
		System.out.println("Ingresa el nombre de usuario a consultar");
		Usuario usuario= (servicios.verUsuario(sc.nextLine()));
		if (usuario!=null) {
			System.out.println(usuario);
		}
		else {
			System.out.println("El usuario no existe");
		}

	}
	public static void verUsuarios() {
		for (Usuario usuario: servicios.verUsuarios()) {
			System.out.println(usuario);
		}
	}
	public static void main(String[] args) {
		int opcion=0;
		while(opcion!=6) {
			mostrarMenu();
			try {
				opcion = Integer.valueOf(sc.nextLine());
				switch (opcion) {
				case 1:
						registroUsuario();
					break;
				case 2:
					borrarUsuario();
					break;
				case 3:
					actualizarUsuario();
					break;
				case 4:
					verUsuario();
					break;
				case 5:
					verUsuarios();
					break;
				case 6:
					System.out.println("Saliendo del sistema");
					break;
				default:
					System.out.println("Opción no valida, favor de
					intentar de nuevo");
					break;
				}
			}catch (NumberFormatException e) {
				System.out.println("Valor ingresado no valido, favor de
				ingresar un numero: "+ e.getMessage());
			}
		}
	}
}
```

Interfaz UsuarioService

```
public interface UsuarioService {
	public boolean registrarUsuario(String username, String password);
	public boolean borrarUsuario(String username);
	public boolean actualizarUsuario(String username, String password);
	public Usuario verUsuario(String username);
	public List<Usuario> verUsuarios();
}
```

Clase Registro

```
public class Registro implements UsuarioService {
	private HashMap<String , Usuario> usuarios = Repositorio.getUsuarios();
	public boolean registrarUsuario(String username, String password) {
		if (usuarios.get(username)==null) {
			usuarios.put(username, new Usuario(username, password, true, new
			Date()));
			return true;
		}
		else {
			return false;
		}
	}
	public boolean borrarUsuario(String username) {
		if (usuarios.get(username)!=null) {
			usuarios.get(username).setActive(false);
			return true;
		}
		else {
			return false;
		}
	}
	public boolean actualizarUsuario(String username, String password) {
		if(usuarios.get(username)!=null) {
			usuarios.put(username, new Usuario(username, password, true, new
			Date()));
			return true;
		}
		else {
			return false;
		}
	}
	public Usuario verUsuario(String username) {
		Usuario usuario= usuarios.get(username);
		if(usuario!=null) {
			Usuario usuarioLimpio = new Usuario(usuario.getUsername(), "****",
			usuario.isActive(), usuario.getCreatedDate());
			return usuarioLimpio;
		}
		else {
			return null;
		}
	}
```

```
	public List<Usuario> verUsuarios() {
		List<Usuario> usuariosRespuesta= new ArrayList<Usuario>();
		Usuario usuarioLimpio= null;
		for(Usuario usuario: usuarios.values()) {
			if(usuario.isActive()) {
				usuarioLimpio= new Usuario(usuario.getUsername(), "*****",
				usuario.isActive(), usuario.getCreatedDate());
				usuariosRespuesta.add(usuarioLimpio);
			}
		}
		return usuariosRespuesta;
	}
}
```

Clase Repositorio

```
 public class Repositorio {
	private static HashMap<String , Usuario> usuarios = new HashMap<>();
	public static HashMap<String, Usuario> getUsuarios() {
		return usuarios;
	}
}
```

Usando lo aprendido hasta el momento, ¿con qué patrón de diseño se realizó dicho sistema? Indicar el por qué.

2.- Se tiene el siguiente sistema para registrar usuarios.

Clase principal

```
 public class Patron2 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		String lugar = "";
		System.out.println("Lugares disponibles");
		System.out.println(SalaCine.getSalaCine().getLugares());
		System.out.println("Ingresa el lugar deseado (o ingrese la palabra
'salir' para terminar la ejecución)");
		do {

			lugar = sc.nextLine();
			new Thread(new Taquilla(lugar), "Taquilla").start();

		} while (!lugar.equals("salir"));
	}
}
```

Clase SalaCine

```
 public class SalaCine {
	private static HashMap<String, Boolean> lugares = new HashMap<String,
Boolean>();
	private static final SalaCine sala = new SalaCine();
	public static SalaCine getSalaCine() {
		return sala;
	}
	private SalaCine() {
		char[] filas = { 'A', 'B', 'C', 'D', 'E', 'F' };
		for (int i = 0; i < filas.length; i++) {
			for (int j = 1; j < 5; j++) {
				lugares.put(filas[i] + String.valueOf(j), true);
			}
		}
	}
	public List<String> getLugares() {
		List<String> lugaresDisponibles = new ArrayList<String>();
		for (String lugar : lugares.keySet()) {
			if (lugares.get(lugar)) {
				lugaresDisponibles.add(lugar);
			}
		}
		return lugaresDisponibles;
	}
	public String ocuparLugar(String lugar) {
		if (lugares.get(lugar) != null) {
			if (!lugares.get(lugar)) {
				return "Lo sentimos, ese lugar acaba de ser ocupado";
			} else {
				lugares.put(lugar, false);
				return "Compra exitosa";
			}
		}
		return "la clave del lugar no existe, favor de verificar";
	}
}
```

Clase Taquilla

```
public class Taquilla implements Runnable {
	private String lugar;
	public Taquilla(String lugar) {
		this.lugar = lugar;
	}
	public void run() {
		try {
			Thread.sleep((1000 * 1));
			if (!lugar.equals("salir")) {

	System.out.println(SalaCine.getSalaCine().ocuparLugar(this.lugar));
					System.out.println("Lugares disponibles");
					System.out.println(SalaCine.getSalaCine().getLugares());
					System.out.println("Ingresa el lugar deseado (o ingrese la
					palabra 'salir' para terminar la ejecución)");
			}
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}
}
```

Usando lo aprendido hasta el momento, ¿con qué patrón de diseño se realizó dicho sistema? Indicar el por qué.

3.- De los patrones mencionados en la práctica, ¿cuál sería el más indicado para realizar una aplicación de chat y por qué?

Rúbrica de evaluación

Rúbrica a evaluar	Forma de evaluar	Respuesta esperada	Puntos
Pregunta 1	Revisión del documento entregado por el alumno.	"Modelo Vista Controlador, se tienen las 3 capas de este modelo identificando la capa de vista como la clase principal que interacciona con el usuario, el modelo que es el Usuario junto con su repositorio y el controlador que es la clase Registro" ó similares	2 puntos (1 punto por la respuesta correcta del modelo-vista-controlador y un punto por describir las capas de éste en el sistema)
Pregunta 2	Revisión del documento entregado por el alumno.	"Singleton, se tiene la clase única 'SalaCine' con su constructor privado y con control de acceso a esta instancia, además de tener un cliente que accede a la misma para obtener y modificar su información" ó similares	2 puntos (1 punto por la respuesta correcta: Singleton y un punto por describir dicho patrón en el sistema)
Pregunta 3	Revisión del documento entregado por el alumno	"El patrón por comportamiento debido a que establece la comunicación entre dos objetos, en este caso emisor y receptor" ó una justificación similar.	1 punto

Puntos totales: 5 Calificación: (puntos/5) *10

4.3 Manual de instalación de plataforma

Para la realización de este proyecto se utilizó un servidor basado en Red Hat Linux, específicamente Fedora 29, por lo cual este manual estará enfocado a dicha distribución

La plataforma está realizada en Django, un framework del lenguaje de programación Python para el desarrollo de aplicaciones web. Fedora ya tiene una versión de Python 3 preinstala, se usa dicha instalación para la configuración del entorno virtual necesario para la instalación, un entorno virtual es una instalación nueva de Python que no afecta a la instalación principal del sistema operativo, lo que es útil para mantener un mejor control de versiones de los paquetes instalados.

Se recomienda crear una carpeta que será usada únicamente para los siguientes procesos de instalación, esto no es necesario, pero permite mantener todos los archivos en un solo directorio de manera que, en caso de ser necesario, puedan ser ubicados de una manera sencilla.

4.3.1 Instalación de dependencias

Para que se pueda instalar de manera correcta Python y el servidor de aplicaciones se requiere tener algunas dependencias instaladas, las cuales se listan a continuación:

- gcc
- openssl-devel
- bzip2-devel
- python3-devel
- libffi-devel

Las dependencias anteriores se deben instalar en el siguiente orden:

```
sudo yum install gcc openssl-devel bzip2-devel

sudo dnf install python3-devel

sudo yum install libffi-devel
```

4.3.2 Instalación del entorno virtual

Para la creación del entorno virtual se usa virtualenv, un paquete disponible mediante el manejador de paquetes de Python3, pip3, por lo cual para su instalación se debe ejecutar el siguiente comando con permisos elevados:

```
pip3 install virtualenv
```

Una vez instalado virtualenv se procede a crear el entorno virtual utilizando el siguiente comando:

```
virtualenv ~/poo
```

Este comando crea el entorno virtual en el directorio base del usuario que ejecute el comando, en este caso el entorno virtual queda en la ruta absoluta /home/devuser/poo, esta ruta es importante ya que se utilizará en futuras configuraciones.

Para activar el entorno virtual se ejecuta el siguiente comando:

```
source ~/poo/bin/activate
```

Una vez activado el entorno virtual, la consola debe de indicar el uso de un entorno virtual mediante el nombre de éste entre paréntesis:

```
(poo) [devuser@localhost ~]$
```

Ahora se procederá a actualizar ‘pip’, que es la herramienta que permite instalar los paquetes requeridos, mediante el siguiente comando:

```
pip install --upgrade pip
```

Finalmente, el sistema desarrollado requiere la instalación de los siguientes paquetes

Paquete	*Versión*
Pillow	5.3.0
django	2.1.3
django-ckeditor	5.6.1

Para la instalación de dichos paquetes se utiliza la herramienta pip, por lo cual se debe generar un archivo de texto plano con el nombre “requirements.txt” como se muestra en la imagen.

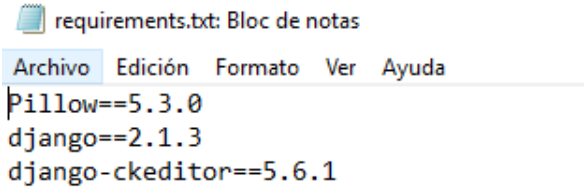


Imagen 4.1: Contenido del archivo requeriments.txt

Una vez se tenga el archivo mostrado, se procede a instalarlo dentro del entorno virtual, esto se logra posicionándose en el directorio que contenga al archivo requirements.txt y ejecutando el siguiente comando:

```
pip install -r requirements.txt
```

Se comenzará la instalación de los paquetes especificados, al terminar se puede verificar las instalaciones de éstos como se observa en la siguiente imagen:

```
(poo) [devuser@localhost ~]$ pip list
Package         Version
--------------- ----------
certifi         2018.11.29
Django          2.1.3
django-ckeditor 5.6.1
django-js-asset 1.2.1
Pillow          5.3.0
pip             19.0.1
pytz            2018.9
setuptools      40.8.0
wheel           0.32.3
```

Imagen 4.2: Lista de paquetes instalados en el entorno virtual.

Una dependencia extra que se debe instalar es psycopg2-binary, que permite la conexión con una base de datos PostgreSQL, para ello se puede utilizar pip de la siguiente manera:

```
pip install psycopg2-binary
```

4.3.3 Instalación de base de datos PostgreSQL

Las siguientes instrucciones describen la instalación de PostgreSQL en una distribución con un manejador YUM (Red hat,Centos, Fedora, etcétera), la administración de los usuarios y contraseñas de la base de datos quedan fuera de alcance de este documento, la creación de las tablas se realizará mediante el ORM de Django.

La versión utilizada es PostgreSQL 10 por lo cual se debe descargar el repositorio de ésta desde la página oficial (https://yum.postgresql.org/) ya sea mediante la interfaz gráfica del sistema operativo ó mediante línea de comandos como se muestra a continuación:

curl -0 https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-7-x86_64/pgdg-centos10-10-2.noarch.rpm

Una vez descargado el repositorio se procede a instalarlo con permisos elevados:

```
 sudo rpm -ivh pgdg-centos10-10-2.noarch.rpm
[sudo] password for user
```

La instalación procederá de manera automática y, al terminar, se podrán instalar mediante YUM los paquetes que se necesiten, para listar los paquetes requeridos se puede utilizar el siguiente comando:

*yum list postgres**

Finalmente, se procede a instalar los 3 paquetes siguientes para poder hacer uso de la base de datos:

yum install postgresql10-server.x86_64 postgresql10-contrib.x86_64 postgresql10-devel.x86_64

Con esto se termina la instalación de la base de datos. El usuario administrador y el esquema que se usará en el entorno productivo deberán ser creados bajo los criterios de la coordinación de la asignatura, a partir de este momento en este documento se utilizarán credenciales del entorno de desarrollo y una dirección localhost para conectarse al servidor de base de datos.

4.3.4 Instalación de la plataforma Django

El desarrollo se ha mantenido con un control de versiones en la plataforma bitbucket a la cual se puede acceder mediante la siguiente liga:

https://bitbucket.org/lizethparrales/django/src/master/

para levantar esta plataforma se deberá usar una cuenta de bitbucket con acceso autorizado al repositorio y clonarlo en un directorio dentro del servidor mediante el siguiente comando

cd dj

Se debe colocar el nombre del usuario autorizado que esté clonando el repositorio, se pedirá una autenticación mediante contraseña antes de empezar la clonación, hay que brindar dicha contraseña y se creará una carpeta llamada “django” en el directorio donde se haya pedido la clonación, este proyecto es la raíz del repositorio, el proyecto se encuentra dentro de esta carpeta con el nombre ‘poo’.

4.3.5 Despliegue de la plataforma Django

Para el despliegue de la plataforma se debe situar dentro del proyecto, es decir, dentro de la carpeta ‘poo’ y, antes de ejecutar el comando runserver, se debe editar la configuración del archivo ‘settings.py’ del proyecto para poder conectarse con el servidor de base de datos. El archivo se encuentra en encuentra en la ruta: {raíz_repositorio}/poo/poo/settings.py.

El proyecto viene con una base de datos sqlite precargada y, por defecto, está configurado para trabajar con ella, para conectar la base de datos instalada se debe modificar ó comentar las siguientes líneas:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Imagen 4.3: Configuración por defecto con conexión a sqlite

Para realizar la configuración para la conexión con PostgreSQL (las credenciales y el nombre de la base de datos son ilustrativas) se deben agregar las líneas del usuario, password, host y puerto, como se muestra a continuación:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'djangoDb',
        'USER': 'postgres',
        'PASSWORD': 'DevUser123.',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Imagen 4.4: Configuración necesaria para conexión a postgresql

Se requiere ahora migrar los modelos de la plataforma a la base de datos postgresql, para lo cual se utilizará el siguiente comando:

python manage.py migrate

Lo anterior genera las tablas de base de datos en el servidor PostgreSQL, las cuales se pueden visualizar desde línea de comandos ó usando un entorno visual para el manejo del servidor. La siguiente imagen es del manejador pgAdmin4.

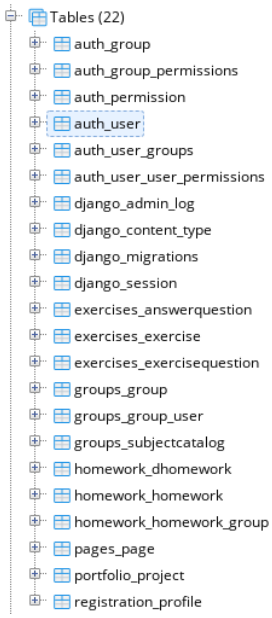


Imagen 4.5: Tablas creadas en la base de datos después del comando migrate.

Para poder administrar los contenidos de la plataforma se requiere la creación de un súper usuario, para ello se utiliza el comando createsuperuser de la siguiente manera:

```
(py37) [devuser@localhost poo]$ python manage.py createsuperuser
Nombre de usuario: admin
Dirección de correo electrónico: admin@django.com
Password:
Password (again):
La contraseña es demasiado similar a la de dirección de correo electrónico.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Igual que las credenciales de la base de datos, la creación del súper usuario debe ser realizada mediante los estándares de la coordinación de la asignatura, lo mostrado en este documento son credenciales del entorno de desarrollo.

Al final del proceso anterior, se creará un usuario administrador en nuestra base de datos PostgreSQL que permitirá el manejo de contenido dentro de la aplicación.

Finalmente se ejecuta el servidor Django mediante el archivo manage,py y se debe mostrar la siguiente salida:

```
(poo2) [devuser@localhost poo]$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
February 28, 2019 - 00:19:42
Django version 2.1.3, using settings 'poo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Una vez que se ha levantado el servicio, se puede acceder a la dirección local 127.0.0.1:8000 para revisar la aplicación:

Imagen 4.6: Imagen principal de la plataforma.

4.3.6 Despliegue en producción

El servidor de aplicaciones Django configurado hasta este momento se ejecuta en modo 'DEBUG' y permite ejecutar archivos estáticos ó multimedia, lo cual no se recomienda por temas de seguridad. Para arreglar esta situación se requiere un servidor apache con sus herramientas de desarrollo. Para instalar un servidor apache se debe ejecutar la siguiente instrucción como usuario root:

```
yum install httpd httpd-devel
```

A partir de aquí se usará el entorno virtual creado anteriormente mediante el comando

```
source ~/poo/bin/activate
```

Además, se requiere instalar mod_wsgi que permite comunicar el servidor apache con el servidor Django. Para ello, se descarga la última versión (4.6.5 para este proyecto) del enlace https://github.com/GrahamDumpleton/mod_wsgi/archive/4.6.5.tar.gz como se muestra a continuación:

```
wget https://github.com/GrahamDumpleton/mod_wsgi/archive/4.6.5.tar.gz
```

Una vez descargado, se procede a extraer los archivos del fichero tar.gz mediante el siguiente comando

```
tar xvfz 4.6.5.tar.gz
```

Esto creará una carpeta llamada mod_wsgi-4.6.5 a la cual se accede mediante el comando cd, para poder compilar el código fuente como se muestra en la imagen:

```
(poo) [devuser@localhost Install]$ cd mod_wsgi-4.6.5/
(poo) [devuser@localhost mod_wsgi-4.6.5]$ ./configure
checking for apxs2... no
checking for apxs... /usr/bin/apxs
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for prctl... yes
checking Apache version... 2.4.6
checking for python... /home/devuser/poo/bin/python
configure: creating ./config.status
config.status: creating Makefile
```

Imagen 4.7: Resultado de la compilación de mod_wsgi.

La necesidad de ejecutar el comando ‘./configure’ utilizando el entorno virtual es debido a que mod_wsgi requiere utilizar una instalación de Python para compilar y ésta debe ser la misma instalación utilizada para ejecutar la plataforma.

Se procede a realizar la instalación de mod_wsgi mediante el siguiente comando

sudo make install

Al acabar la instalación se debe revisar que mod_wsgi tenga la configuración adecuada como muestra la siguiente imagen:

```
(poo) [devuser@localhost mod_wsgi-4.6.5]$ ldd /etc/httpd/modules/mod_wsgi.so
        linux-vdso.so.1 =>  (0x00007ffd51573000)
        libpython3.7m.so.1.0 => /lib64/libpython3.7m.so.1.0 (0x00007f2850b79000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f285095d000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f2850759000)
        libutil.so.1 => /lib64/libutil.so.1 (0x00007f2850556000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f2850254000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f284fe87000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f28512eb000)
```

Imagen 4.8: Configuración de mod_wsgi.

Ahora se debe revisar que el servicio de apache funcione correctamente, para ello se debe inicar el servidor:

systemctl start httpd

Con el servidor ejecutándose, se accede a localhost desde un navegador web y se debe observar una pantalla como la siguiente:

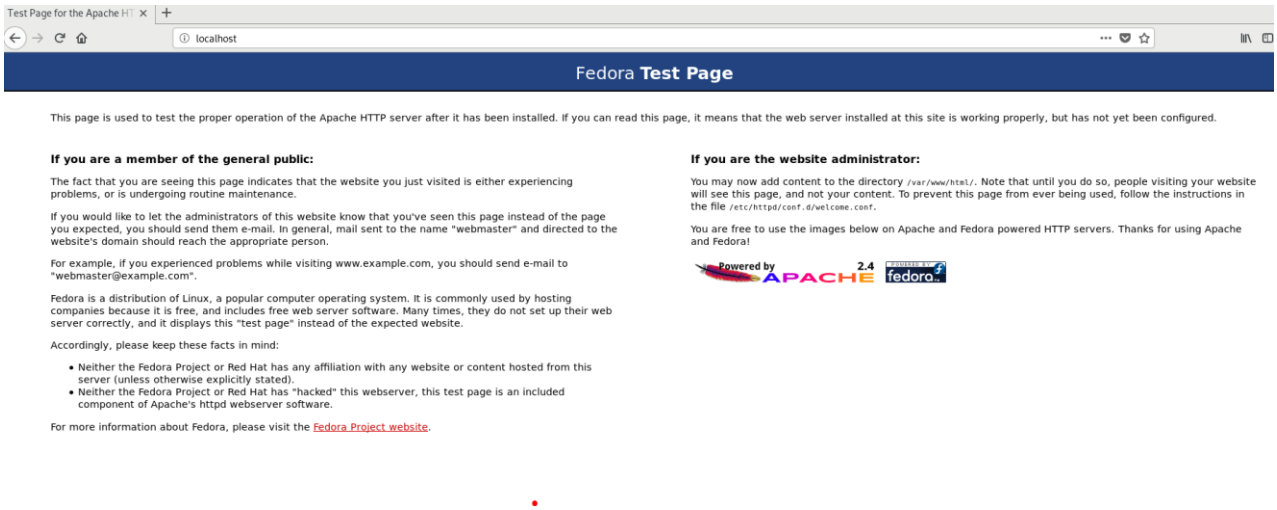


Imagen 4.9: Pantalla principal del servidor apache.

Por seguridad se debe usar un certificado ssl para el cifrado de datos enviados y recibidos por el servidor, para este paso se requiere un certificado y su llave los cuales serán provistos por la coordinación, para fines de este manual se usará un certificado auto firmado que se puede generar mediante las siguientes instrucciones:

Se debe instalar openssl para la creación de certificados y el modulo de apache mod_ssl para el uso de éste, por lo cual se ejecuta el siguiente comando:

```
sudo yum install mod_ssl openssl
```

una vez instalados dichos paquetes, se procede a cambiar al directorio donde se almacenan los certificados del equipo (/etc/pki/tls/certs) y se ejecuta el siguiente comando:

```
sudo openssl req -x509 -nodes -newkey rsa:2048 -keyout localhost.key -out localhost.crt
```

la palabra localhost puede ser cambiada por el nombre de host del servidor, al terminar se habrán creado los archivos del certificado y su llave en el directorio como se puede ver en la imagen:

```
[devuser@localhost certs]$ pwd
/etc/pki/tls/certs
[devuser@localhost certs]$ ls
ca-bundle.crt  ca-bundle.trust.crt  localhost.crt  localhost.key
```

Imagen:4.10 archivos del certificado creados.

Al instalar mod_ssl también se creó un archivo de configuración que se puede encontrar en /etc/httpd/conf.d bajo el nombre de ssl.conf, en dicho archivo se tienen dos propiedades con las rutas de los certificados, dichas rutas deberán ser verificadas y modificadas para apuntar a nuestros archivos localhost.crt y localhost.key de la siguiente manera:

```
SSLCertificateFile /etc/pki/tls/certs/localhost.crt

SSLCertificateKeyFile /etc/pki/tls/certs/localhost.key
```

Una vez realizado este cambio se procede a reiniciar el servidor apache (systemctl restart httpd) y se deberá poder acceder al sitio por defecto del servidor apache mediante el protocolo https, nos mostará una advertencia sobre la validez del certificado ya que es un certificado autofirmado, al agregar la excepción se nos debe de mostrar la pagina por defecto como se observa en la imagen

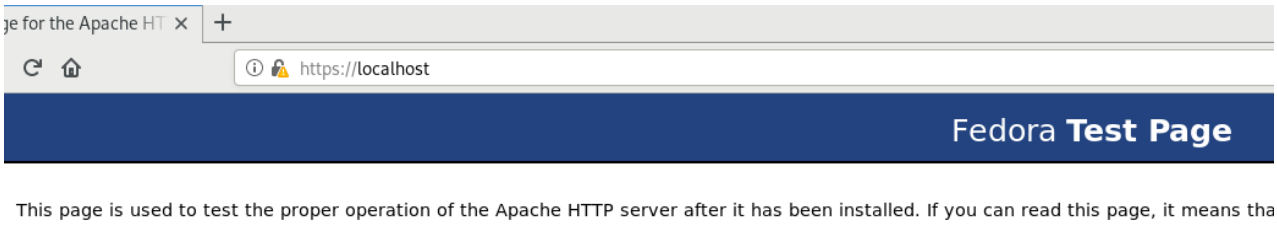


Imagen 4.11 Acceso al servidor apache mediante https

El paso siguiente es cargar el módulo wsgi en el servidor apache, para lo cual se debe crear un archivo de configuración con el nombre wsgi.conf en el directorio /etc/httpd/conf.d/:

```
sudo vi /etc/httpd/conf/httpd.conf
```

Dentro del archivo de configuración, se debe de configurar el modulo wsgi de la siguiente manera (se debe cambiar las rutas según sea el caso)

```
LoadModule wsgi_module modules/mod_wsgi.so
WSGIScriptAlias / /home/devuser/django/poo/poo/wsgi.py
WSGIDaemonProcess poo python-path=/home/devuser/poo:/home/devuser/poo/lib/python3.7/site-
packages
WSGIProcessGroup poo
Alias /media/ /home/devuser/django/poo/media/
<Directory /home/devuser/django/poo/media>
   Require all granted
</Directory>
Alias /static/ /home/devuser/django/poo/static/
<Directory /home/devuser/django/poo/static>
   Require all granted
</Directory>
<Directory /home/devuser/django/poo/poo/>
   <Files wsgi.py>
      Require all granted
   </Files>
</Directory>
```

El código anterior carga el módulo wsgi al servidor apache y configura los parámetros necesarios para crear un servicio Daemon llamado poo al cual se le especifica el path del entorno virtual que se está utilizando y el directorio donde se encuentra la aplicación, los alias de media y static se configuran para que el servidor apache pueda acceder a estos directorios y servir de manera correcta dichos archivos, además que la carpeta de la aplicación debe de tener permisos de ejecución y lectura, además de que se deben de crear excepciones en el cortafuegos y permisos de SElinux para el correcto

funcionamiento de la aplicación, por lo cual se requiere ejecutar los siguientes comandos (modificar las rutas según sea necesario)

```
chmod -R 755 /home/devuser/django

firewall-cmd --permanent --add-service=http

firewall-cmd --permanent --add-service=https

systemctl restart firewalld

setsebool -P httpd_enable_homedirs true

chcon -R -t httpd_sys_content_t /home/devuser/django/poo

setsebool -P   httpd_can_network_connect_db=1
```

La aplicación aun requiere unos ajustes para poder ser desplegada mediante apache. Se debe crear un directorio static en la raíz del proyecto como se muestra en la siguiente imagen:

```
(poo) [devuser@localhost poo]$ pwd
/home/devuser/Poo/django/poo
(poo) [devuser@localhost poo]$ ls
a.out  core  db.sqlite3  exercises  groups  homework  manage.py  media  pages  poo  practices  profiles  registration  static
```

Imagen 4.12: Estructura de carpetas dentro del proyecto

Una vez creado el directorio static se deben realizar algunas modificaciones en el archivo settings ubicado en el archivo: /django/poo/poo/settings.py. Dentro del archivo, se debe desactivar la opción de menú y se debe añadir la IP del servidor ó el dominio de la plataforma a la lista “Allowed Host” (en este caso localhost, en productivo se deberá de configurar acorde a la ip ó al dominio) como muestra la imagen:

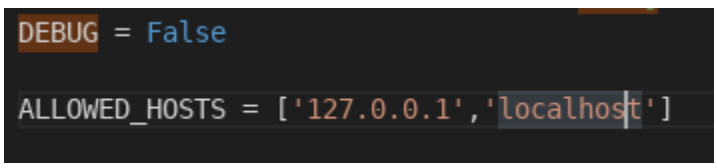


Imagen 4.13: Configuración de dominios permitidos.

Dentro del archivo settings.py, también se debe crear una referencia a la carpeta static utilizando la directiva STATIC_ROOT debajo de la directiva STATIC_URL como se muestra a continuación:

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, "static")
```

Imagen 4.14: Configración del directorio static.

Finalmente se deberá de recolectar todos los ficheros estáticos en esta carpeta, para ello se debe utilizar el archivo manage.py de la siguiente manera:

python manage.py collectstatic

Ademas se debe de realizar una ultima configuración en el archivo wsgi.py que se encuentra en la carpeta principal de la aplicación (poo), esta configuración consite en añadir a la ruta del sistema los archivos de la aplicación y se debe de activar el entorno virtual, se deberán modificar las rutas a conveniencia.

```
import os,sys
from django.core.wsgi import get_wsgi_application
sys.path.append('/home/devuser/django/poo/')
os.environ.setdefault("LANG","es_MX.UTF-8")
os.environ.setdefault("LC_ALL","es_MX.UTF-8")
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'poo.settings')
activate_this = '/home/devuser/poo/bin/activate_this.py'
exec(open(activate_this).read())
application = get_wsgi_application()
```

Imagen 4.15: Configuración wsgi.py

Una vez que se haya realizado toda esta configuración, se deberá iniciar el servicio httpd e ingresar a la dirección localhost ó a la configurada en producción (IP ó dominio) como se muestra en las siguientes imágenes:

```
(poo) [devuser@localhost mod_wsgi-4.6.5]$ systemctl start httpd
```

Imagen 4.16: Inicio del servicio http

Imagen 4.17: Acceso a la plataforma usando apache

5.- Manual de administración

Para acceder a la consola de administración de Django, se debe acceder desde un navegador web a la dirección localhost/admin. En esa URL se mostrará un formulario de autenticación en el cual se ingresan las credenciales del usuario. Dependiendo del tipo de usuario las opciones están limitadas, sólo el usuario administrador tiene los permisos necesarios para realizar cualquier acción.

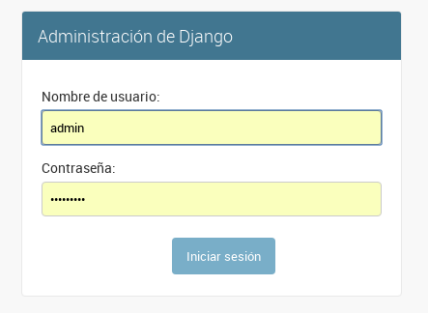


Imagen 5.1: formulario de autenticación

Al autenticarse de manera correcta aparece la consola de administración del servidor.

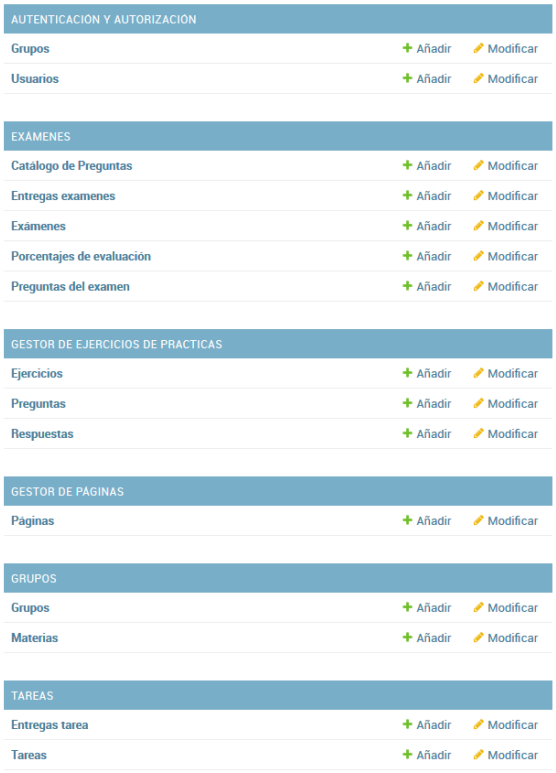


Imagen 5.2: Vista de la consola de administración para el súper usuario

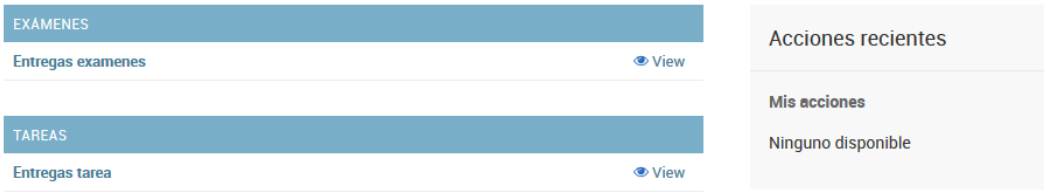


Imagen 5.3: Vista de la consola de administración para alumno

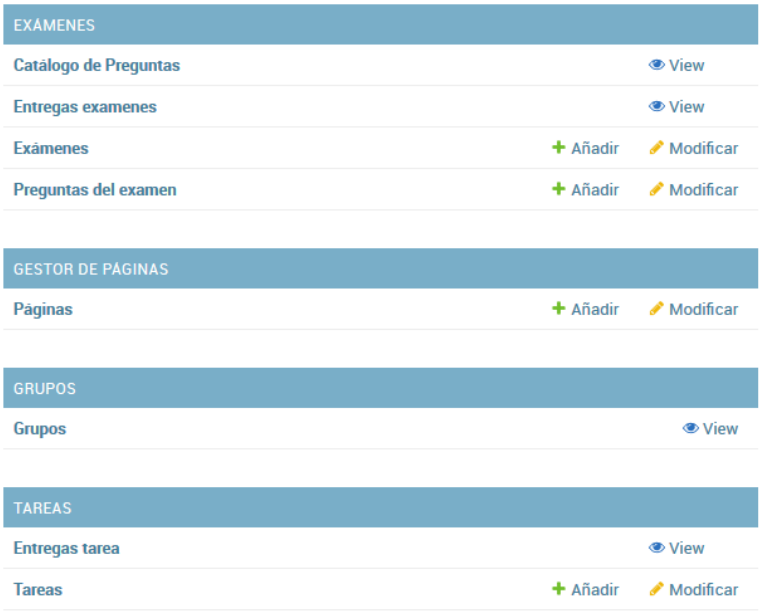


Imagen 5.4: Vista de la consola de administración para profesor

5.1 Autenticación y autorización

5.1.1 Grupos

Esta sección permite crear y administrar los grupos de usuarios y los permisos de éstos.

El sistema maneja dos grupos, Alumno y Profesor.

Imagen 5.5: Grupos de usuarios del sistema.

Si se desea tener un tercer tipo de grupo se puede añadir un nuevo grupo mediante el botón de la interfaz 

el cual muestra la siguiente interfaz

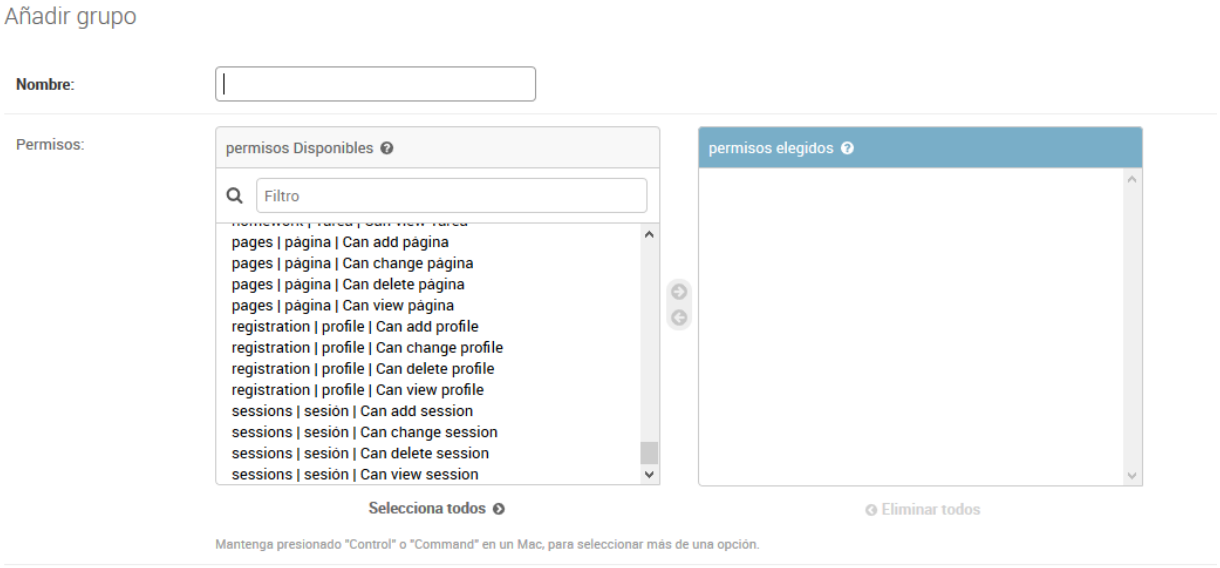


Imagen 5.6: Administración de grupos

En el campo Nombre se coloca el identificador del grupo, además, se pueden asignar los permisos que requiera este nuevo grupo. Cada módulo del sistema tiene 4 permisos: add, change, delete y view, estos permisos afectan la manera en la que la consola de administración interactúa con el usuario autenticado. Cada permiso describe la acción permitida, por ejemplo, un permiso “can View Tarea” implica que el usuario puede únicamente visualizar los elementos asociados al modelo ‘Tarea’, estos permisos se utilizan tanto en las vistas de la plataforma como en la consola de administración sin ser dependientes una de la otra, por tanto, un grupo puede tener el permiso de crear exámenes. Sin embargo, también existe una vista en la plataforma dedicada para este propósito, la cual sólo está disponible si el usuario autenticado tiene el permiso “Can add Examen” lo cual limita la funcionalidad de los usuarios para manipular información sensible (ejemplos y detalles de esta situación se encuentran en el manual técnico).

Los permisos de los dos grupos iniciales de la plataforma se muestran en las siguientes tablas:

Modelo	*Permiso*
DExam	Can view Entrega Examen
Groups	Can view Porcentaje de evaluación
DExamUserAnswers	Can view D exam user answers
DHomeWork	Can View Entrega Tarea

Tabla 5.1: Grupo Alumnos

Modelo	Permisos
Group	Can view Grupo
Pages	Can add, change, view Página
DHomework	Can view DHomework
HomeWork	Can add, change, delete, view Entrega
AnswerQuestion	Can add, view AnswerQuestion
DExam	Can view DExam
Exam	Can add, change, view Exam
ExamQuestion	Can add, change, view, delete ExamQuestion
CatalogQuestion	Can view CatalogQuestion
NotePercentage	Can add, change, view, delete NotePercentage

Tabla 5.2: Grupo Profesor

5.1.2 Usuarios

En esta sección se puede manipular la información de los usuarios, esto sólo lo puede hacer el súper usuario del sistema, la plataforma muestra una tabla con los usuarios registrados en la plataforma.

NOMBRE DE USUARIO	DIRECCIÓN DE CORREO ELECTRÓNICO	NOMBRE	APELLIDOS	ES STAFF
308021-85	308021-85@cambiar.com	Tania	App9App9	✓
30804888-1	juan.delcamino.r@gmail.com	Juan	App1App1	✓
40804888-1	40804888-1@cambiar.com	Sandra	App1App1	✓
40804888-2	40804888-2@cambiar.com	Pedro	App2App2	✓
40804888-3	40804888-3@cambiar.com	Sara	App3App3	✓
40804888-4	40804888-4@cambiar.com	Sarahi	App4App4	✓
40804888-5	40804888-5@cambiar.com	Laura	App5App5	✓
40804888-6	40804888-6@cambiar.com	Luis	App6App6	✓
40804888-7	40804888-7@cambiar.com	Saul	App7App7	✓
40804888-8	40804888-8@cambiar.com	Hector	App8App8	✓

Imagen 5.7: Lista de usuarios.

Si se selecciona el perfil de un alumno se puede modificar su información personal, agregarlo a un grupo de usuarios ó añadir permisos específicos como se muestra en las siguientes imágenes.

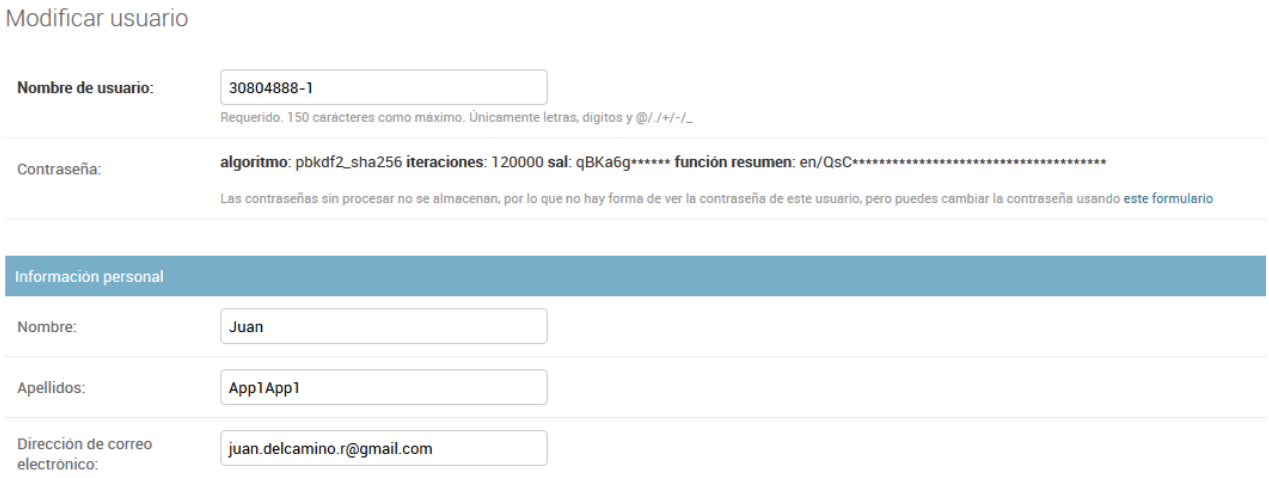


Imagen 5.8: Edición de información básica del perfil

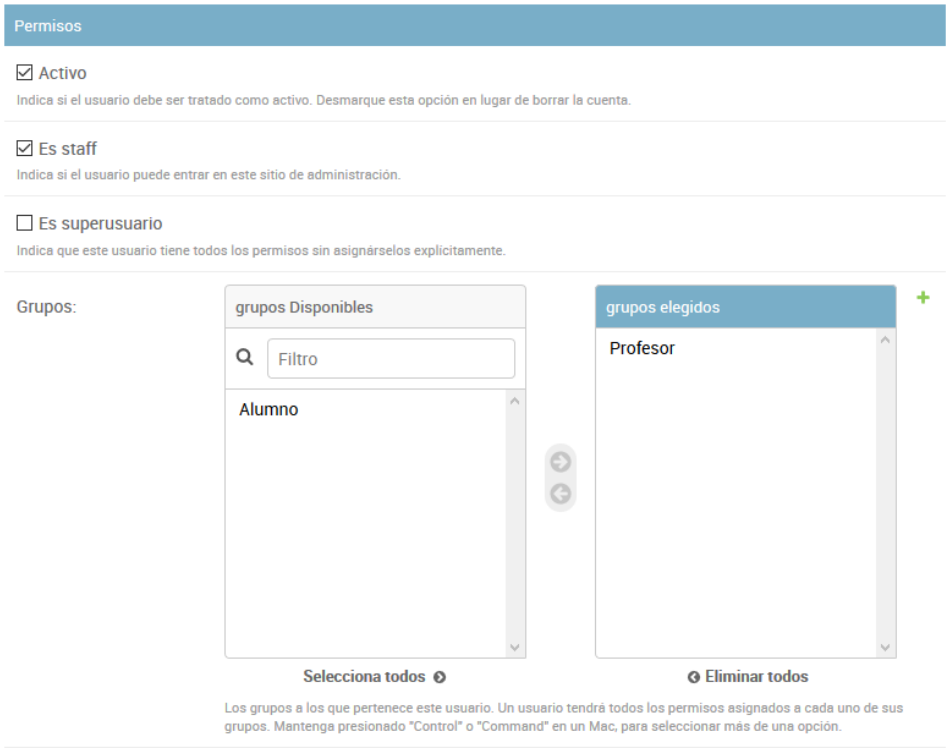


Imagen 5.9: Manejo de Grupos de un usuario

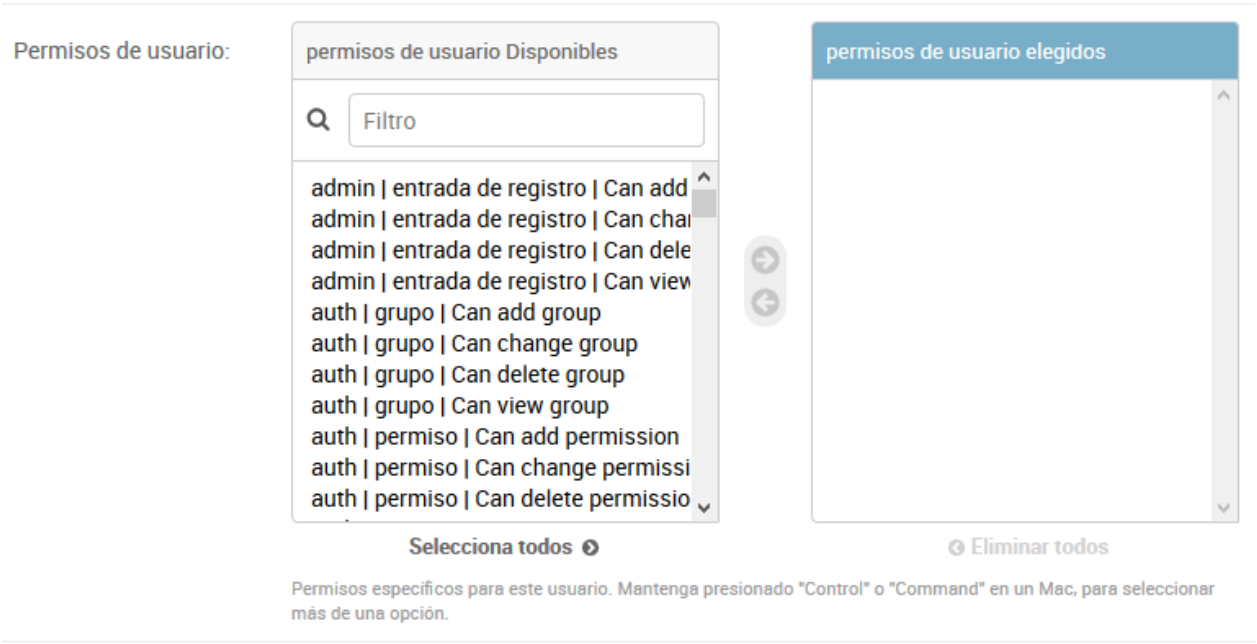


Imagen 5.10: Manejo de permisos de usuario.

5.2 Exámenes

5.2.1 Catálogo de preguntas

Esta sección permite manejar la información del catálogo de preguntas. Sirve para almacenar preguntas propuestas por los profesores y revisadas por la academia, dichas preguntas podrán ser seleccionadas por un profesor para la creación de exámenes para sus grupos.

☐	PREGUNTA	TEMA	FECHA DE CREACIÓN
☐	Pregunta1	Tema 1	5 de Marzo de 2019 a las 16:07
☐	Pregunta2	1,2	8 de Marzo de 2019 a las 14:25

Al añadir una pregunta del catálogo se debe registrar la pregunta, el tema al cual pertenece, las opciones de respuesta indicando cual es la opción correcta y la retroalimentación asociada a la respuesta, además se tiene un campo de vigencia que funciona como una bandera que permite que la pregunta sea usada ó no, en caso de desactivar este recuadro la pregunta no estará disponible al momento de la creación de un examen.

Esta opción sólo aparece al usuario administrador del sistema ya que todas las preguntas deben de pasar por una validación de la coordinación, el profesor podrá entregar propuestas de preguntas de manera escrita, de manera que una vez aceptadas el administrador del sistema las pueda agregar al catálogo.

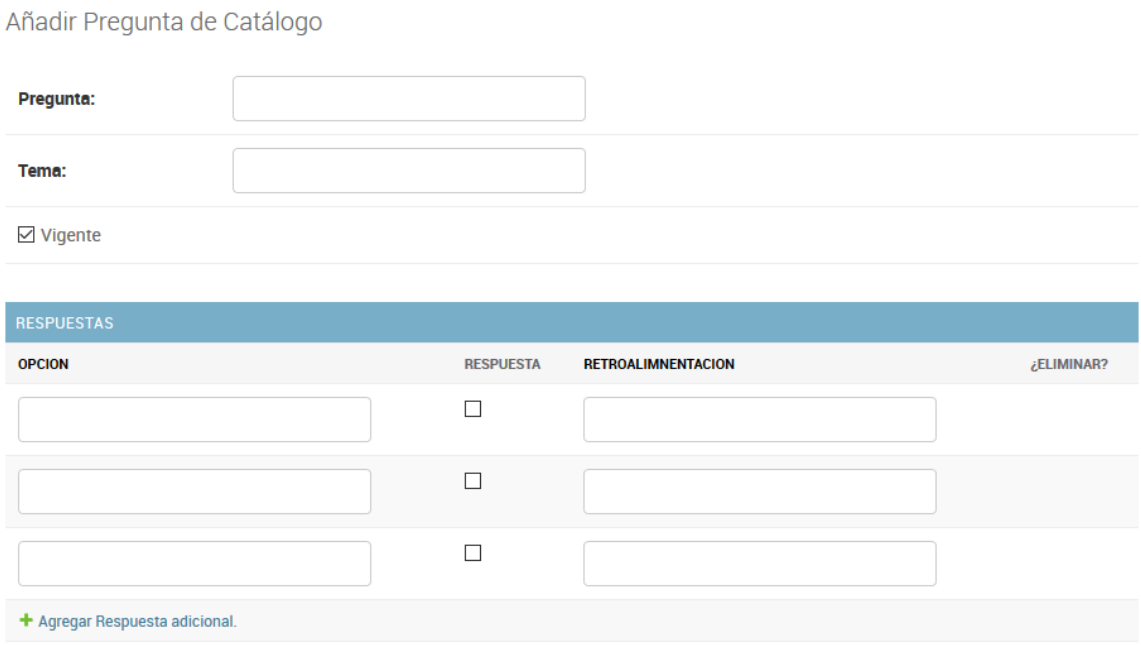


Imagen 5.11: Creación de preguntas para cátalogo

5.2.2 Entrega de exámenes

Esta sección está habilitada para el administrador del sistema y para alumnos. Los alumnos sólo podrán ver los resultados de sus exámenes, mientras que el administrador tiene acceso a visualizar el resultado de cualquier alumno.

Imagen 5.12: Vista del panel de entrega de exámenes siendo súper usuario.

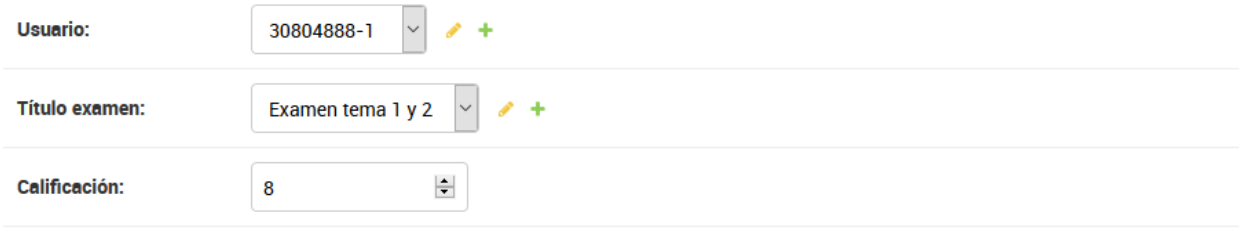


Imagen 5.13: Vista de edición de entrega de exámenes siendo súper usuario.

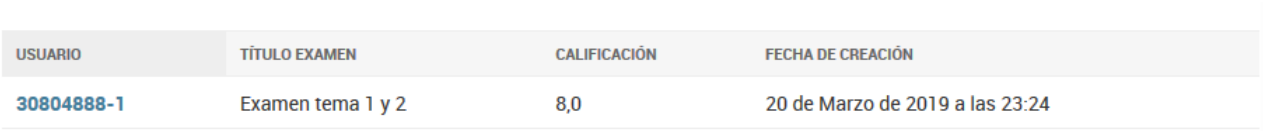


Imagen 5.14: Vista del panel de entrega de exámenes siendo Alumno

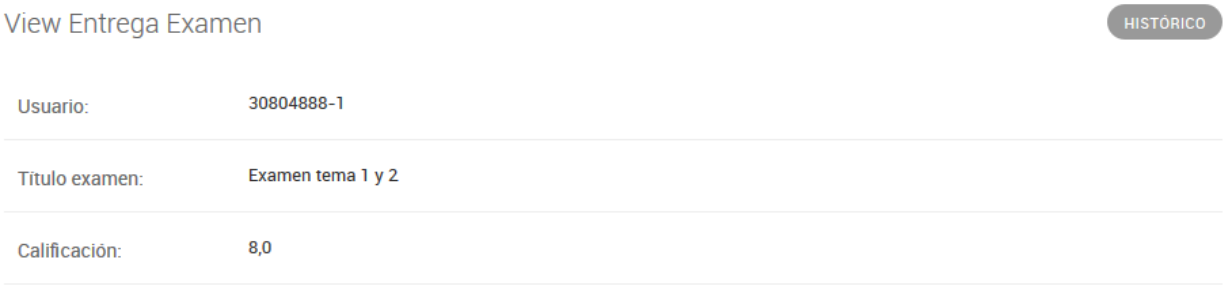


Imagen 5.15: Vista de detalles del examen siendo Alumno.

5.2.3 Exámenes

Esta sección está habilitada para el súper usuario y para miembros del grupo Profesor. El súper usuario podrá crear, editar, ver y borrar exámenes para todos los grupos mientras que el profesor únicamente podrá crear, editar y ver los exámenes creados por él para su grupo. La creación de exámenes por parte del profesor se recomienda sea realizada mediante la vista diseñada para esto (para detalles consultar manual de usuario) ya que esa vista permite filtrar las preguntas por temas, funcionalidad que en el panel de administración de Django no es posible.

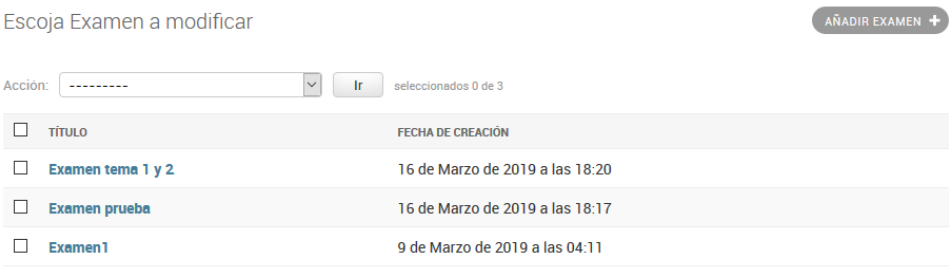


Imagen 5.16: Vista de exámenes siendo súper usuario.

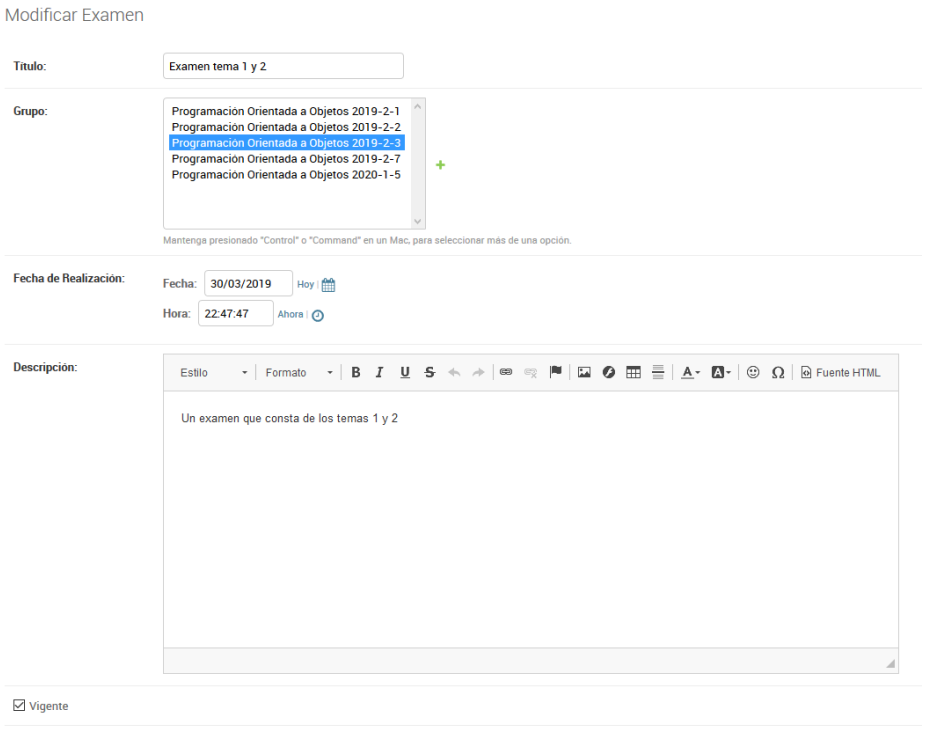


Imagen 5.17: Vista de edición de examen siendo súper usuario.

Imagen 5.18: Modificación de preguntas del examen siendo súper usuario.

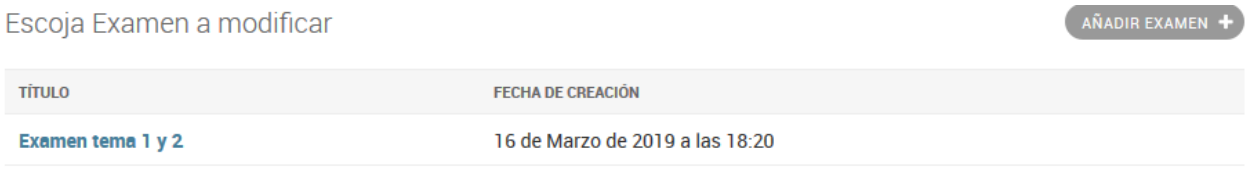


Imagen 5.19: Vista de exámenes siendo Profesor.

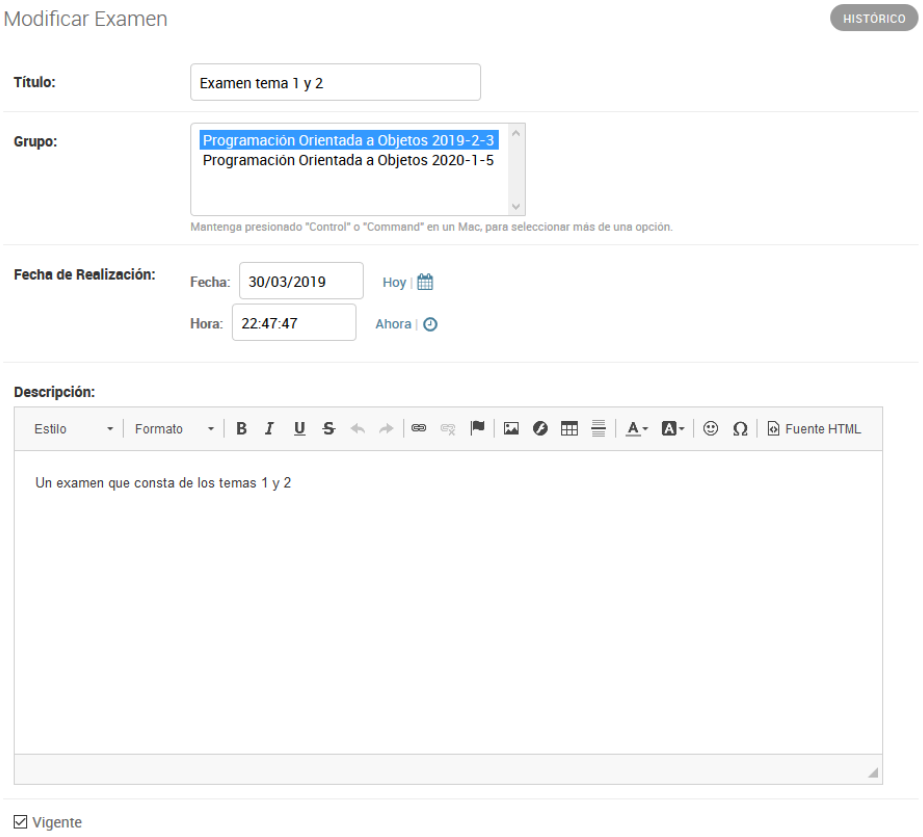


Imagen 5.20: Vista de edición de examen siendo Profesor

Imagen 5.21 Modificación de preguntas siendo Profesor

5.2.4 Preguntas de examen

Esta sección está habilitada para profesores y para el súper usuario, permite añadir una pregunta del catálogo de un examen ya creado.

Imagen 5.22: Panel de preguntas de examen registradas.

Imagen 5.23: Modificación de detalles de una pregunta de examen.

5.3 Gestor de ejercicios de prácticas

Esta sección sólo está habilitada para el súper usuario y permite manejar los ejercicios de las prácticas, así como sus preguntas y respuestas de los ejercicios que no generan calificación

Imagen 5.24: Panel del gestor de ejercicios de prácticas.

5.3.1 Ejercicios

Esta sección sirve para crear y modificar los ejercicios que no generan calificación (ejercicios de realimentación).

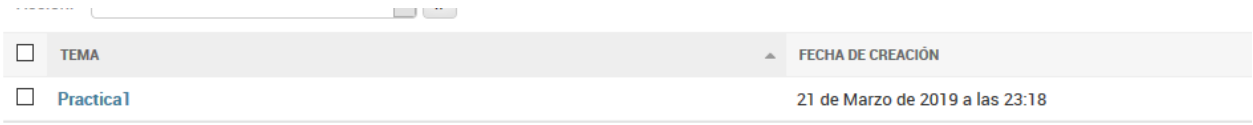


Imagen 5.25: Panel de ejercicios.

Al acceder a un ejercicio se presenta la oportunidad de añadir preguntas al mismo.

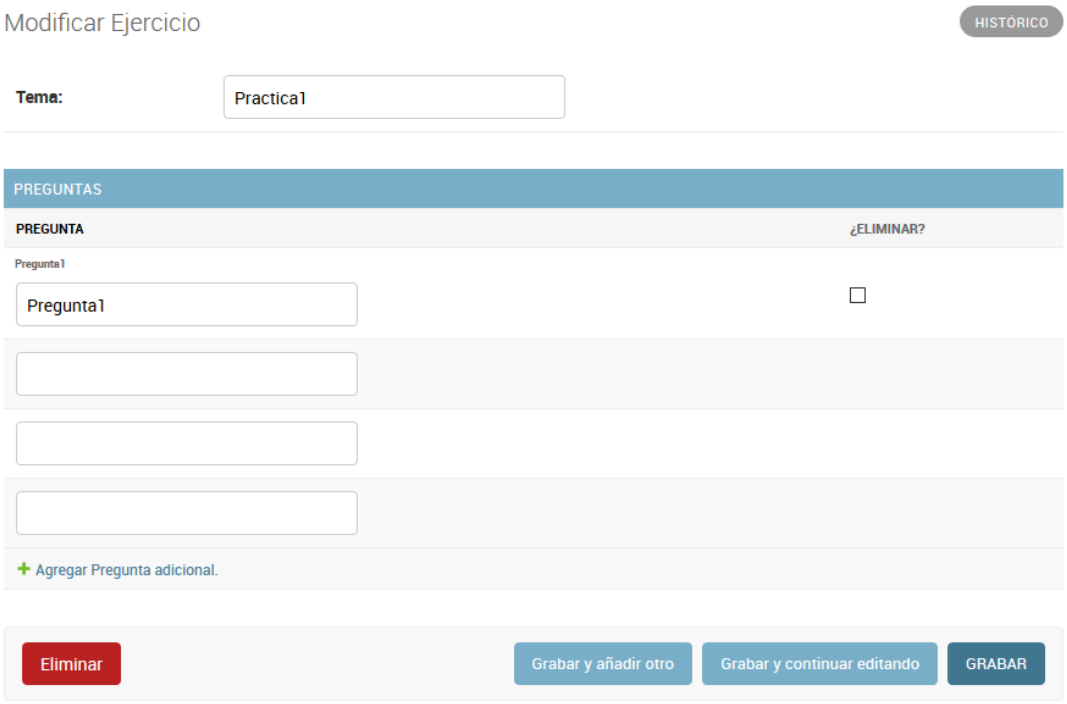


Imagen 5.26: Panel de modificación de ejercicios.

5.3.2 Preguntas

En esta sección se permite añadir nuevas preguntas a un ejercicio creado ó añadir las opciones de respuesta a dichas preguntas.

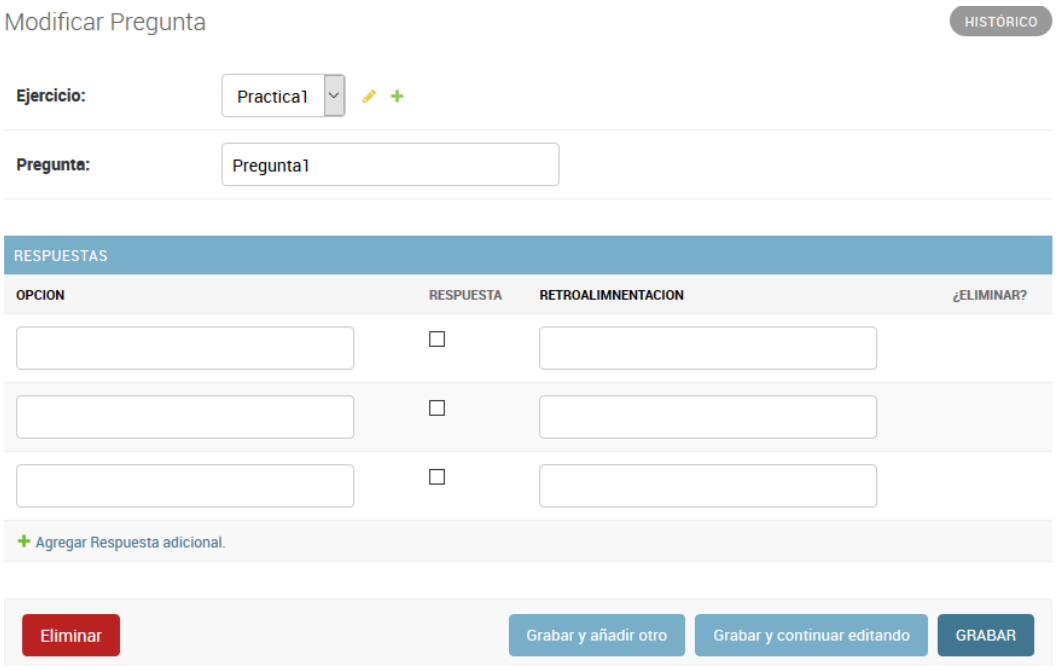


Imagen 5.27 Panel de modificación de pregunta.

5.3.3 Respuestas

Esta sección permite modificar las respuestas a una pregunta creada mediante cualquiera de los métodos anteriores ó crear una respuesta nueva a una pregunta de un ejercicio ya existente

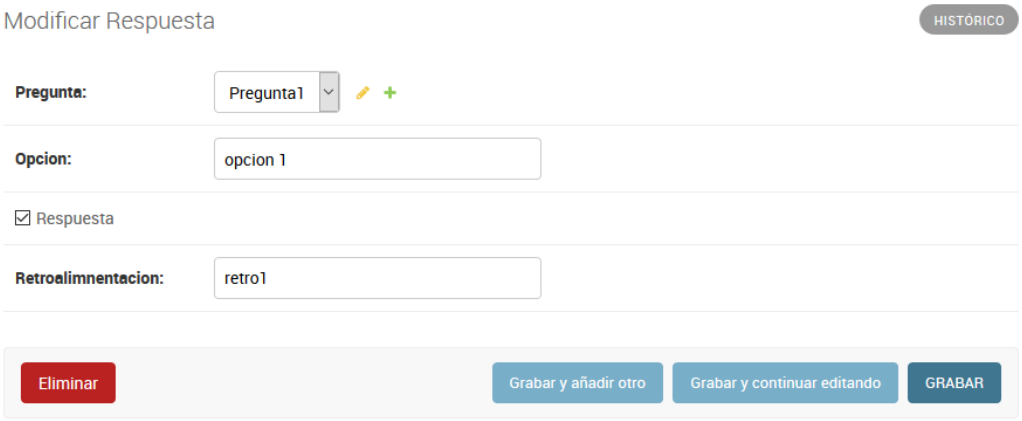


Imagen 5.28 Panel de modificación de respuestas almacenadas.

5.4 Gestor de páginas

Esta sección está habilitada para el súper usuario y para usuarios que pertenecen al grupo Profesor. Éstas paginas sirven para crear comunicados en el portal y que los alumnos puedan verlos en todo momento.

5.4.1 Paginas

Esta sección muestra la lista de páginas creadas. Para el súper usuario se muestran todas las paginas creadas por cualquier profesor, mientras que para el grupo Profesor sólo se muestran las paginas creadas para sus grupos.

Imagen 5.30: Panel de Páginas.

En el panel de edición de las paginas se puede modificar el título, texto y el grupo al que va dirigida la página, el orden (un indicador de prioridad, siendo 0 el de más alta prioridad, funciona para mostrar en orden de importancia las paginas para el alumno).

Para el usuario miembro del grupo Profesor la lista de grupos seleccionables está filtrada para mostrar únicamente los grupos a los cuales fue asignado.

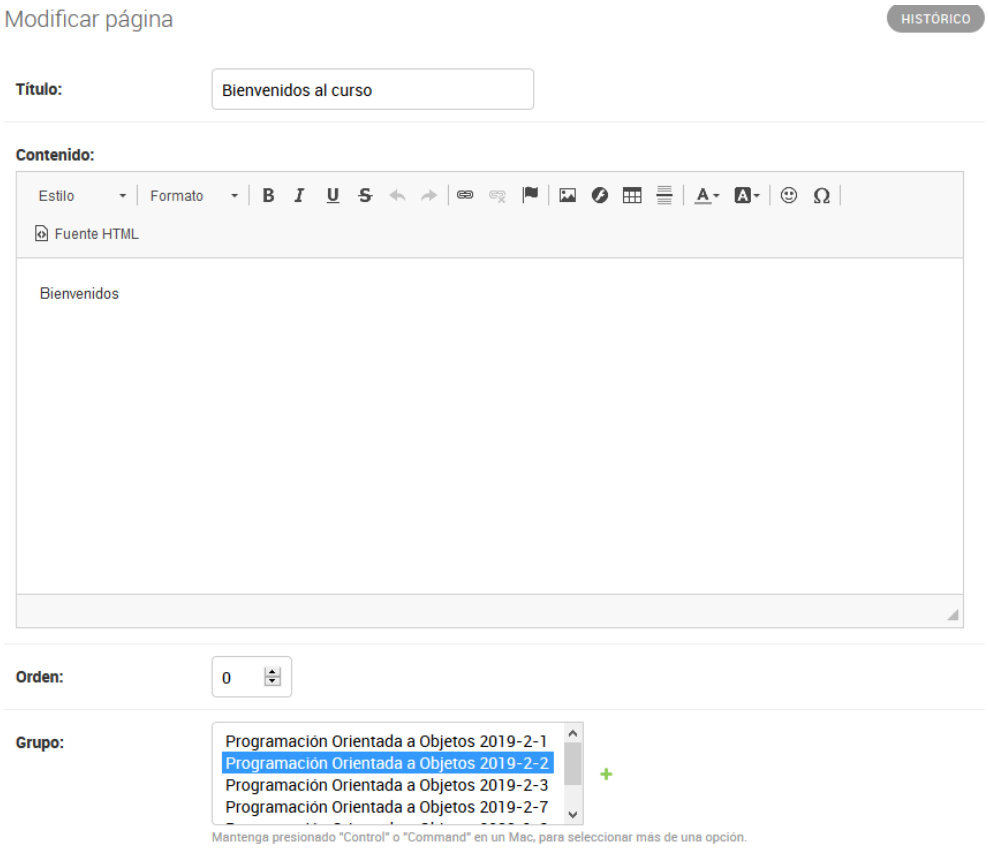


Imagen 5.31: Modificación de página.

5.5 Grupos

Esta sección permite manejar la información de los grupos de alumnos. Django no permite carga masiva de usuarios, por lo cual existe una vista en la plataforma encargada de esta labor, esta sección debe utilizarse sólo para cambios pequeños. Muchas funcionalidades de esta sección están limitadas al súper usuario.

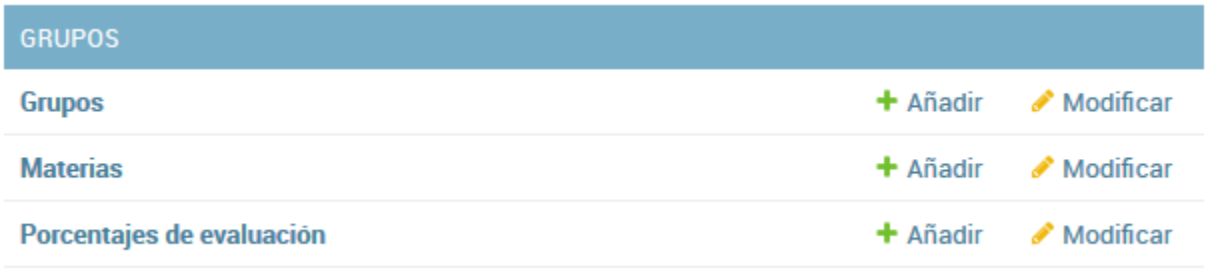


Imagen 5.32 Panel de grupos.

5.5.1 Grupos

Esta sección muestra la lista de los grupos en la plataforma, para los usuarios miembro del grupo Profesor la funcionalidad está limitada a sólo ver los grupos en los cuales él está asignado, mientras que el súper usuario puede crear nuevos grupos, dar de baja ó alta a usuarios de un grupo ó modificar información de éste.

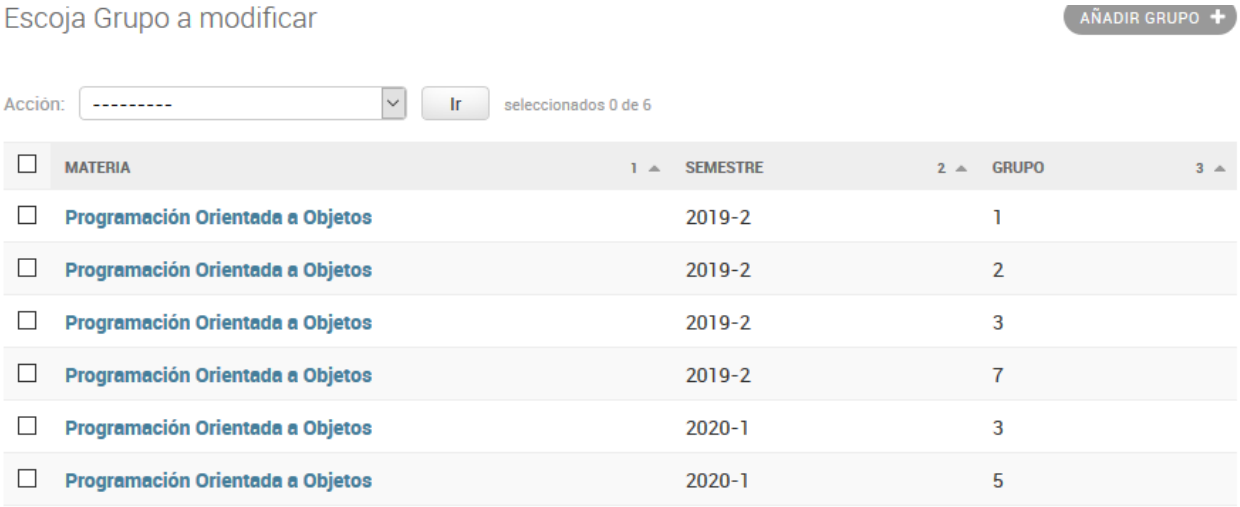


Imagen 5.33 Panel de grupos.

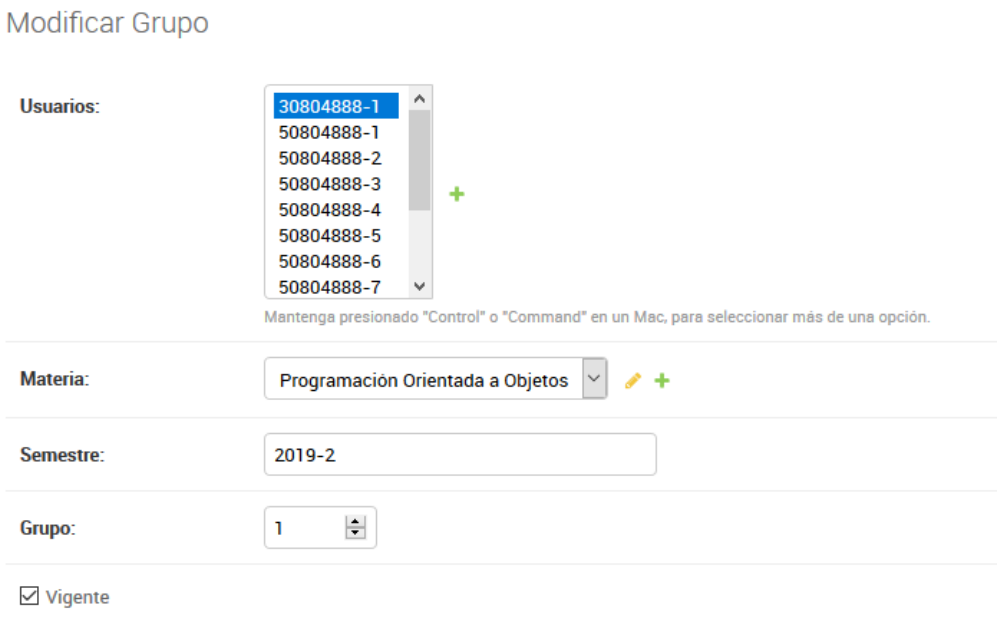


Imagen 5.34: Vista de modificación de un grupo para súper usuario

Imagen 5.35: Vista de detalles de un grupo para miembros de la clase Profesor

5.5.2 Materias

Esta sección permite crear materias, el alcance de este proyecto es para la materia Programación orientada a objetos, pero se diseñó de manera que puedan existir diferentes materias dentro de la plataforma.

Esta funcionalidad sólo está habilitada para el súper usuario ya que se requiere tener un control estricto de esta información.

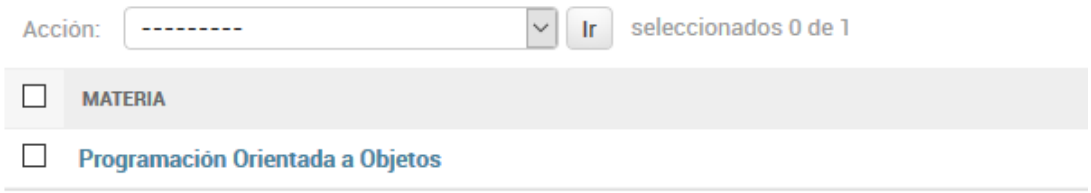


Imagen 5.36: Panel de materias.

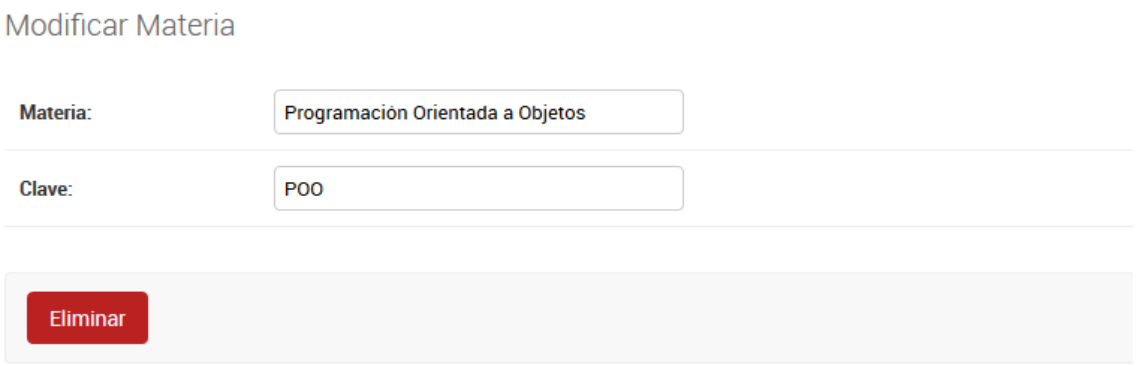


Imagen 5.37: Vista de edición de materias para súper usuario

5.5.3 Porcentajes de evaluación

Esta sección está habilitada para súper usuario, miembros del grupo Profesor y Alumnos. Está diseñada para que el Profesor asigne los porcentajes de evaluación de los grupos donde esté asignado, los Alumnos únicamente podrán visualizar los porcentajes de evaluación y el súper usuario tiene acceso a la modificación ó creación de porcentajes de evaluación de cualquier grupo registrado en la plataforma.

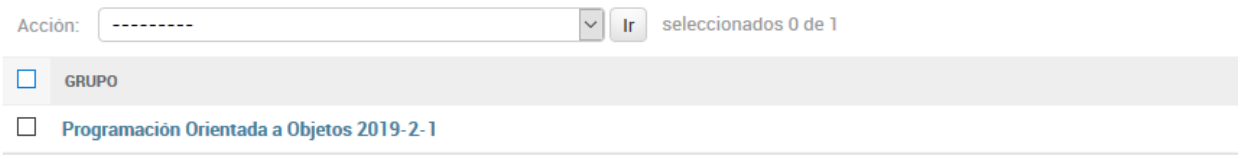


Imagen 5.38: Panel de porcentajes de evaluación

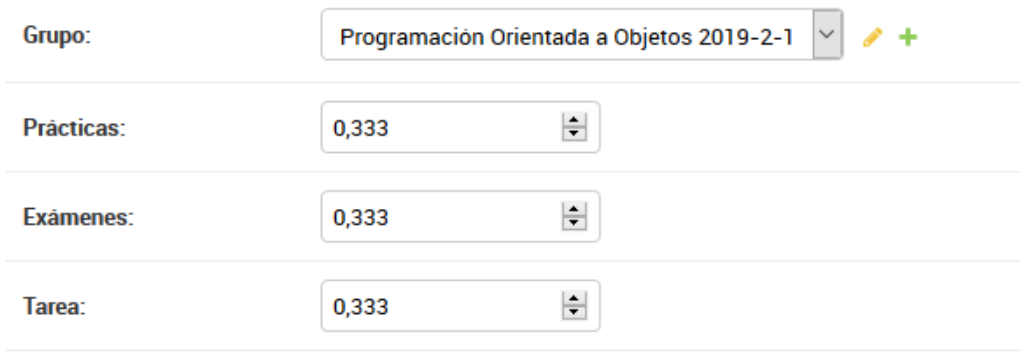


Imagen 5.39: Vista de detalles de porcentajes de evaluación siendo súper usuario ó Profesor

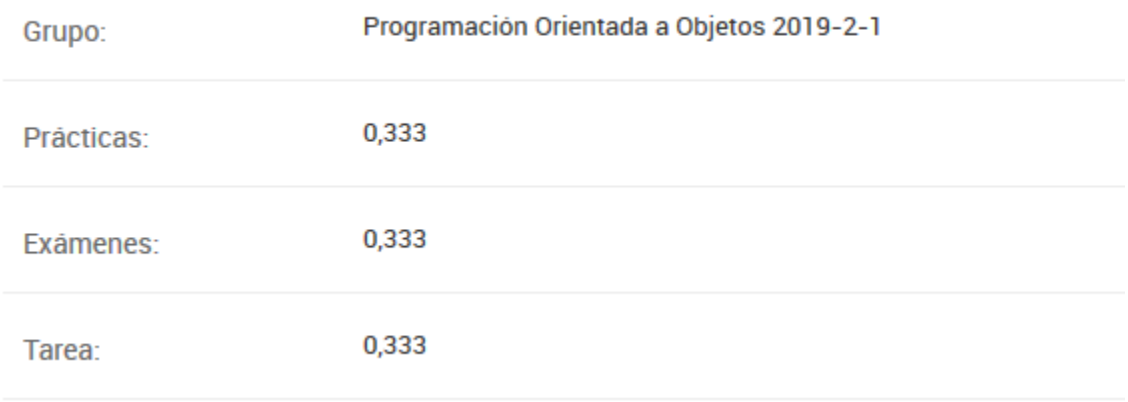


Imagen 5.40: Vista de detalles de porcentajes de evaluación siendo Alumno

5.6 Tareas

Esta sección se encarga de la creación de tareas y de las entregas de los alumnos, está habilitada para el súper usuario y los grupos Alumno y Profesor.

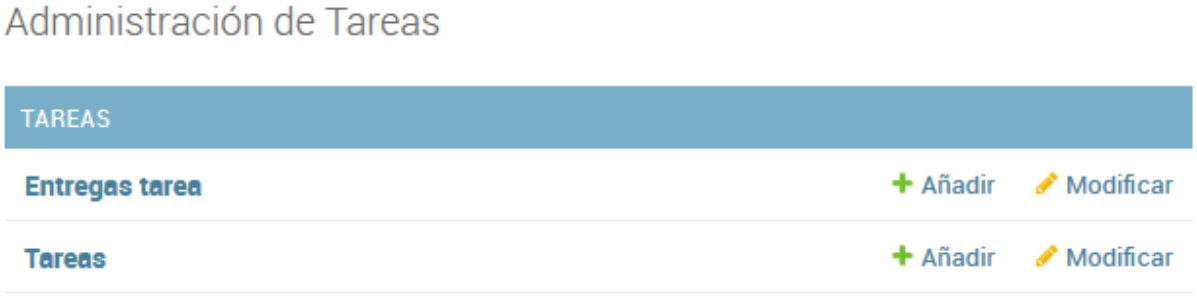


Imagen 5.41: Panel de tareas

5.6.1 Entrega de tarea

Esta sección tiene diferentes permisos dependiendo del usuario. Para el súper usuario se muestran todas las entregas de todos los alumnos. Para el alumno sólo se muestran sus tareas entregadas y no pueden ser modificadas por él, para esta acción existe una vista especial en la plataforma. Para los profesores se muestran las entregas de sus alumnos, el profesor puede añadir la calificación del entregable de sus alumnos.

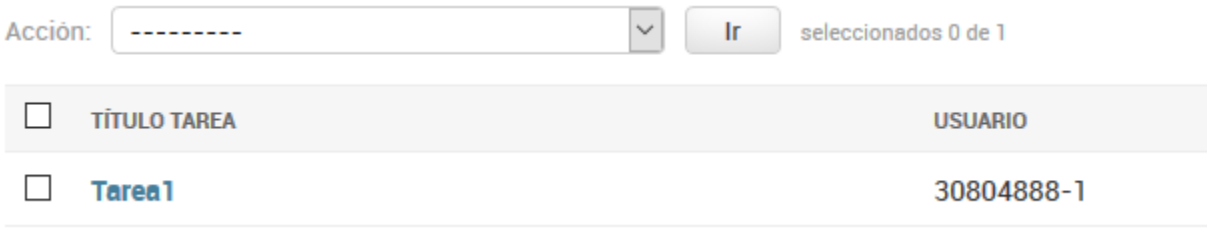


Imagen 5.42: Panel de entregas de tarea.

Imagen 5.43: Vista de detalle de una entrega de tarea para el súper usuario y Profesor

Imagen 5.44: Vista de detalle de una entrega de tarea para un Alumno

5.6.2 Tareas

Esta sección sólo está habilitada para el súper usuario y usuarios miembros del grupo Profesor. Desde este panel se puede crear una nueva tarea a un grupo que aparecerá disponible para que el alumno pueda subir el entregable en una vista de la plataforma, el súper usuario puede añadir una tarea a cualquier grupo registrado mientras que el profesor sólo podrá añadir tareas a un grupo al cuál esté asignado.

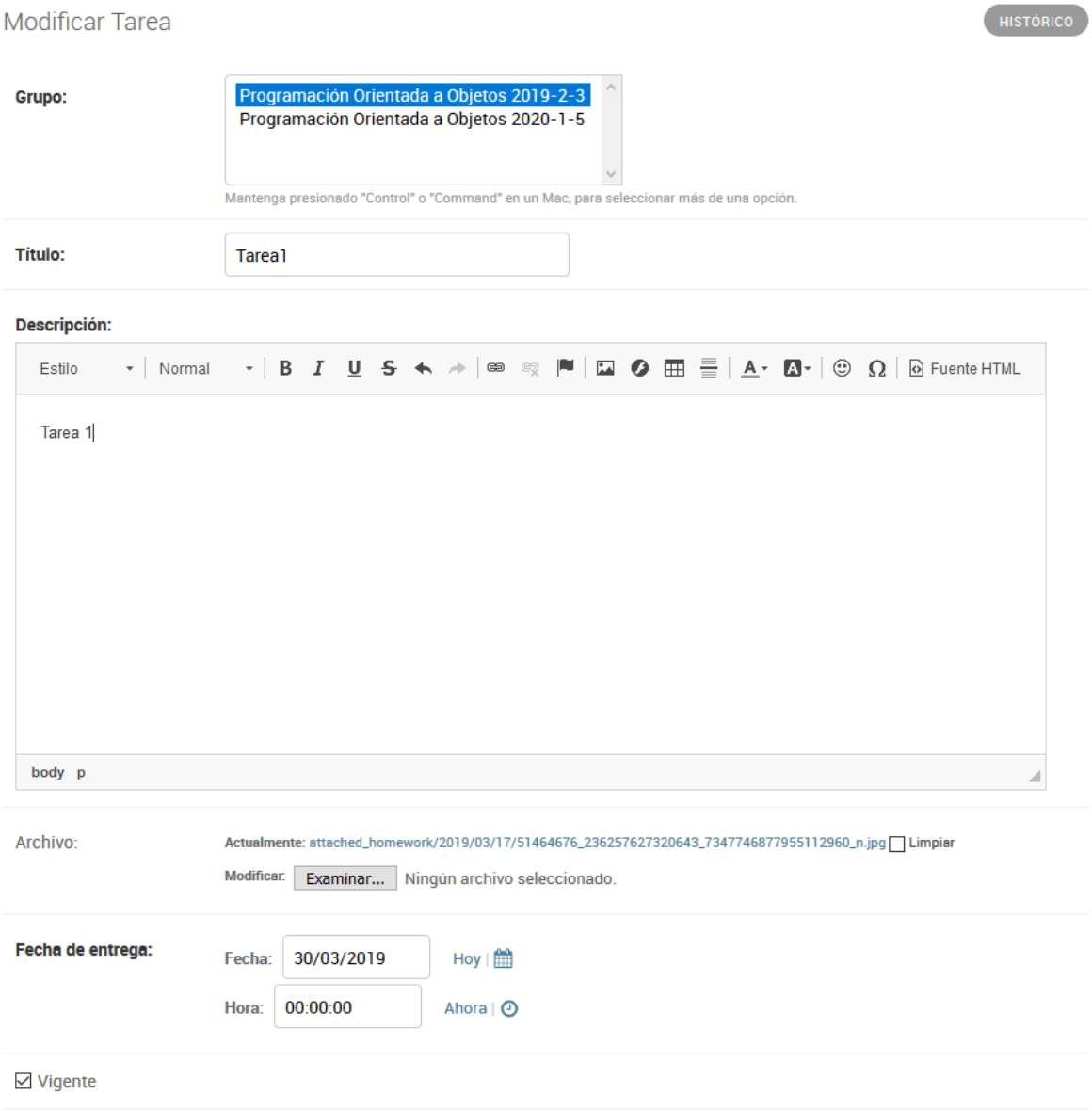


Imagen 5.45: Modificación de una tarea.

5.7 Encuestas

Esta sección es la encargada de mantener el control y almacenamiento de las encuestas realizadas por cada practica y de las encuestas al final de curso, la sección está disponible únicamente para el súper usuario.

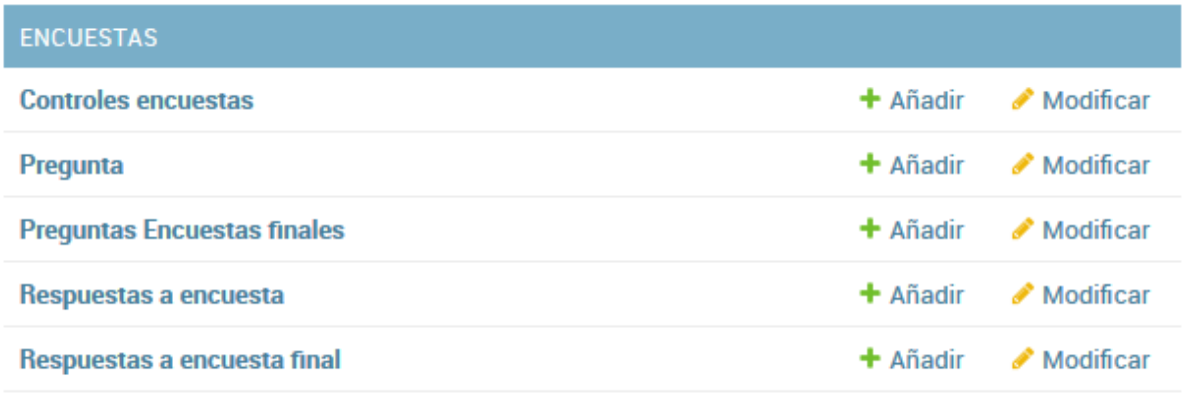


Imagen 5.46: Panel de encuestas

5.7.1 Controles encuestas

Esta sección almacena una variable que permite habilitar la encuesta final, si esta bandera está desactivada los alumnos no podrán realizar la encuesta final, la razón para almacenar dicha variable es por los tiempos del curso, la encuesta final no debe de estar habilitada en todo momento, sólo en los últimos días del curso.

Imagen 5.47: Variable de control de encuesta final.

5.7.2 Pregunta

En esta sección es la encargada de almacenar las preguntas que aparecen en la encuesta por práctica, cada pregunta tiene una categoría, un número de pregunta y una bandera que indica si la pregunta tiene un trato especial, dicha bandera se utiliza en el sistema para determinar si la pregunta consiste de un rango de valores ó contiene campos “Sí”, “No” ó “No Aplica”

	NUMERO	PREGUNTA	CATEGORIA
☐	17	¿Consideras que se cumplieron los objetivos de la práctica?	OBJETIVOS DE LA PRÁCTICA
☐	16	¿El profesor dio a conocer el objetivo al inicio de la práctica?	OBJETIVOS DE LA PRÁCTICA
☐	15	¿Las fallas se atendieron oportunamente?	ATENCIÓN A FALLAS
☐	14	¿Se presentaron fallas del equipo de cómputo durante las prácticas?	ATENCIÓN A FALLAS
☐	13	¿El servicio proporcionado en el laboratorio cumplió con tus expectativas iniciales en cuanto a los conocimientos y habilidades adquiridas?:	SATISFACCIÓN DEL SERVICIO
☐	12	Mi satisfacción respecto al servicio proporcionado en el laboratorio fue:	SATISFACCIÓN DEL SERVICIO
☐	11	Consideras que el orden, la limpieza y las condiciones de seguridad fueron:	INSTALACIONES
☐	10	¿Se cuenta con las mesas, sillas y/o bancos necesarios para la realización de las prácticas?	INSTALACIONES
☐	9	Las instalaciones del laboratorio fueron:	INSTALACIONES
☐	8	La puntualidad al iniciar la práctica fue:	PROFESOR
☐	7	La aclaración de dudas durante el desarrollo de la práctica fue:	PROFESOR
☐	6	La manera en que el profesor explicó y supervisó el desarrollo de la práctica fue:	PROFESOR
☐	5	En caso de haber requerido la atención por parte del personal del laboratorio, ésta fue:	PERSONAL DEL LABORATORIO
☐	4	La disponibilidad de la página web del laboratorio fue:	DESARROLLO DE LAS PRÁCTICAS
☐	3	El funcionamiento del software fue:	DESARROLLO DE LAS PRÁCTICAS
☐	2	El funcionamiento del equipo de cómputo para realizar la práctica fue:	DESARROLLO DE LAS PRÁCTICAS
☐	1	La disponibilidad del equipo de cómputo para realizar la práctica fue:	DESARROLLO DE LAS PRÁCTICAS

Imagen 5.48: Preguntas almacenadas correspondientes a las encuestas por práctica

5.7.3 Preguntas encuestas finales

En esta sección se tienen almacenadas las preguntas dedicadas a la encuesta de evaluación al final del curso, de la misma manera que en la preguntas de la encuesta por practica, se almacena una bandera que indica si se requiere un trato especial en dicha pregunta.

NUMERO	PREGUNTA	CATEGORIA
12	¿Las fallas se atendieron oportunamente?	ATENCIÓN A FALLAS
11	¿Se presentaron fallas del equipo de cómputo durante las prácticas?	ATENCIÓN A FALLAS
10	¿El servicio proporcionado por el laboratorio cumplió con tus expectativas iniciales en cuanto a los conocimientos y habilidades adquiridas?:	SATISFACCIÓN DEL SERVICIO
9	Mi satisfacción respecto al servicio proporcionado en el laboratorio fue:	SATISFACCIÓN DEL SERVICIO
8	Consideras que el orden, la limpieza y condiciones de seguridad fueron:	INSTALACIONES
7	¿Se cuenta con las mesas, sillas y/o bancos para la realización de las prácticas?	INSTALACIONES
6	Las instalaciones del laboratorio fueron:	INSTALACIONES
5	En caso de haber requerido la atención por parte del personal del laboratorio, ésta fue:	PERSONAL DEL LABORATORIO
4	La disponibilidad de la página web del laboratorio fue:	DESARROLLO DE LAS PRÁCTICAS
3	El funcionamiento del software fue:	DESARROLLO DE LAS PRÁCTICAS
2	El funcionamiento del equipo de cómputo para realizar las prácticas fue:	DESARROLLO DE LAS PRÁCTICAS
1	La disponibilidad del equipo de cómputo para realizar las prácticas fue:	DESARROLLO DE LAS PRÁCTICAS

Imagen 5.49: Preguntas almacenadas correspondientes a la encuesta al final del curso.

5.7.4 Respuestas a encuesta

En esta sección se almacenan las respuestas de los alumnos a las encuestas de cada practica, el sistema se encarga de no permitirle al usuario contestar la encuesta dos veces y así evitar duplicados.

PREGUNTA	VALOR NUMERICO	VERDADERO/FALSO	NO APLICA	PRACTICA
¿Consideras que se cumplieron los objetivos de la práctica?	-	✗	?	1
¿El profesor dio a conocer el objetivo al inicio de la práctica?	-	✓	?	1
¿Las fallas se atendieron oportunamente?	-	?	✓	1
¿Se presentaron fallas del equipo de cómputo durante las prácticas?	-	✗	?	1
¿El servicio proporcionado en el laboratorio cumplió con tus expectativas iniciales en cuanto a los conocimientos y habilidades adquiridas?:	100	?	?	1

Imagen 5.50: Respuestas almacenadas generadas por los alumnos en la evaluación por práctica.

5.7.5 Respuestas a encuesta final

Esta sección almacena las respuestas de los alumnos a la encuesta al final de curso, el sistema se encarga de manera automática de evitar que el mismo usuario conteste varias veces la encuesta, a diferencia de las respuestas a las practicas, en esta se almacena el grupo evaluado.

	PREGUNTA	VALOR NUMERICO	VERDADERO/FALSO	NO APLICA	GRUPO
☐	¿Las fallas se atendieron oportunamente?	-	✓	?	Programac
☐	¿Se presentaron fallas del equipo de cómputo durante las prácticas?	-	✗	?	Programac
☐	¿El servicio proporcionado por el laboratorio cumplió con tus expectativas iniciales en cuanto a los conocimientos y habilidades adquiridas?:	100	?	?	Programac

Imagen 5.51: Respuestas almacenadas generadas por los alumnos en la evaluación del curso.

6.- Manual de usuario

6.1 General

La plataforma tiene diferentes opciones dependiendo del tipo de usuario que esté autenticado. Este capítulo está dedicado a describir las funcionalidades de la plataforma.

6.1.1 Inicio de sesión

Al entrar a la plataforma se mostrará la página principal junto con una barra de navegación con dos opciones, “Inicio” y “Acceder” para iniciar sesión se da clic en “Acceder” y se insertan las credenciales de usuario.

Imagen 6.1: Pantalla de inicio de sesión.

Al dar clic en acceder se redirecciona a la página del perfil del usuario que ingresó sus credenciales y en la barra de navegación aparecen nuevas opciones.

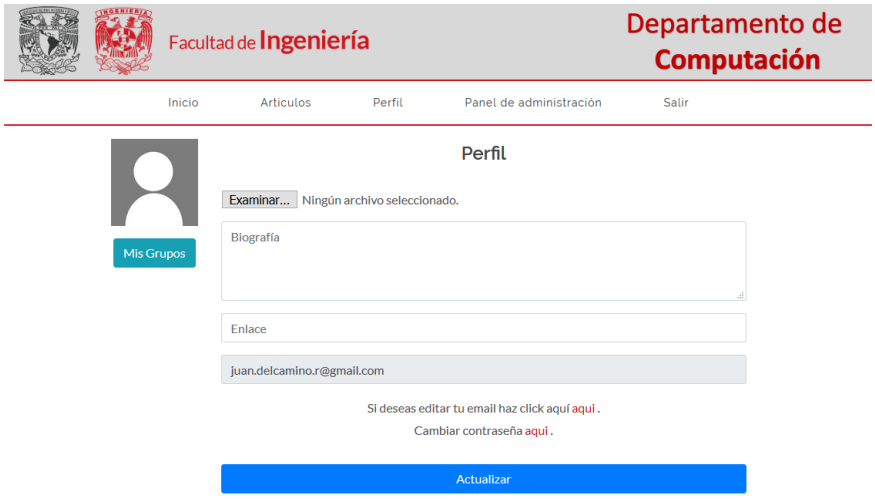


Imagen 6.2: Pantalla de perfil de usuario.

6.2 Artículos

En esta sección se muestran todas las paginas que pertenecen al grupo donde el usuario esté inscrito, dichas páginas son creadas por el administrador del sistema ó por el Profesor asignado al grupo y sirven para notificar al alumno sobre fechas de examen y de entregas de tarea.

Imagen 6.3: Pagina de artículos.

En esta página sólo se muestran vistas previas al artículo, al dar clic en “leer más” se puede acceder al artículo completo.

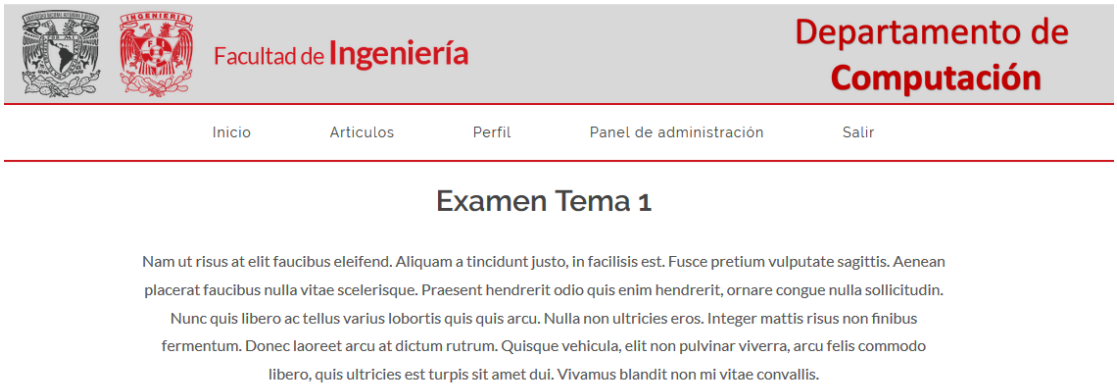


Imagen 6.4: Detalles de un artículo.

6.3 Perfil de usuario

En la página de perfil de usuario se puede modificar la foto de perfil, la biografía ó la descripción, añadir una ruta a una página propia y editar el correo electrónico ó contraseña del usuario.

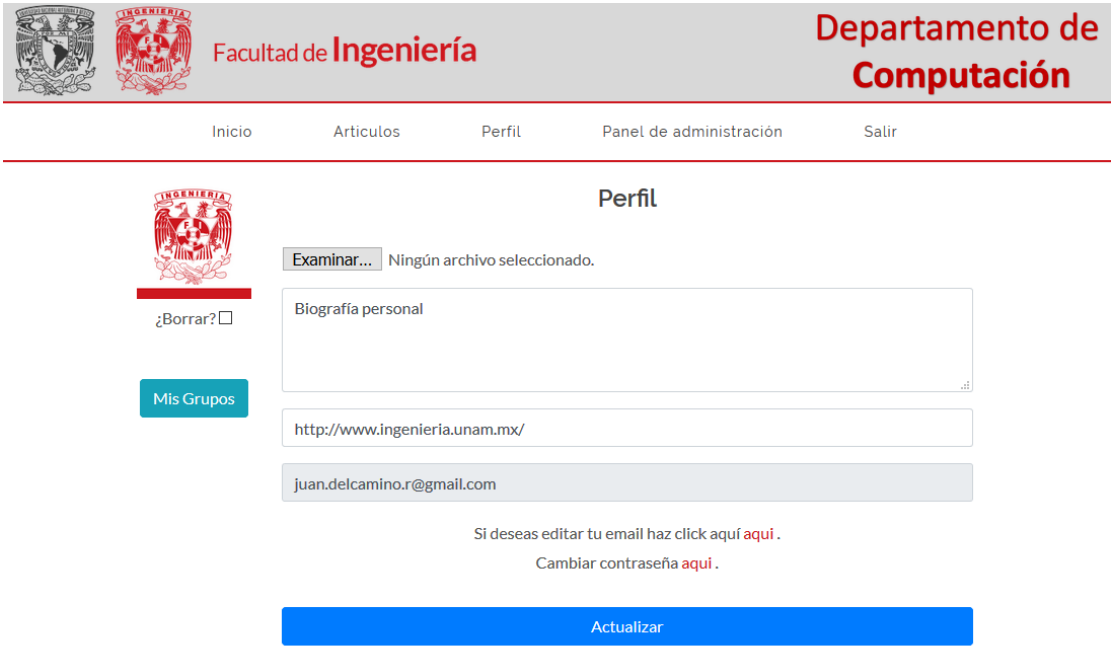


Imagen 6.5: Perfil de usuario con información modificada.

6.3.1 Mis Grupos

En la página de perfil existe un botón que dirige a una página los grupos a los cuales se está inscrito (si es alumno) ó se tiene acceso (si es profesor).

Imagen 6.6: Lista de grupos del usuario.

Al dar clic en alguno de los grupos mostrados, se muestran los detalles del grupo, en esta página se puede ver la lista de tareas asignadas a este grupo así como los exámenes que pueden ser realizados.

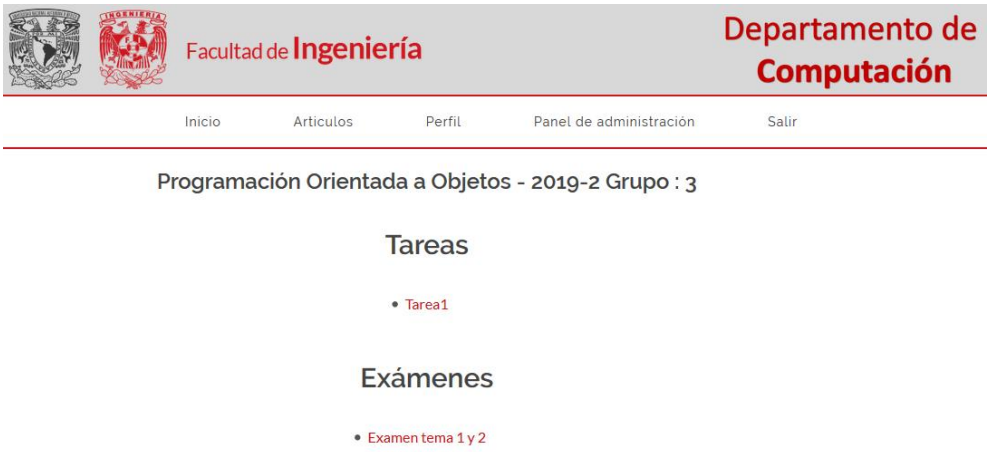


Imagen 6.7: Lista de tareas y exámenes disponibles para el grupo.

6.3.2 Tareas

Al seguir el enlace de una tarea se presenta una vista que permite ver los detalles de la tarea a entregar. En esta página se puede subir el entregable solicitado por la tarea. Una tarea puede tener un archivo adjunto, en caso de existir se mostrará un enlace que permitirá la descarga de éste.

Imagen 6.8: Ejemplo de la descripción de una tarea y su enlace de entrega.

Al dar clic en “Subir archivo” se mostrará una ventana emergente dónde se podrá seleccionar el archivo a entregar.

Imagen 6.9: Formulario para entregar una tarea.

Una vez seleccionado el archivo y al dar clic en el botón enviar la ventana se cerrará y en la pantalla de detalles de tarea se puede ver y modificar el entregable, si así se desea. La posibilidad de cambiar el entregable sólo está disponible hasta la fecha límite de entrega (dada por el profesor), una vez pasada la fecha no se podrá añadir un archivo ni modificar el entregable. Cuando el profesor califique la tarea aparecerá la calificación en esta misma página.

Imagen 6.10: Vista con posibilidad de actualizar la entrega

Imagen 6.11: Vista una vez pasado el tiempo de entrega

Imagen 6.12: Vista con calificación otorgada

6.3.3 Exámenes

Al dar clic en un examen asignado al grupo se permite realizarlo mediante un formulario de opción múltiple como se muestra en la siguiente imagen (las preguntas y respuestas son de prueba):

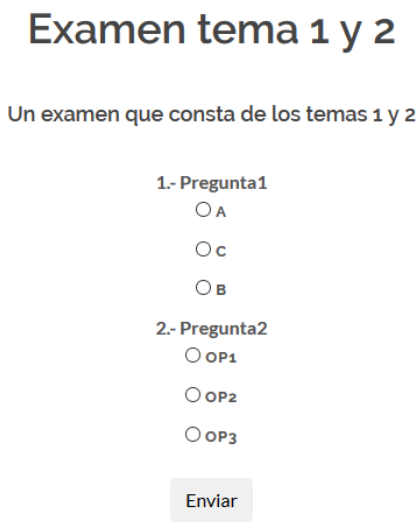


Imagen 6.13: Ejemplo de un formulario de examen.

Al terminar el examen, se redirige al usuario a la página de sus grupos, la siguiente visita a la página del examen se mostrará un mensaje indicando que ya se realizó el examen.

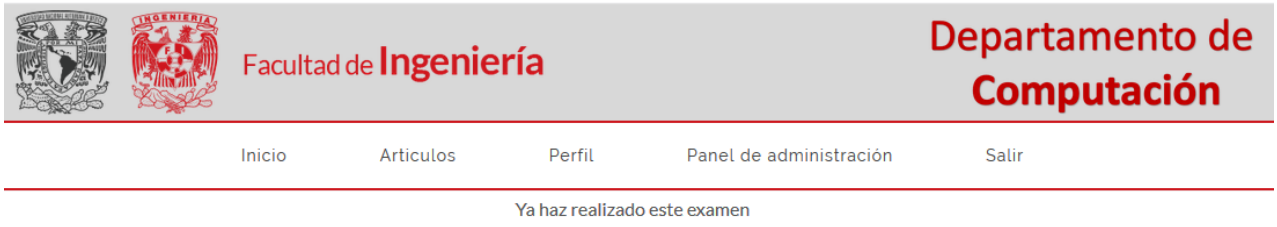


Imagen 6.14: Vista de un examen realizado.

La calificación del examen se puede revisar en el Panel de administración.

6.3.4 Prácticas

En esta sección del grupo aparecen las prácticas disponibles para realizar, al seguir el enlace aparecerá el cuerpo de la practica y en el pie de página de esta aparecerá la opción de realizar los ejercicios correspondientes y la evaluación de la práctica

Imagen 6.15: Sección de prácticas

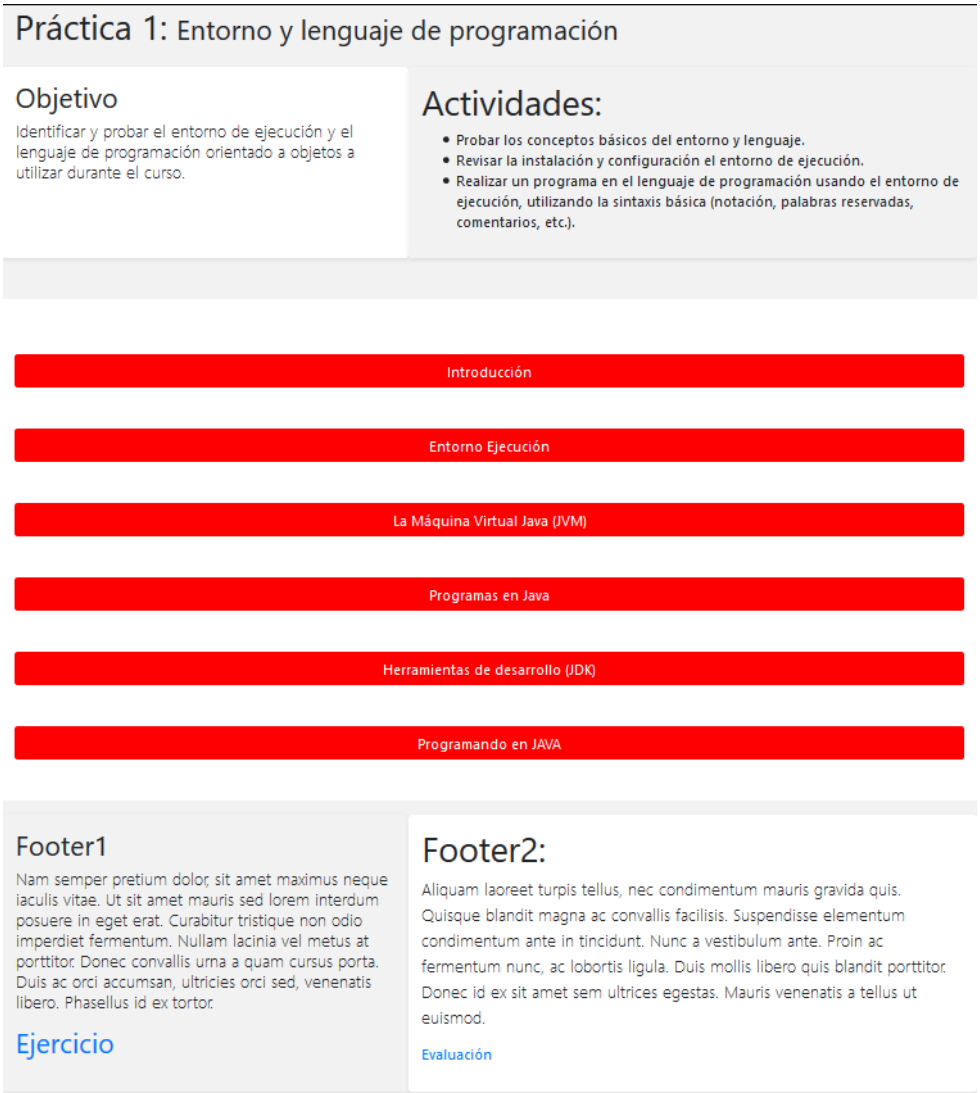


Imagen 6.16: Ejemplo de práctica con sus enlaces a evaluación y ejercicio

Al acceder al enlace del ejercicio se nos presenta una ventana flotante donde se puede realizar el ejercicio asociado a la práctica.

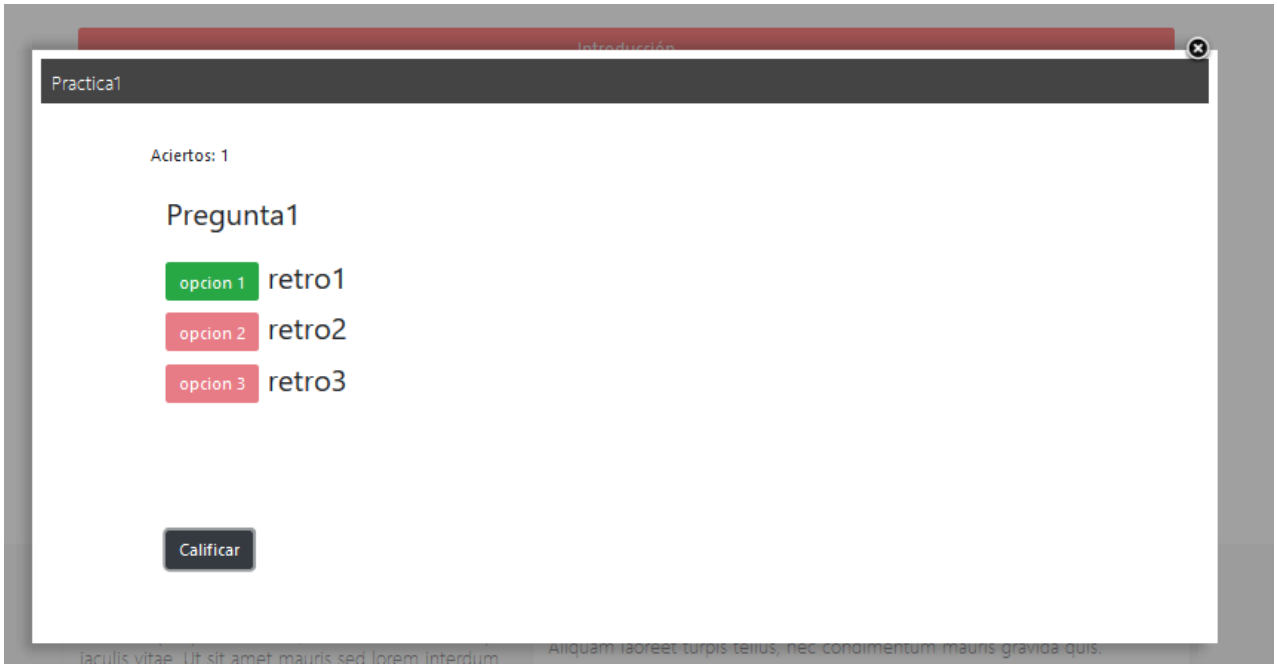


Imagen 6.17: Ejemplo de ejercicio.

Al seguir el enlace de evaluación se nos redirige a la encuesta de evaluación de la práctica.

Imagen 6.18: Algunas preguntas de la encuesta de evaluación.

6.3.5 Encuesta de evaluación final

En esta sección aparece un enlace a la encuesta de evaluación del curso, dicho enlace debe ser habilitado por el administrador, por lo cual sólo estará disponible en los últimos días del curso.

Imagen 6.19: Enlace de la encuesta final del curso.

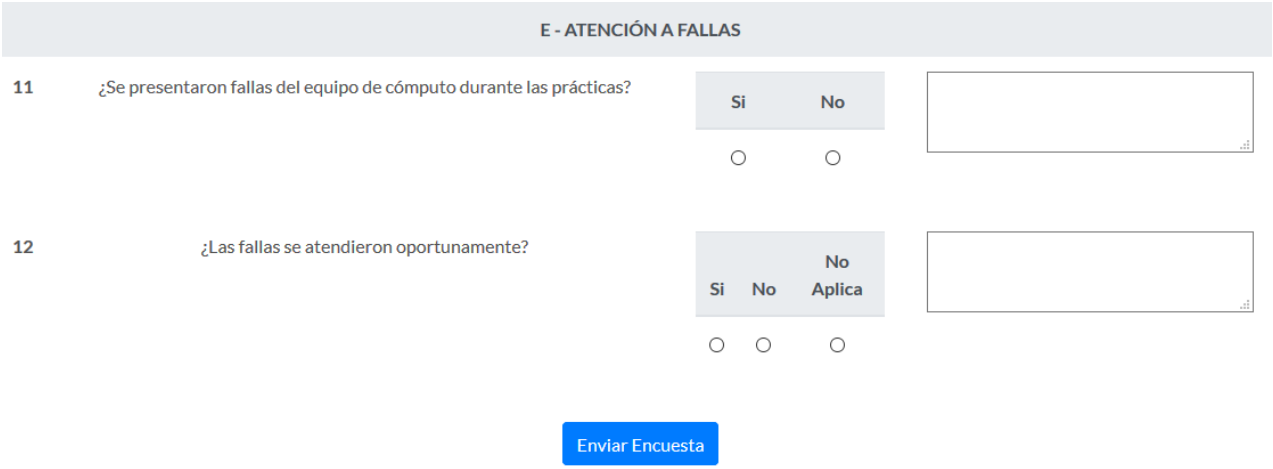


Imagen 6.20: Algunas preguntas de la encuesta de evaluación final.

6.4 Panel de administración

Esta sección tiene diferentes opciones dependiendo del usuario autenticado. Para los alumnos sólo aparece un enlace que dirige a la consola de administración de Django. Para los miembros del grupo Profesor se agrega un enlace a una vista que permite crear exámenes para sus grupos. Para el súper usuario se agrega una opción extra para subir un archivo csv que contenga la información de un grupo para poder realizar una carga masiva de usuarios.

Imagen 6.15: Vista del panel de administración de un Alumno

Imagen 6.16: Vista del panel de administración de un Profesor

Imagen 6.17: Vista del panel de administración del súper usuario.

6.4.1 Panel de administración de contenidos

Este panel y sus funcionalidades fueron descritas en el manual de administración (punto 5 de este documento) por lo cual en esta sección sólo se muestran las interfaces disponibles para cada usuario.

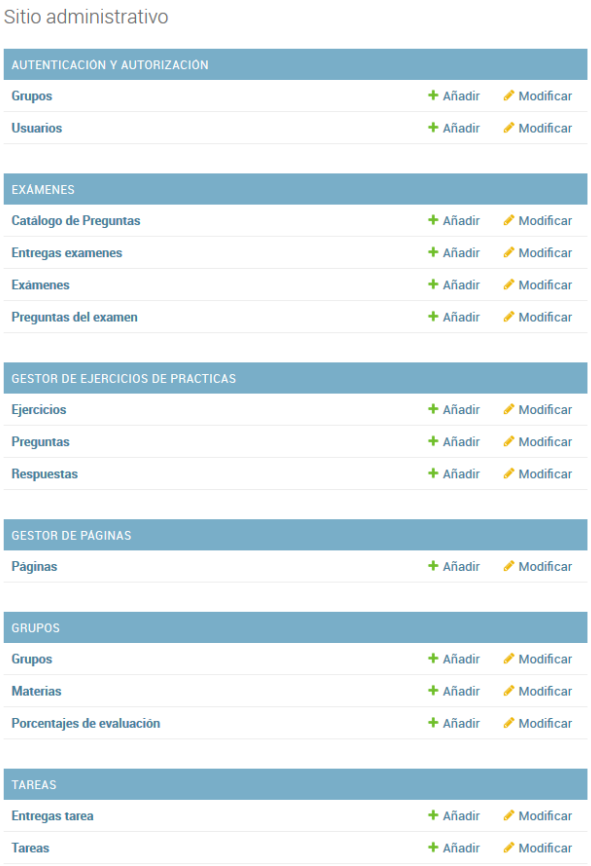


Imagen 6.18: Interfaz de súper usuario

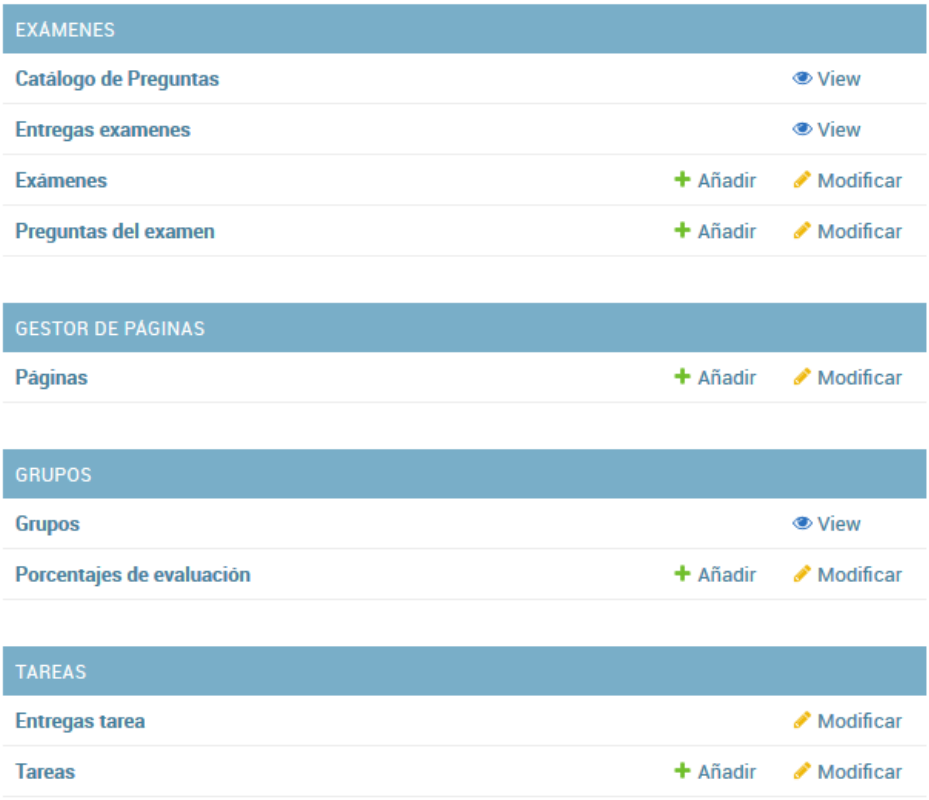


Imagen 6.19: Interfaz de Profesor.

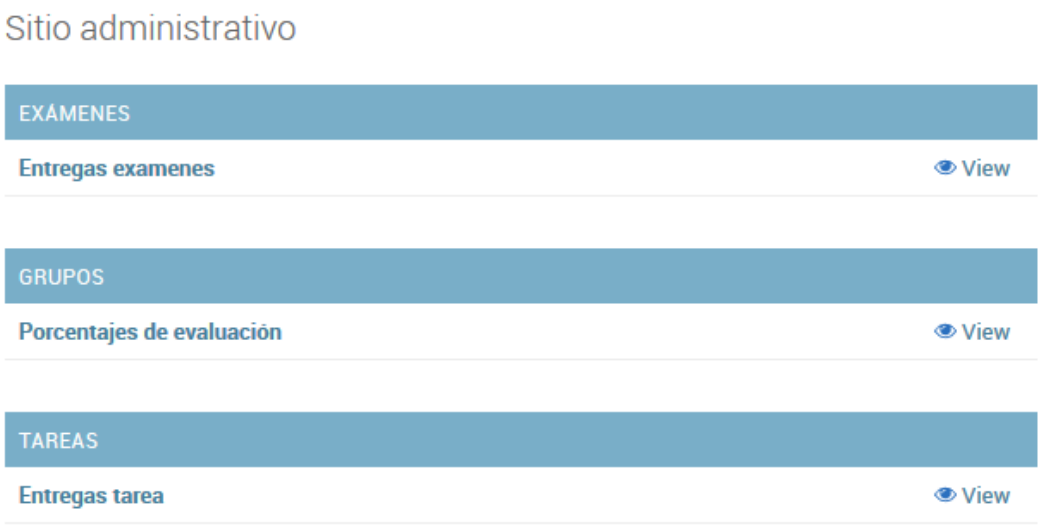


Imagen 6.20: Interfaz de Alumno.

6.4.2 Creación de examen

Esta sección está diseñada para poder registrar un examen en la plataforma. Para esta acción se requiere llenar un formulario y seleccionar preguntas del catálogo para incluir a dicho examen, en esta vista se tiene un buscador incluido para filtrar preguntas por tema, para esto se debe incluir en la barra de búsqueda el número de tema que se desea y dar clic en la lupa, lo cual mostrará en la tabla todas las preguntas que coincidan con dicho tema, en caso de no ingresar ningún tema en la búsqueda se mostrarán todas las preguntas registradas en el catálogo.

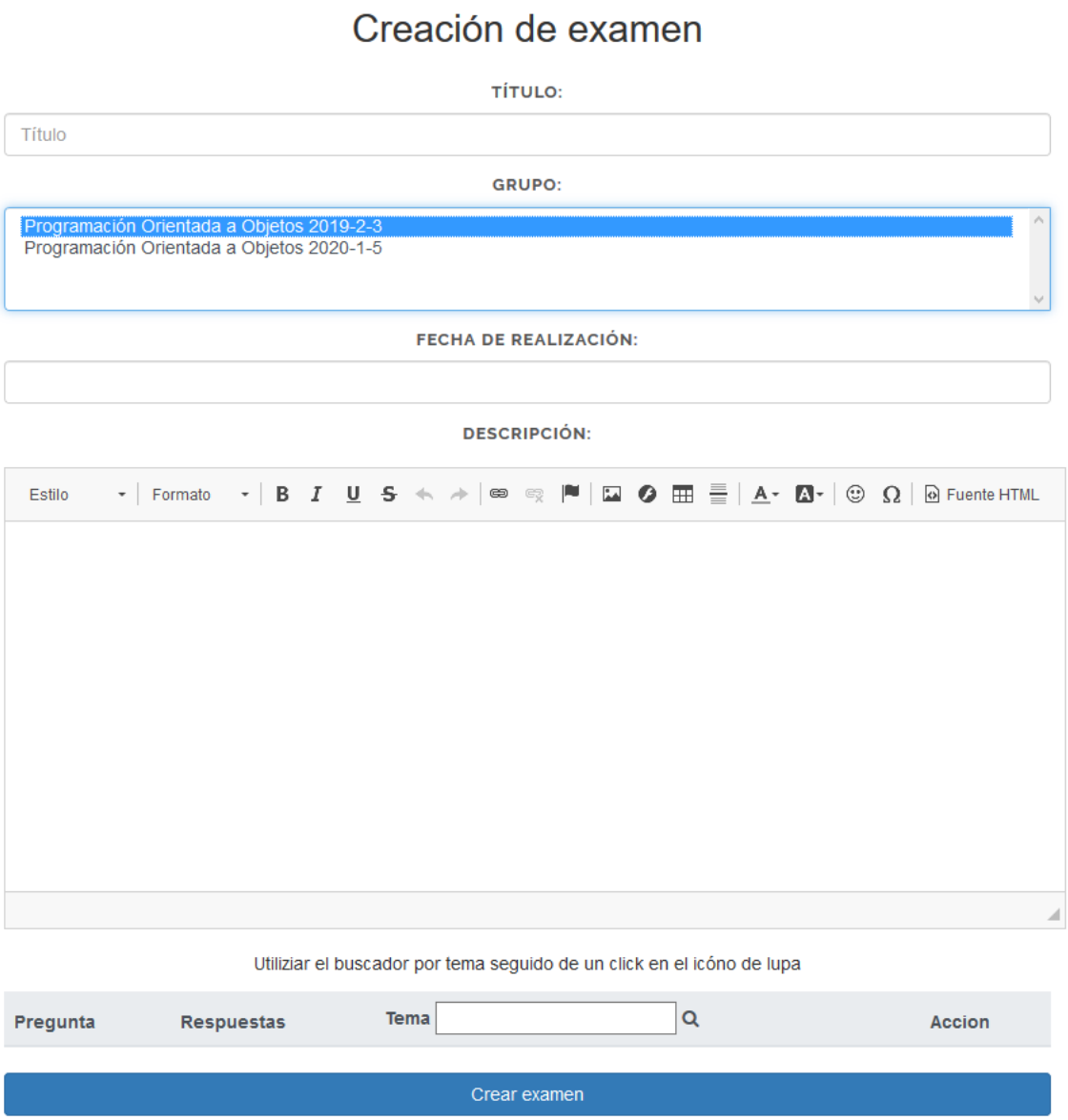


Imagen 6.21 Formulario de creación de examen.

Al realizar una búsqueda en la tabla de preguntas aparecerá el texto de la pregunta, sus opciones y la opción de añadir ó eliminar preguntas al examen.

Imagen 6.22: Tabla con resultados de búsqueda.

El examen debe llevar título, descripción, fecha límite de realización y preguntas. Una vez que haya pasado la fecha de realización, el examen no estará disponible para su aplicación.

Al dar clic en “Crear examen” se redireccionará a la página de “Mis grupos” y desde ahí se podrá revisar que se haya creado correctamente.

6.4.3 Carga de usuarios

Esta tarea está limitada al súper usuario, al entrar en esta página se muestra un formulario donde se puede seleccionar un archivo csv y llenar los datos del grupo que se va a crear, (materia, semestre y número de grupo).

Imagen 6.23: Formulario de creación de grupo.

El archivo csv debe contener las siguientes columnas en el orden mostrado:

Numero, Nombres, Apellido Paterno, Apellido Materno, Numero Cuenta/Empleado, Password, Rol

La Columna Numero solamente es un contador de registro que va de 1 a n dependiendo de la cantidad de usuarios. La columna Nombres, Apellido Paterno y Apellido Materno se utilizan para colocar la información personal del usuario. La columna Numero Cuenta/Empleado será el nombre de usuario que se usará para iniciar sesión en la plataforma. La columna Password es la contraseña del usuario, por defecto se recomienda que sea igual al nombre de usuario (es decir el número de cuenta ó de empleado), con la recomendación de que el usuario debe cambiar su contraseña desde la opción “Cambiar contraseña” de su perfil. Finalmente, la columna Rol funciona para indicar al sistema que tipo de usuario se va a registrar, es decir a que grupo pertenece (Alumno ó Profesor). A continuación se muestra un ejemplo de cómo debe estar estructurado el archivo csv:

Numero	Nombres	Apellido Paterno	Apellido Materno	Numero Cuenta/Empleado	Password	Rol
1	Sandra	App1	Apm1	40804888-1	40804888-1	Alumno
2	Pedro	App2	Apm2	40804888-2	40804888-2	Alumno
3	Sara	App3	Apm3	40804888-3	40804888-3	Alumno
4	Sarahí	App4	Apm4	40804888-4	40804888-4	Alumno
5	Laura	App5	Apm5	40804888-5	40804888-5	Alumno
6	Luis	App6	Apm6	40804888-6	40804888-6	Alumno
7	Saul	App7	Apm7	40804888-7	40804888-7	Alumno
8	Hector	App8	Apm8	40804888-8	40804888-8	Alumno
9	Tania	App9	Apm9	308021-85	308021-85	Profesor

Imagen 6.24: Ejemplo de csv válido.

Es muy importante que el nombre de la columna sea exactamente igual al mostrado, debido a que es el identificador que utiliza el sistema para poder realizar la carga masiva.

Una vez llenado el formulario se procede a dar clic en subir información, lo cual redireccionará al panel de administración, donde se indica el estatus de la carga de usuarios y se puede verificar mediante el panel de administración de contenidos siendo súper usuario en la sección “Usuarios”.

☐	40804888-1	40804888-1@cambiar.com	Sandra	App1App1	✓
☐	40804888-2	40804888-2@cambiar.com	Pedro	App2App2	✓
☐	40804888-3	40804888-3@cambiar.com	Sara	App3App3	✓
☐	40804888-4	40804888-4@cambiar.com	Sarahí	App4App4	✓
☐	40804888-5	40804888-5@cambiar.com	Laura	App5App5	✓
☐	40804888-6	40804888-6@cambiar.com	Luis	App6App6	✓
☐	40804888-7	40804888-7@cambiar.com	Saul	App7App7	✓
☐	40804888-8	40804888-8@cambiar.com	Hector	App8App8	✓

Imagen 6.25 Usarios cargados mediante csv.

7.- Manual técnico

Esta sección está dedicada a sentar las bases e indicar los puntos clave para el mantenimiento de la plataforma, además se describirá el patrón de diseño utilizado por Django para la creación de módulos.

7.1 Patrón Modelo-Vista-Template (MVT)

Este modelo es muy similar al modelo vista controlador (MVC), en este caso el template es el archivo html que se renderiza en el navegador web, la vista es un archivo python que recibe la petición HTTP, lo cual en el patrón de diseño Modelo Vista Controlador (MVC) sería lo equivalente al Controlador y el Modelo es la abstracción de la información de un objeto de base de datos mapeada en una clase de python.

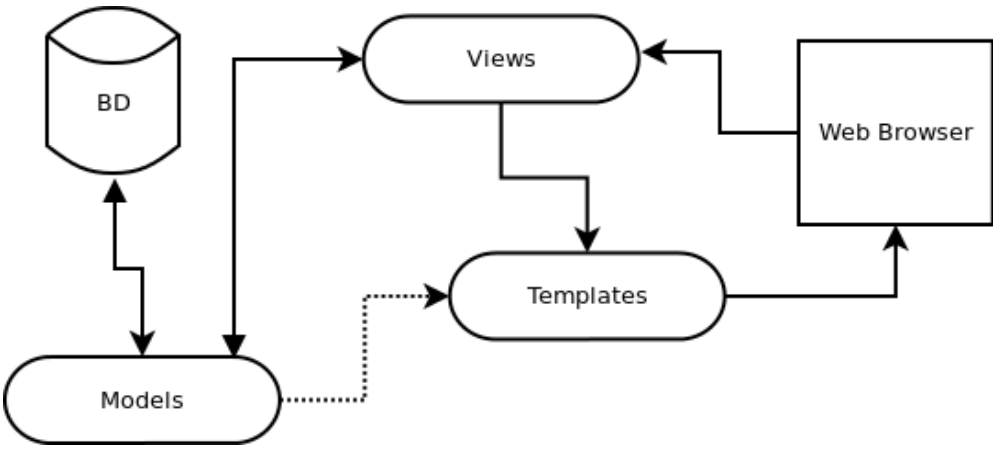


Imagen 7.1: Diagrama del MVT

7.2 Estructura de un proyecto en Django

Un proyecto en Django es un conjunto de aplicaciones (apps), cada app se puede visualizar como un módulo del sistema que se encarga únicamente de trabajos específicos, cada app tiene sus modelos y sus templates, lo cual permite la reutilización de código de manera sencilla, para la creación de una app se utiliza el archivo manage.py como se muestra a continuación:

python manage.py startapp {nombre_app}

Una vez creada la aplicación, se deberá instalar en el proyecto lo cual se realiza mediante el archivo settings.py que se encuentra en la carpeta principal del proyecto (en este caso poo) en la lista de INSTALLED_APPS:

```
INSTALLED_APPS = [
    'registration',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'ckeditor',
    'core',
    'groups.apps.GroupsConfig',
    'pages.apps.PagesConfig',
    'profiles',
    'practices',
    'homework.apps.HomeworkConfig',
    'exercises.apps.ExercisesConfig',
    'exams.apps.ExamsConfig',
    'administrator',
]
```

Imagen 7.2: Arreglo de apps del sistema.

Como se puede observar en la imagen 7.2, se pueden configurar ciertos parámetros de una app mediante un archivo de configuración (apps.py). En este archivo se configuran principalmente los nombres a mostrar en el panel de administración de contenidos de Django (/admin) mediante la propiedad “verbose_name”

```
class ExercisesConfig(AppConfig):
    name = 'exercises'
    verbose_name = "Gestor de ejercicios de practicas"
```

Imagen 7.2: Clase de configuración de una app.

7.2.1 Estructura de una App

La estructura de una app se muestra a continuación:

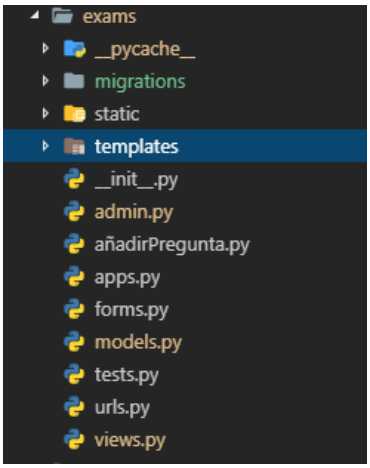


Imagen 7.3: Estructura de carpetas de una app.

Las carpetas static y templates no se crean de manera automática al crear una app, éstas deberán ser creadas manualmente. La carpeta static sirve para almacenar los ficheros estáticos como scripts de JavaScript u hojas de estilo css, mientras que la carpeta templates almacena los archivos html que se renderizarán al realizar una petición al servidor.

Al momento del despliegue de aplicaciones, Django ve una sola carpeta “static” y una sola carpeta “templates” para todas las aplicaciones, por tanto, se recomienda crear una subcarpeta con el nombre de la app en cada una de estas carpetas (static y templates), esto para evitar que haya conflictos con los nombres de otras aplicaciones a la hora de responder a la petición HTTP enviada

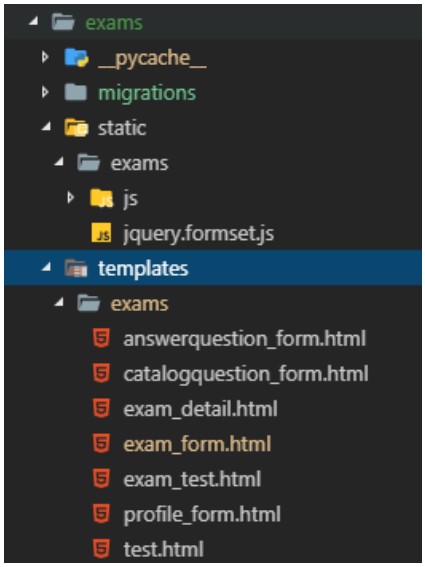


Imagen 7.4: Estructura interna de las carpetas static y templates.

7.2.1.1 Models

En el archivo models.py se deben registrar los modelos a utilizar. Este modelo se creará en la base de datos automáticamente al realizar las migraciones debidas. En esencia, estos modelos son el núcleo de la funcionalidad de la app, se definen mediante clases que contienen atributos y que pueden tener relaciones de uno a muchos ó de muchos a muchos con otros modelos, creando así una estructura relacional.

```
class Exam(models.Model):
    title= models.CharField(verbose_name="Título", max_length=200)
    #una examen puede ser de 1 o mas grupos y un grupo puede tener 1 o muchos examenes
    group=models.ManyToManyField(Group,verbose_name="Grupo", related_name="get_exams")
    delivery_date=models.DateTimeField(verbose_name="Fecha de Realización")
    description= RichTextField(verbose_name="Descripción")
    active= models.BooleanField(verbose_name="vigente",default=True)
    created = models.DateTimeField(auto_now_add=True, verbose_name="Fecha de creación")
    updated = models.DateTimeField(auto_now=True, verbose_name="Fecha de edición")

    class Meta:
        verbose_name="Examen"
        verbose_name_plural="Exámenes"
        #Considerar si es necesario el siguiente permiso
        permissions=(("can_see_homeworks","see_homeworks"),)

    @property
    def is_past_due(self):
        return datetime.today().replace(tzinfo=pytz.utc)>self.delivery_date.replace(tzinfo=pytz.utc)

    @property
    def delivered(self):
        return self.get_exam

    def __str__(self):
        return self.title
```

Imagen 7.4: Definición del modelo Exam.

En la clase mostrada en la imagen 7.4 se muestran los elementos más importantes al definir un modelo, los cuales se listan a continuación:

- La clase debe heredar de la clase models.Model.
- Debe contener atributos que son objetos de “models”.
- Cada atributo tiene opciones de personalización dependiendo el tipo de dato que vaya a manejar.
- Verbose_name es una propiedad que permite asignar un nombre público para el modelo, para que se pueda diferenciar de manera sencilla en el administrador de Django.

- Para las relaciones se puede añadir una propiedad “related_name” que permite acceder a los objetos relacionados desde el dueño de la relación, para este ejemplo un objeto ‘group’ tiene un atributo “get_exams” que regresa todos los objetos “Exam” relacionados a este grupo.
- Se debe definir qué tipo de relación tienen dos modelos, éstos pueden ser ForeignKey (uno a muchos), OneToOne (uno a uno), OneToMany (uno a muchos) ó ManyToMany (muchos a muchos).
- Se tiene una clase interna “Meta” donde se pueden definir nuevos permisos al modelo (además de los principales: add, change, view, delete) y asignar el nombre público del modelo.
- Se pueden añadir propiedades al modelo, estas propiedades permiten que los modelos tengan atributos extras que pueden ser utilizados en las vistas para tomar decisiones sobre qué elementos mostrar al usuario (para más detalles se puede consultar la sección 7.2.1.5 ‘templates’), estos métodos deben estar decorados con la etiqueta “property”.
- Se puede sobre escribir el método __str__ el cual funciona para identificar a un objeto del modelo, en este caso regresa un string equivalente al título del examen, en caso de no definirlo Django utiliza un nombre genérico del tipo “Object(n)”.

7.2.1.2 Views

Las vistas, como ya se mencionó, sirven como el punto de entrada a la petición HTTP y al resolver dicha petición manda una respuesta a la misma con el template adecuado.

Django tiene clases con patrones definidos para cada una de las vistas más comunes, estas clases se llaman vistas basadas en clases (cbv por su abreviatura en inglés), estas clases permiten un desarrollo ágil y sencillo para añadir funcionalidad al proyecto, en https://ccbv.co.uk/ se puede encontrar información acerca de todas las clases, métodos y atributos disponibles así como un enlace directo a la documentación oficial de cada una de ellas. Los métodos que se pueden sobre escribir para cada clase pueden variar dependiendo del tipo de vista que se quiera implementar, sin embargo, el método **get_context_data** está disponible en cada cbv ya que este método es el que se encarga de añadir toda la información disponible para renderizar en el template.

En la siguiente imagen se puede observar el uso de “DetailView”, esta clase permite acceder a los detalles del objeto de un modelo que se especifica con el atributo “model”, además se puede ver la sobre escritura del método get_context_data que realiza una consulta a los objetos del modelo DExam con la condición de que dicho objeto haya sido entregado por el usuario que está mandando la petición y el resultado lo almacena en el contexto de la respuesta HTTP con el nombre ‘delivered’ de manera que en el template se

podrá acceder a dicho valor para mostrar un mensaje avisando que el examen ya ha sido realizado.

```python
class ExamDetailView(DetailView):
    model = Exam

    def get_context_data(self, **kwargs):
        context = {}
        if self.object:
            context['object'] = self.object
            context_object_name = self.get_context_object_name(self.object)
            if context_object_name:
                context[context_object_name] = self.object
        try:
            #Check if this exam has been realized by this user
            realized=DExam.objects.filter(exam=context['object'],user=self.request.user)
            if realized:
                context['delivered'] = True
            else:
                context['delivered'] = False
        except:
            pass
        context.update(kwargs)
        return super().get_context_data(**context)
```

Imagen 7.5: Clase que define una vista basada en un modelo Exam.

Para identificar exactamente que objeto del modelo “Exam” se está manipulando en la vista se utiliza una URI (identificador de recursos uniforme) definida en el archivo urls.py.

Otra forma de crear vistas es a partir de métodos, estas vistas no están asociadas a ningún modelo por lo cual son convenientes cuando no se requiere manipular un modelo aunque también son convenientes cuando se busca manipular más de un modelo a la vez. La imagen 7.6 muestra una vista que atiende la petición que se envía al terminar la realización de un examen, en esta situación utilizar una cbv complica el proceso debido a que clase de la vista donde se realiza la petición no tiene un método que acepte peticiones post, por lo tanto, dicha petición se envía a este método que recibe la información enviada, la procesa y responde con una redirección.

```python
def processExam(request,exam_id):
    user= request.user
    userExam= DExam(user=user,exam=Exam.objects.get(id=exam_id))
    total=0
    correct=0
    #Key is question and value is answer
    for key,value in request.POST.items():
        if(key!='csrfmiddlewaretoken'):
            total=total+1
            userAnswer=AnswerQuestion.objects.get(id=value)
            DExamUserAnswers(dexam=userExam,question=CatalogQuestion.objects.get(id=key),answer=userAnswer)
            if userAnswer.correct:
                correct = correct +1
    note= (correct/total)*10
    userExam.note=note
    userExam.save()
    return HttpResponseRedirect(reverse('groups:list'))
```

Imagen 7.6: Metodo que procesa una petición y responde una vista.

7.2.1.3 Archivo urls.py

En el archivo urls.py se definen los puntos de entrada de las peticiones HTTP y se asocian a una vista, cuando se requiere manejar un objeto en específico dicha URL deberá estar asociada a dicho objeto, esto se logra utilizando el identificador único del objeto (pk), es decir se utiliza una URI.

```python
exams_patterns = ([
    path('new/exam', ExamCreate.as_view(), name='newExam'),
    path('new/exam/table',questions_table,name="tabla"),
    path('<int:pk>/<slug:slug>',ExamDetailView.as_view(),name="exam"),
    path('process/<int:exam_id>/',processExam,name="process"),
    ]
,'exams')
```

Imagen 7.7: Configuración de rutas de la aplicación.

Como se observa en la imagen 7.7, las urls se manejan a través de túplas, donde el primer elemento es la lista de direcciones disponibles, cada una está asociada a una vista basada en clases ó a un método que recibe la petición, además tienen un nombre que permite identificar dicha dirección; el segundo elemento de la tupla permite tener un control sobre las direcciones accesibles, de manera que no se puede repetir ninguna dirección debido a que se utiliza el identificador de la app, este identificador permite hacer referencias a las direcciones de manera simple utilizando el siguiente patrón: {identifidador:name [argumentos]}, en la sección de templates se ve más a fondo el uso de estos patrones.

Además se debe de registrar esta tupla en el archivo urls.py del proyecto, dicho archivo se encuentra en la carpeta principal del proyecto (poo) como se muestra en la imagen 7.8.

```
urlpatterns = [
    path('',include('core.urls')),
    path('pages/',include(pages_patterns)),
    path('admin/', admin.site.urls),
    #Paths de auth
    path('accounts/',include('django.contrib.auth.urls')),
    path('accounts/',include('registration.urls')),
    path('profiles/',include(profiles_patterns)),
    #Paths de Groups
    path('groups/',include(groups_patterns)),
    #Paths de HomeWorks
    path('homework/',include(homeworks_patterns)),
    #Paths de HomeWorks
    path('practices/',include(practices_patterns)),
    #Paths de Exercises
    path('test/',include(exercises_patterns)),
    #Paths de Exams
    path('exams/',include(exams_patterns)),
    #Paths de administrator
    path('administrator/',include(administrator_patterns)),
]
```

Imagen 7.8: Configuracion de rutas globales de la plataforma

Como puede observarse, el registro de direcciones se hace mediante una tupla ‘path’, el primer elemento hace referencia al contexto en donde se están registrando las direcciones de las app, por ejemplo, para la app ‘exams’ las direcciones están registradas en el contexto ‘exams/’ de manera que para acceder a la dirección ‘newExam’ registrada en urls.py de la app se deberá ingresar mediante la url completa: localhost:8000/exams/new/exam.

7.2.1.4 Archivo forms.py

El archivo forms.py no es creado de manera automática al iniciar una nueva app, sin embargo, es un archivo que permite personalizar los formularios de las vistas basadas en clases que utilicen dicho formulario (create/update).

El uso de estos formularios es sencillo, se crea una clase que herede de ModelForm y se asigna el modelo en el cual está basado dicho formulario, esto permite modificar la estética del formulario mediante la propiedad ‘attrs’ que es la manera en la que Django asigna propiedades html a elementos de un formulario.

```python
class ExamForm(forms.ModelForm):
    class Meta:
        model= Exam
        #Exclusion of active so it doesn't appear at view
        exclude = ('active',)
        widgets={
            'delivery_date':forms.DateTimeInput(attrs={'class':'datetime-input form-control'}),
            'description':forms.Textarea(attrs={'class':'form-control'}),
            'title':forms.TextInput(attrs={'class':'form-control', 'placeholder':'Título'}),
            'group': forms.SelectMultiple(attrs={'class':'form-control'}),
        }
    def __init__(self, *args, **kwargs):
        #Get the info about the 'profesor' so groups can be filtered
        profesor = kwargs.pop('profesor')
        super(ExamForm, self).__init__(*args, **kwargs)
        self.fields['group'].queryset = Group.objects.filter(user=profesor,active=True)
```

Imagen 7.9: Clase que define un formulario basado en el modelo Exam.

Para usar este formulario personalizado se debe especificar en la vista mediante el atributo form_class como se muestra a continuación:

```python
class ExamCreate(CreateView):
    form_class=ExamForm
    template_name='exams/exam_form.html'
    success_url= reverse_lazy('groups:list')

    def get_form_kwargs(self):
        kwargs = super(ExamCreate, self).get_form_kwargs()
        #add profesor value to current user so the form can filter groups
        kwargs.update({'profesor': self.request.user})
        return kwargs
```

Imagen 7.10: Clase que define una vista a partir de un formulario.

Además se puede sobre escribir el metodo __init__ para limitar los elementos relacionados al formulario en caso de requerirlo. En las imágenes 7.9 y 7.10 se puede observar como se limitan los objetos para el campo ‘groups’, para esto se asignó un valor al contexto del formulario mediante el método get_form_kwargs en la vista, esto permite que al momento de inicializar dicho formulario se pueda utilizar el diccionario kwargs y extraer el valor de ‘profesor’ para así utilizarlo en la consulta y sólo permitir el manejo de ciertos datos.

7.2.1.5 Templates

El sistema de templates de Django permite utilizar etiquetas llamadas template tags que facilitan la creación de los elementos visuales en la plataforma, en esta sección se explican tags utilizados en la plataforma.

El proyecto tiene una app llamada core, la cual funciona como núcleo para todas las paginas mostradas en la aplicación, esto es debido a que se tiene un sistema de herencia que permite reutilizar código html ya realizado lo cual crea una experiencia de seguir navegando en la misma página, esto se logra mediante un sistema de etiquetas de Django.

Etiqueta block

Esta etiqueta permite crear un espacio en el cual se insertará código html mediante otro template, su utilización requiere iniciar con {% block {block_id}%} y terminar el bloque con {% endblock %} el espacio para block_id puede ser definido por el programador.

La plataforma tiene como template principal el archivo 'base.html' de la app core, en este archivo se puede visualizar la siguiente línea de código.

```
<title>{% block title %}{% endblock %}</title>
```

Imagen 7.11: Uso de la etiqueta 'title'

Este tag permite que una página que herede de este template pueda colocar cualquier bloque de código dentro de estos tags, permitiendo así crear una experiencia dinámica en la navegación donde se pueden cambiar algunos detalles dependiendo de la página que se esté visitando, para este caso se modifica la información del título de la página.

Etiqueta extends

La etiqueta extends permite colocar bloques de código en los bloques heredados y que se cargue dicho código en el template principal, siguiendo el ejemplo del bloque anterior se tiene en el archivo exam_form.html el siguiente bloque de código:

```
{% extends 'core/base.html' %}
{% load static %}
{%block title%}Creación de examen{%endblock%}
{% block content %}
```

Imagen 7.12: Uso de la etiqueta extends.

Se coloca en el inicio del archivo la etiqueta extends lo cual permite heredar todos los bloques creados en el template 'base.htm' lo cual permite añadir información dentro del bloque y se tenga una navegación fluida entre páginas.

Etiqueta load

Esta etiqueta permite cargar ciertos recursos de la plataforma, en este caso se le indica que cargue los elementos estáticos de la plataforma, esto se requiere para poder utilizar la etiqueta static que permite localizar los archivos estáticos sin necesidad de escribir la ruta completa.

```
{% load static %}
```

Imagen 7.13: Uso de la etiqueta load.

Etiqueta static

Esta etiqueta permite acceder a la carpeta static configurada en poo/settings.py, facilitando así el manejo de referencias a archivos estáticos utilizados en la plataforma ya que sólo se debe hacer referencia a la app y a los archivos dentro de la carpeta static.

```
<link rel="stylesheet" href="{% static 'core/css/style.css' %}" type="text/css" />
<link rel="stylesheet" href="{% static 'core/css/dark.css' %}" type="text/css" />
<link rel="stylesheet" href="{% static 'core/css/font-icons.css' %}" type="text/css" />
<link rel="stylesheet" href="{% static 'core/css/animate.css' %}" type="text/css" />
<link rel="stylesheet" href="{% static 'core/css/magnific-popup.css' %}" type="text/css" />
<link rel="stylesheet" href="{% static 'core/css/colors.css' %}" type="text/css" />
<link rel="stylesheet" href="{% static 'core/css/responsive.css' %}" type="text/css" />
```

Imagen 7.14: Uso de la etiqueta static.

Etiqueta if - else

Esta etiqueta permite tomar decisiones al momento de renderizar código html, utiliza un tag de apertura y un tag de cierre. La evaluación de la condición dentro del tag puede llegar a ser muy compleja, se recomienda que en casos donde la evaluación requiera muchas condiciones se diseñe una propiedad en el modelo (visto anteriormente en la sección de Modelos) para vistas basadas en clases ó se añada una variable al contexto sobre escribiendo el método get_context_data debido a que no es recomendable realizar mucho procesamiento con las etiquetas.

En la imagen 7.15 se verifica que el usuario esté autenticado en la plataforma y, dependiendo del resultado, se muestran ciertas opciones en la barra de navegación.

```
{% if not request.user.is_authenticated %}
<li class="nav-item px-lg-4"><a href="{% url 'login' %}"><div>Acceder</div></a></li>
{% else %}
<li class="nav-item px-lg-4"><a href="{% url 'pages:pages' %}"><div>Articulos</div></a></li>
<li class="nav-item px-lg-4"><a href="{% url 'profile'%}"><div>Perfil</div></a></li>
{% if request.user.is_staff %}
<li class="nav-item px-lg-4" ><a href="{% url 'administrator:index' %}"><div>Panel de administración</div></a></li>
{%endif%}
<li class="nav-item px-lg-4"><a href="{% url 'logout' %}"><div>Salir</div></a></li>
{% endif %}
```

Imagen 7.15: Uso de la etiqueta if-else.

Etiqueta for

Esta etiqueta permite iterar sobre una lista de objetos que se encuentre en el contexto actual de la vista, requiere una etiqueta de inicio y una de cierre, por ejemplo en el archivo group_list.html de la app groups se encuentra el siguiente código:

```
{% for group in request.user.get_groups.all %}
    <li><a href="{% url 'groups:detail' group.id group.subject.name|slugify %}"> {{group.subject.name}} - {{ group.semester }}</a></li>
{% empty %}
    <li>No estás inscrito en ningún grupo</li>
{% endfor %}
```

Imagen 7.16: Uso de la etiqueta for.

Este bloque de código itera por cada elemento dentro de los grupos del usuario autenticado y va mostrando a manera de lista cada uno de ellos, en caso de que la lista esté vacía muestra un mensaje indicando que no se está inscrito a ningún grupo.

Etiqueta url

Esta etiqueta permite hacer referencia a una dirección dentro de la plataforma utilizando su identificador y el nombre de la dirección.

```
<li class="nav-item px-lg-4"><a href="{% url 'pages:pages' %}"><div>Articulos</div></a></li>
```

Imagen 7.17: Uso de la etiqueta url.

El enlace dirige a pages:pages, siendo la primera parte del patrón la referencia a la app y la segunda parte el nombre de la dirección, si se observa el archivo urls.py de la app pages se tiene lo siguiente:

```
pages_patterns = ([
    path('', PagesListView.as_view(), name='pages'),
    path('<int:pk>/<slug:slug>/', PageDetailView .as_view(), name='page'),
    path('create/', PageCreateView.as_view(), name='create'),
    ]
,'pages')
```

Imagen 7.18: Configuraciones de rutas de la app.

Es decir, la etiqueta {% url ‘pages:pages’ %} está haciendo referencia a la vista basada en clases ‘PagesListView’

```
path('pages/',include(pages_patterns)),
```

Imagen 7.19: Configuración de rutas global para la app de Paginas.

Observando que el path de la app ‘pages’ en urls.py del proyecto (poo/urls.py) se encuentra en ‘pages/’ y que la vista ‘PagesListView’ tiene el path asociado a la raíz de su aplicación (‘’) la dirección localhost:8000/pages/ mostrará la vista basada en clases ‘PagesListView’

Imagen 7.20: Página que muestra la vista asociada a la ruta ‘/pages’.

En el ejemplo de la etiqueta for se realizó una lista de los grupos a los cuales el usuario autenticado estaba inscrito, cada elemento de esta lista tiene un enlace a los detalles de dicho grupo, como ya se mencionó antes, esto se logra a través de una URI, es decir, una dirección asociada a un elemento único, la creación de este enlace puede verse en la siguiente imagen:

```
<li><a href="{% url 'groups:detail' group.id group.subject.name|slugify %}">
```

Imagen 7.21: Creación de un enlace a un elemento con identificador unico.

Como se puede observar en la Imagen 7.21, se está utilizando una etiqueta url al igual que antes, sin embargo, esta vez se están pasando dos argumentos necesarios, el id del grupo y el nombre de la materia, éste último al poder contener espacios en blanco no permitidos en las direcciones web registradas en urls.py, requiere pasar por un proceso llamado ‘slugify’ que en esencia intercambia los espacios en blanco por guiones (-), el orden en los que son pasados los argumentos debe coincidir con el registrado en urls.py.

```
groups_patterns = ([
    path('', GroupListView.as_view(), name='list'),
    path('<int:pk>/<slug:slug>/', GroupDetailView.as_view(), name='detail'),
], "groups")
```

Imagen 7.22: Configuración de rutas con identificador único.

Por tanto, un ejemplo de una dirección válida sería: 127.0.0.1:8000/groups/4/programacion-orientada-a-objetos/. La cual regresaría la vista basada en clases ‘GroupDetailView’

Imagen 7.23: Vista de un elemento con identificador único.

Mostrar información de un objeto

En la sección de la etiqueta for también se mostró que se puede visualizar la información de un objeto que exista dentro del contexto de la vista, para esto no se requiere una etiqueta en específico, solamente se requiere colocar la información que se desea mostrar dentro de doble llave como se visualiza en la imagen 7.24, al mostrar el nombre y el semestre del grupo.

```
<li>
    <a href="{% url 'groups:detail' group.id group.subject.name|slugify %}">
        {{group.subject.name}} - {{ group.semester }}
    </a>
</li>
```

Imagen 7.24: Visualización de información del objeto.

Propiedades del modelo

Como ya se ha mencionado, se pueden definir propiedades en el modelo que pueden utilizarse en los templates, un ejemplo de esto se puede ver en exam_detail.html.

```
{% if exam.is_past_due %}
  La fecha de realización del  examen ya pasó
{%else%}
```

Imagen 7.25: Uso de la propiedad personalizada ‘is_past_due’ del modelo

Hay que recordar que la propiedad “is_past_due” se definió a nivel de modelo

```
@property
def is_past_due(self):
    return datetime.today().replace(tzinfo=pytz.utc)>self.delivery_date.replace(tzinfo=pytz.utc)
```

Imagen7.26: Uso de @property para definir propiedades.

Por lo tanto, al utilizar “exam.is_past_due” en el template se obtiene el valor regresado por la función, en este caso como es una comparación regresa un valor booleano utilizado para determinar si se muestra ó no un mensaje en la vista.

7.2.1.6 Archivo admin.py

El archivo admin.py permite registrar qué modelos van a poder ser manipulados desde el panel de administración de Django y permite modificar los objetos a los que puede acceder cada usuario.

Para registrar un modelo se requiere crear una clase que herede de ModelAdmin y que contenga un atributo list_display que es una lista de los atributos del modelo que serán mostrados en la tabla principal del panel de administración.

```python
class ExamAdmin(admin.ModelAdmin):
    list_display=['title','created']
    inlines=[
        ExamQuestionInLine
    ]
    def get_queryset(self, request):
        qs = super().get_queryset(request)
        if request.user.is_superuser:
            return qs
        return qs.filter(group__in=request.user.get_groups.all())

    def formfield_for_manytomany(self, db_field, request, **kwargs):
        if db_field.name == "group":
            if request.user.is_superuser:
                return super().formfield_for_manytomany(db_field, request, **kwargs)
            else:
                kwargs["queryset"] = Group.objects.filter(user=request.user)
                return super().formfield_for_manytomany(db_field, request, **kwargs)
```

Imagen 7.27: Configuración del panel de administración para el modelo Exam.

En la imagen 7.27 se observa la propiedad “inlines”. Esta propiedad permite manipular las relaciones entre objetos desde la administración de la clase padre de la relación. Para la imagen anterior se tiene una relación entre ExamQuestion y Exam, es decir, la llave foránea está en el modelo ExamQuestion, sin embargo, se quiere agregar objetos de ExamQuestion al momento de estar administrando un Objeto Exam por lo cual se debe crear la clase ExamQuestionInLine mostrada a continuación.

```python
class ExamQuestionInLine(admin.TabularInline):
    model=ExamQuestion
```

Imagen 7.28: Clase para editar relaciones dentro de la consola de administración.

Esto permite que en el panel de administración la edición de un objeto exam pueda manipular a su vez los objetos relacionados, un ejemplo de esto se puede ver en la imagen 7.29 y 7.30.

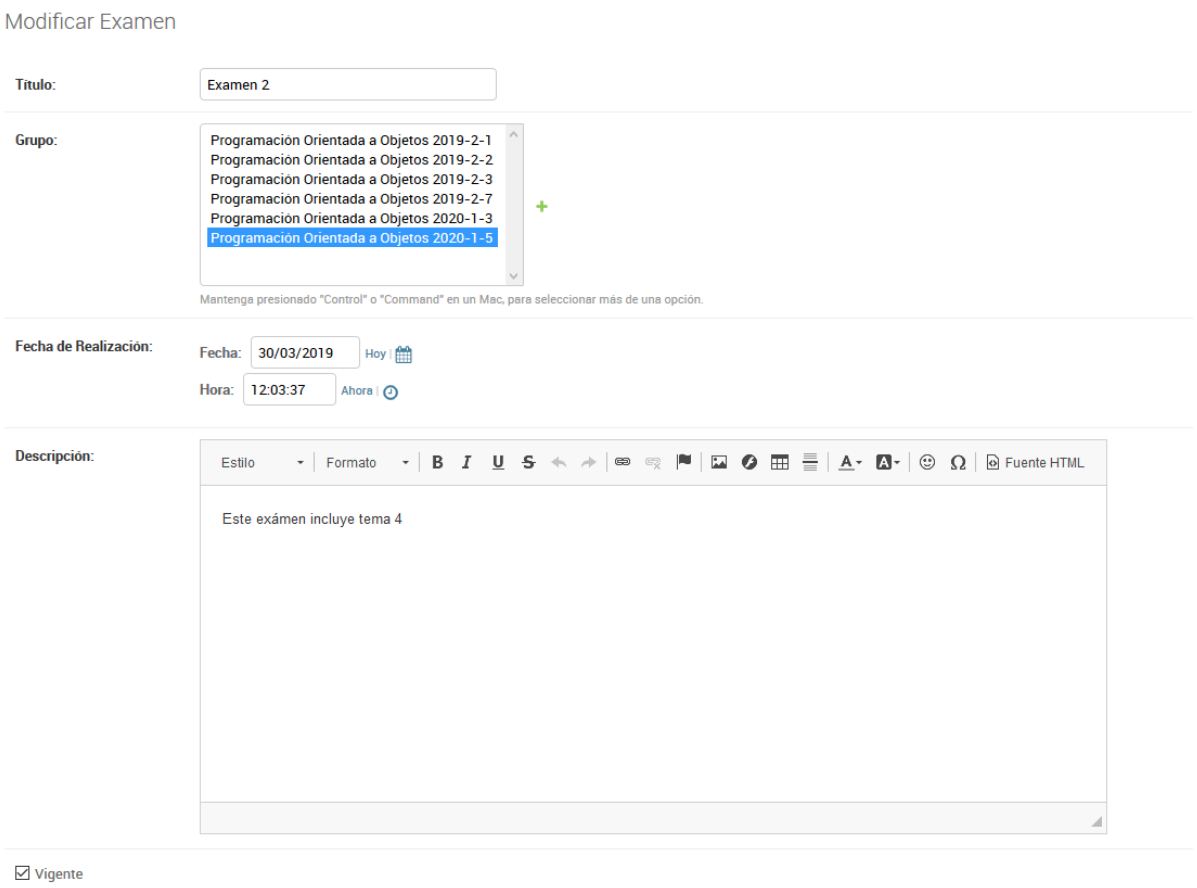


Imagen 7.29: Edición de examen.

Imagen 7.30: Formulario dentro de edición de examen dedicado a la relación las preguntas de catálogo.

En la misma clase se sobre escriben los métodos get_queryset y formfield_for_manytomany.

- get_queryset: este método es el encargado de consultar a qué objetos tendrá acceso el usuario autenticado, como puede verse en la imagen, en caso de que el usuario sea un súper usuario regresa todos los objetos del modelo Exam, en caso contrario se tiene la restricción de obtener sólo los exámenes que estén relacionados a los grupos en los cuales pertenece el usuario.

```
def get_queryset(self, request):
    qs = super().get_queryset(request)
    if request.user.is_superuser:
        return qs
    return qs.filter(group__in=request.user.get_groups.all())
```

Imagen 7.31: Método que permite filtrar los objetos en los páneles de administración

- formfield_for_manytomany: este método es el encargado de limitar qué objetos aparecerán en los campos de selección del formulario de edición que tengan una relación muchos a muchos. Por ejemplo, los modelos Group y Exam tienen este tipo de relación, por lo cual, este método es ejecutado para consultar los objetos que se deben mostrar y si el campo es ‘group’ se hace una validación similar que con el método get_queryset, para el súper usuario se obtienen todos los grupos relacionados mientras que para el resto sólo se muestran los grupos donde el usuario autenticado pertenezca.

```
def formfield_for_manytomany(self, db_field, request, **kwargs):
    if db_field.name == "group":
        if request.user.is_superuser:
            return super().formfield_for_manytomany(db_field, request, **kwargs)
        else:
            kwargs["queryset"] = Group.objects.filter(user=request.user)
            return super().formfield_for_manytomany(db_field, request, **kwargs)
```

Imagen 7.32: Método que permite filtrar los objetos dentro de un campo de selección ‘muchos a muchos’ en la consola de administración.

De la misma manera que existe este método también existen sus equivalente para las demás relaciones, por ejemplo para una relación con un campo ForeignKey.

```
def formfield_for_foreignkey(self, db_field, request, **kwargs):
    if db_field.name == "group":
        if request.user.is_superuser:
            return super().formfield_for_foreignkey(db_field, request, **kwargs)
        else:
            kwargs["queryset"] = Group.objects.filter(user=request.user)
            return super().formfield_for_foreignkey(db_field, request, **kwargs)
```

Imagen 7.33: Método que permite filtrar los objetos dentro de un campo de selección ‘1 a muchos’ en la consola de administración.

No basta con la creación de las clases administrativas, si se dessea que aparezcan en el panel de administración de Django se deben registrar en éste como se muestra al final del archivo admin.py

```
admin.site.register(NotePercentage,NotePercentagesAdmin)
admin.site.register(SubjectCatalog ,SubjectAdmin)
admin.site.register(Group,GroupAdmin)
```

Imagen 7.34: Registro de clases administrativas en la consola de administración.

7.3 Diagramas entidad relación

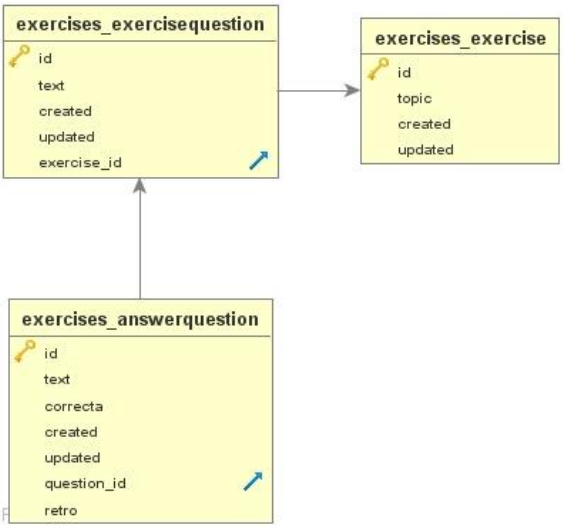


Imagen 7.35: Diagrama entidad relación de la app exercises

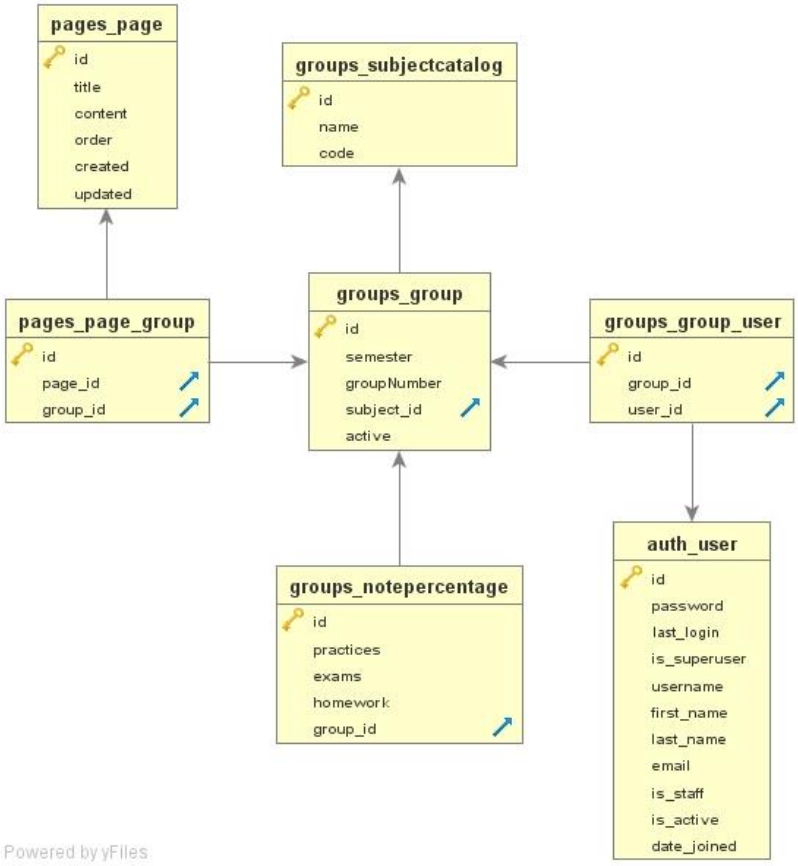


Imagen 7.36: Diagrama entidad relación de la app groups y pages

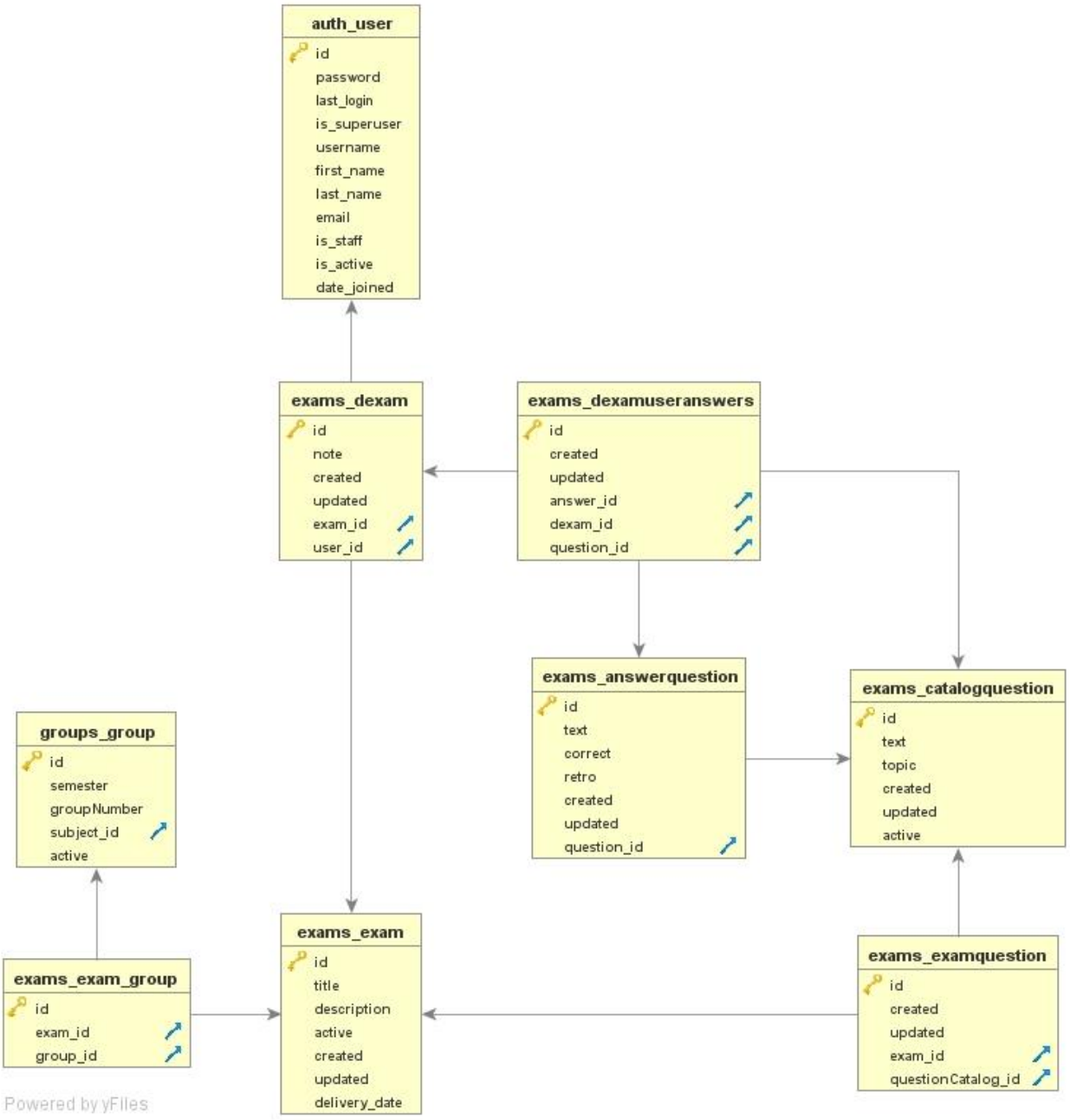


Imagen 7.37:Diagrama entidad relación de la app exams

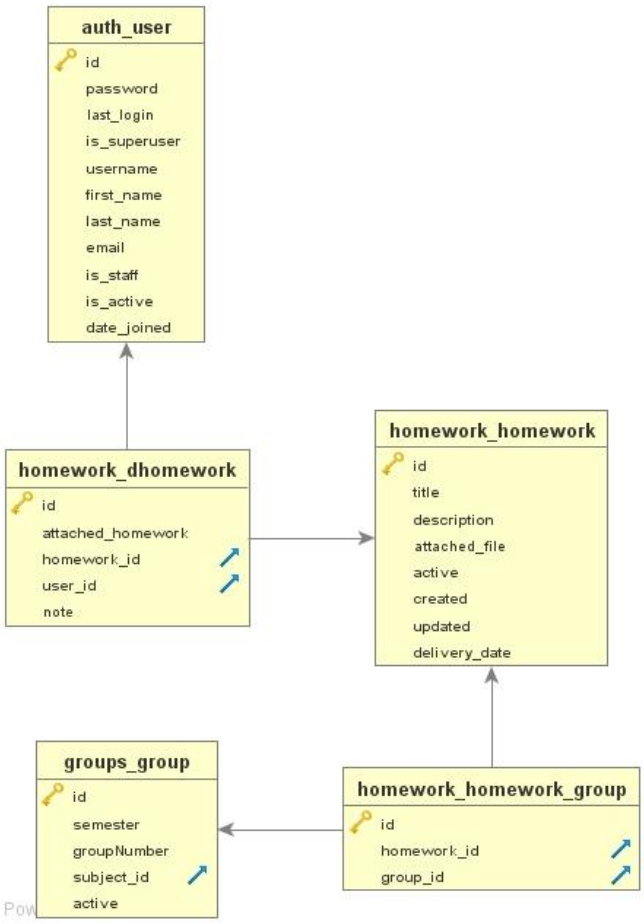


Imagen 7.38 : Diagrama entidad relación de la app homework

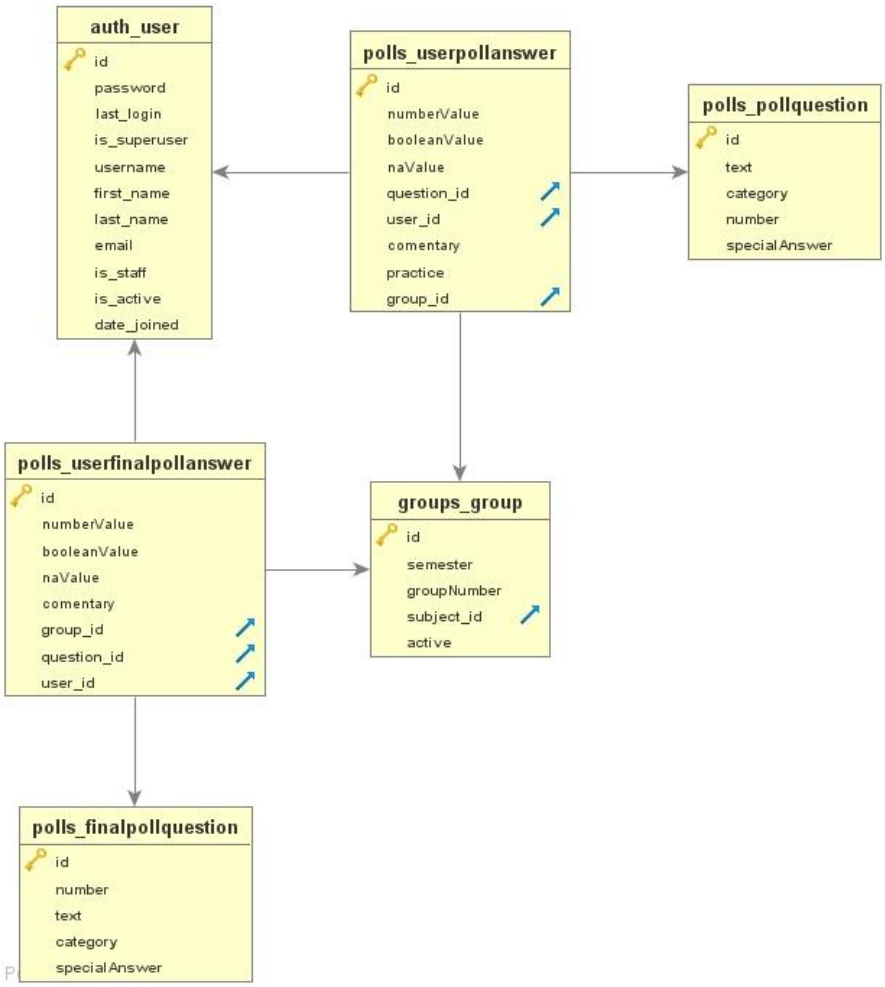


Imagen 7.39 : Diagrama entidad relación de la app polls

7.4 Diagramas de casos de uso

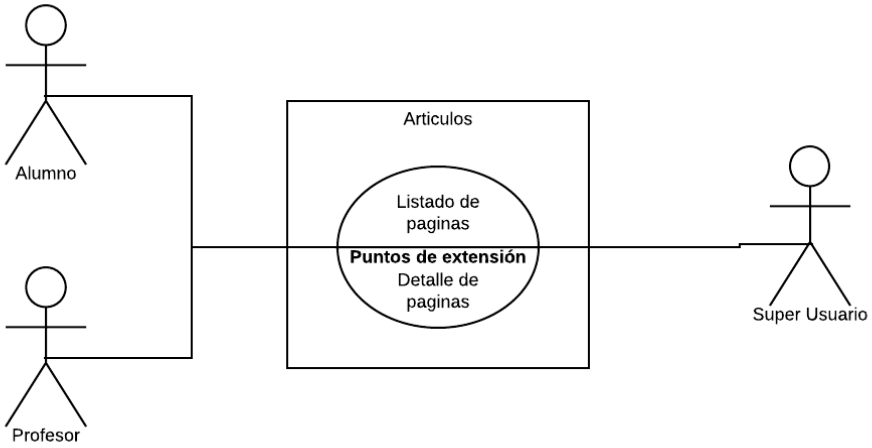


Imagen 7.35: Diagrama de casos de uso de Artículos

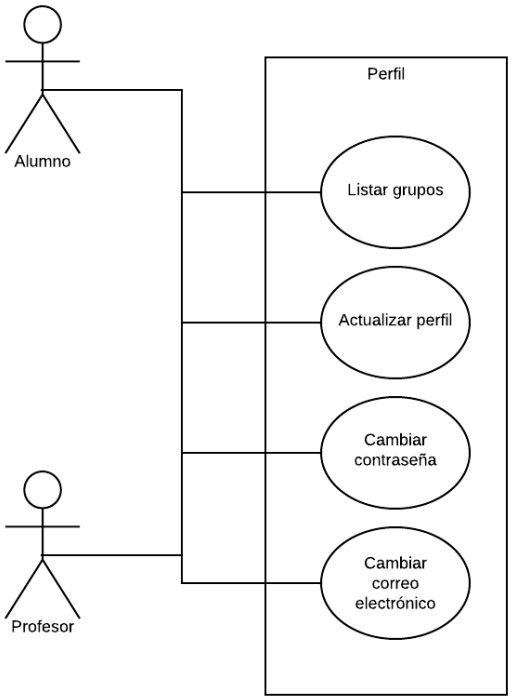


Imagen 7.36: Diagrama de casos de uso de Perfil

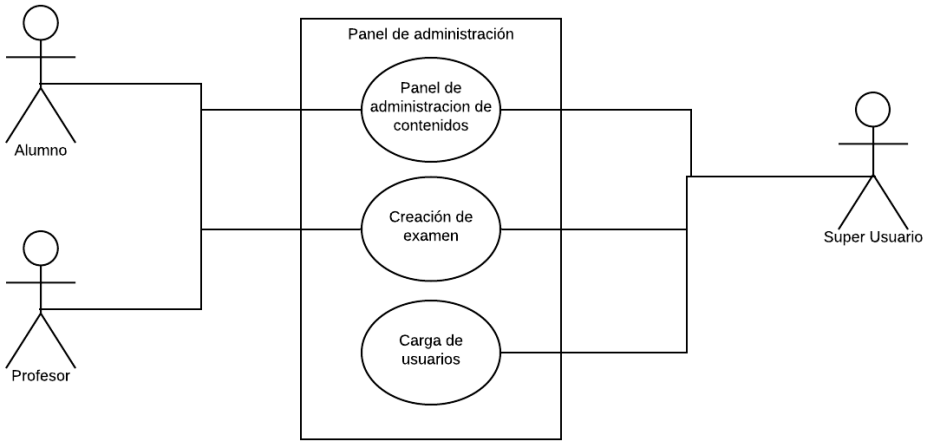


Imagen 7.37: Diagrama de casos de uso de panel de administración.

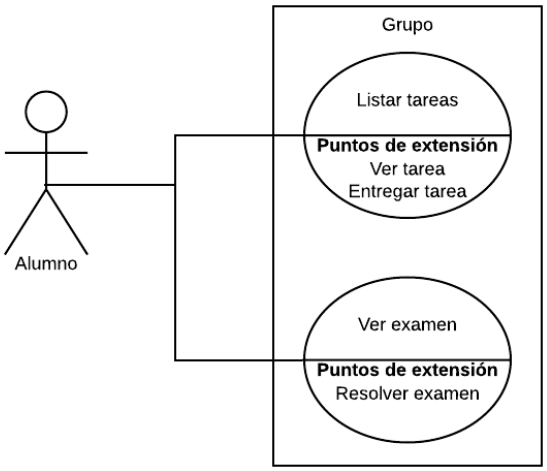


Imagen 7.38: Diagrama de casos de uso de Grupo.

7.5 Diagrama de componentes

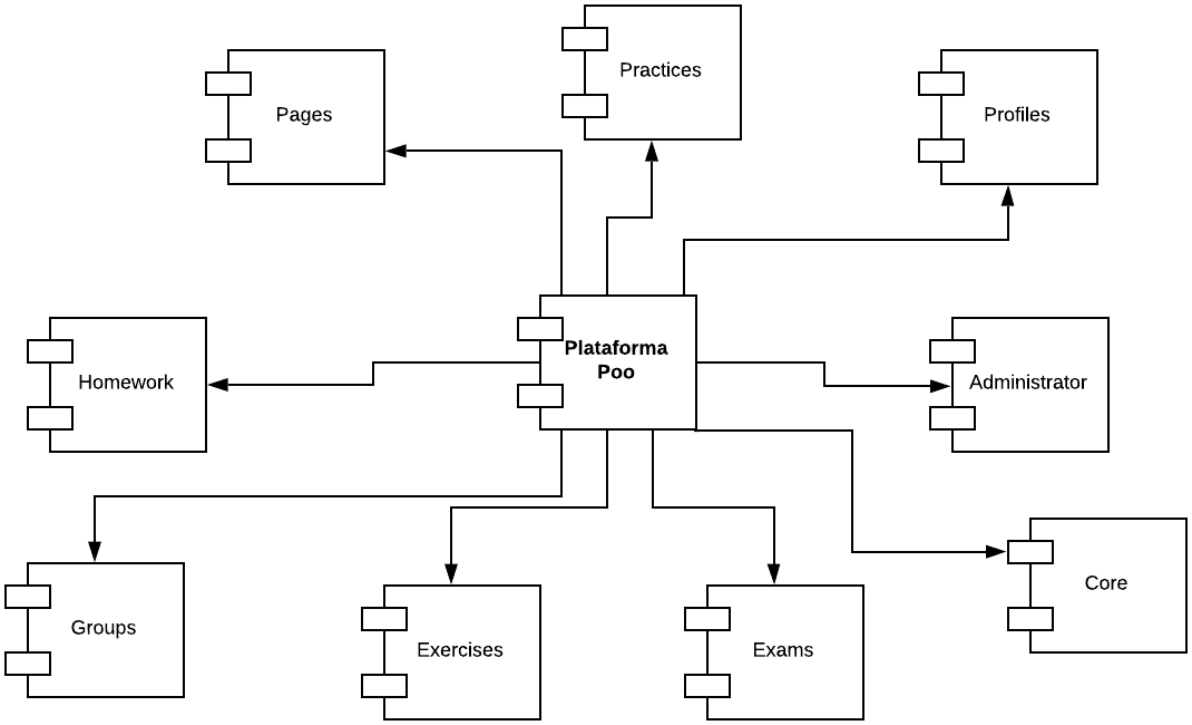


Imagen 7.39: Diagrama de componentes de la plataforma.

8.- Resultados esperados

Se espera que la plataforma sirva como una herramienta en línea para que el alumno no tenga impedimentos al momento de aprender temas de la asignatura de Programación orientada a objetos y que los conocimientos adquiridos le sean útiles en su carrera ya que los ejercicios de las prácticas están basados en situaciones semejantes a las que se pueden enfrentar en un ambiente laboral. Además, se espera que se mejore el cumplimiento del Sistema de Gestión de Calidad ya que los alumnos no se ven limitados a la presencia de un profesor para seguir realizando actividades académicas como lo son las prácticas y, por lo tanto, su aprendizaje no sea afectado por circunstancias ajenas él.

Finalmente, se plantea que el material de apoyo generado funcione como un punto de retroalimentación constante entre académicos y alumnos y esto permita que se aumente la calidad del material entregado ó material desarrollado en un futuro dentro ó fuera de la plataforma entregada.