UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Plataforma para gestión de laboratorios de telecomunicaciones con PHP, MySQL y Azure

TESINA

Que para obtener el título de

Ingeniero en Telecomunicaciones

P R E S E N T A

José Ángel De La Cruz Sánchez

DIRECTORA DE TESINA

M.I. Elizabeth Fonseca Chávez

Ciudad Universitaria, Cd. Mx., 2022

Índice

1. Programa

Nombre del programa: *Procesamiento digital de señales y multimedia*

2. Objetivos

Realizar una plataforma de comunicación entre alumnos, profesores y administrativos, para mejorar la administración de calificaciones del laboratorio de alta frecuencia en la carrera de Ingeniería en Telecomunicaciones brindando un servicio eficaz, rápido y permanente.

- Implementar una base de datos relacional para profesores/alumnos cumpliendo con las reglas de normalización para su diseño y almacenarla en un proveedor de servidores web.
- Desarrollar el diseño de la plataforma para su visualización en computadoras de escritorio, además de la parte responsiva para dispositivos móviles.

3. Marco Teórico

El departamento de telecomunicaciones de la Facultad de Ingeniería debe ofrecer a sus estudiantes y académicos una forma eficaz y cómoda para la administración de los laboratorios de alta frecuencia, brindando un servicio de registro de estudiantes y profesores para la evaluación de los laboratorios inscritos por semestre, por lo que se requiere un diseño completo de tipo *front-end* y *back-end* utilizando las últimas tecnologías de desarrollo de aplicaciones web y almacenamiento de bases de datos en un servidor virtual.

Existen dos partes en las que un desarrollador de aplicaciones sea de escritorio, páginas web o móviles dedica gran parte del tiempo, una parte es llamada como *Frontend* y la otra *Backend*. La diferencia radica en que la primera está orientada a la interacción de los usuarios a través de un cliente, que en este caso es su navegador. Todas las imágenes, vídeos, colores, menús, enlaces, etc. son parte exclusiva de la estructura *Frontend*. Considerando la parte de desarrollo, existen algunos lenguajes en los cuales se puede ir diseñando esta cara de la aplicación, uno es conocido como el **Lenguaje de Marcas de Hipertexto** (del inglés *HyperText Markup Language*), en resumidas cuentas, **HTML**, aunque dentro de la comunidad no lo consideran un lenguaje de programación, es más bien un lenguaje de etiquetas que nos ayuda a darle estructura al diseño de la página en este caso de tipo web.

Siguiendo con la estructura de *Frontend*, hoy en día el 90 % de las páginas son diseñadas para que el usuario se sienta cómodo al navegar por ella, no solo que visualice textos planos, formularios, imágenes en primer plano, sino también, que pueda tener un diseño visual totalmente amigable para el usuario, por ello se utiliza un lenguaje que le da decoración a toda esta parte, el cual es conocido como **Hojas de Estilo en Cascada** (del inglés *Cascading Style Sheets* - CSS). Estas hojas, son escritas como su nombre lo dice, en cascada y contienen una gran variedad de instrucciones para darle apariencia a la aplicación o plataforma desde cambiar el color de los textos hasta darle dimensiones dinámicas a la página para que tenga cierto comportamiento en distintos dispositivos.

Dado que muchos académicos, alumnos o administrativos navegan gran parte del tiempo en otros dispositivos que no son computadoras de escritorio o portátiles, como pueden ser dispositivos móviles, tablets e inclusive relojes inteligentes, es necesario que la plataforma tenga diseños adaptables a todos estos, lo cual en el argot del desarrollo de aplicaciones se le conoce como *modo responsivo (del inglés responsive)* que alude a que todo el diseño se adapte perfectamente a las dimensiones de los dispositivos más pequeños o grandes que los habituales.

Hasta ahora, se ha partido de lo general a lo particular. En este sentido, una vez definidas las características de apariencia, es necesario darle más dinámica a la aplicación o plataforma en cuestión. Cuando se navega por alguna de estas apps en línea, continuamente se observa que los mensajes, imágenes, videos, textos, etc., tienen efectos que lo hacen ver con un "plus" a la plataforma, para ello (esta vez ya no es solo un lenguaje), este es de programación y se conoce como **JavaScript**; lo es porque en el se pueden usar sentencias de control, funciones, objetos, programación orientada a objetos etc, para controlar toda la parte de HTML y CSS con programación. No solamente se puede hacer todo lo descrito anteriormente, sino inclusive controlar la parte del servidor para ejecutar operaciones con la base de datos a lo cual, esto es llamado como **AJAX** (del inglés *Asynchronous JavaScript And XML*) lo cual se adaptará perfectamente a las necesidades requeridas por el departamento de Telecomunicaciones para establecer la comunicación cliente-servidor entre alumnos y académicos de los laboratorios.

Una vez descrita la parte de *Frontend*, vendrá la esencia de la conexión con la plataforma. Está fracción del diseño como se ha mencionado anteriormente es el *Backend* y en ella radicará toda la estructura principal de la base de datos. A diferencia de la parte *Frontend*, en *Backend* se tiene que utilizar un lenguaje o lenguajes de programación que permitan establecer la conexión de una base de datos con un servidor alojado en alguna parte de internet. Si bien, ya se describió el principio de todo esto con *AJAX* y *JavaScript*, no es suficiente. Para ello, se estableció el uso de **SQL** (del inglés - *Structured Query Language*) para el diseño de la base de datos utilizando su servidor **MySQL** para implementar una base de datos de tipo *relacional*. Como parte de la comunicación de la plataforma es necesario establecer un lenguaje intermediario entre SQL y el servidor, este será manejado por **PHP** (del inglés - *Hypertext Preprocessor*).

Finalmente, como parte del diseño, existen algunos en el desarrollo de aplicaciones web que se implementan a manera de facilitar, reducir y automatizar el código. Con PHP se puede construir un modelo conocido como *Model/View/Controller - MVC* que es una arquitectura que permitirá organizar todo el proyecto.

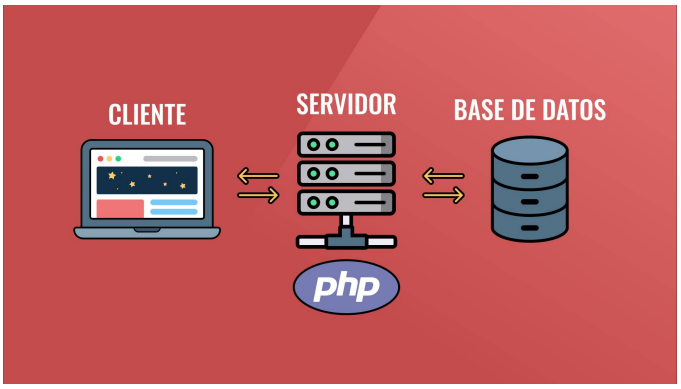


Figura 1: Esquema de un cliente/servidor. Fuente: A. (2021, 23 junio). Diseño y programación web. https://www.antoniobonastre.com/diseno-web-cliente-servidor/

4. Tecnologías de desarrollo

4.1. HTML5 (*Hypertext Markup Language*)

HTML es el conjunto de símbolos o códigos de marcado que se colocan en un archivo que está destinado a mostrarse en una página web. Estos códigos y símbolos de marcado identifican elementos estructurales como párrafos, encabezados y listas. HTML también se puede usar para colocar medios (como gráficos, video y audio) en una página web y describir formularios para completar. El navegador interpreta el código de marcado y muestra la página. HTML permite la visualización de información independiente de la plataforma a través de una red. No importa en qué tipo de computadora se haya creado una página web, cualquier navegador que se ejecute en cualquier sistema operativo puede mostrar la página.

Cada código de marcado individual se denomina **elemento o etiqueta**. Cada etiqueta tiene un propósito. Las etiquetas se incluyen entre corchetes angulares, los símbolos **<y>**. La mayoría de las etiquetas vienen en pares: una etiqueta de apertura y una etiqueta de cierre. Estas etiquetas actúan como contenedores y, a veces, se denominan etiquetas de contenedor. Por ejemplo, el texto que se encuentra entre las etiquetas **<title>** y **</title>** en una página web se mostraría en la barra de título de la ventana del navegador. Algunas etiquetas se utilizan solas y no forman parte de un par. Por ejemplo, una etiqueta *
* que configura un salto de línea en una página web es una etiqueta independiente o autónoma y no tiene una etiqueta de cierre. La mayoría de las etiquetas se pueden modificar con atributos que describan mejor su propósito.

4.2. CSS3 (*Cascade Style Sheet*)

Los diseñadores web utilizan CSS para separar el estilo de presentación de una página web de la información en la página web. CSS se utiliza para configurar texto, color y diseño de página. CSS no es nuevo; fue propuesto por primera vez como estándar por el W3C en 1996. En 1998, se introdujeron propiedades adicionales para posicionar elementos de página web en el lenguaje con CSS nivel 2 (CSS2), que se utilizó durante más de una década antes de llegar a estado de "recomendación" oficial en 2011. Las propiedades de nivel 3 de CSS (CSS3) admiten funciones como la incrustación de fuentes, esquinas redondeadas y transparencia. Hoy en día se está diseñando el cuarto nivel de CSS, pero, aún no está 100 % en su versión oficial.

Las hojas de estilo se componen de reglas de estilo que describen el estilo que se aplicará. Cada regla tiene dos partes: un selector y una declaración:

- **Selector de reglas de estilo CSS**: El selector puede ser un nombre de elemento HTML, un nombre de clase o un nombre de identificación.

- **Declaración de reglas de estilo CSS**: La declaración indica la propiedad CSS que está configurando (como el color) y el valor que está asignando a la propiedad.

4.3. JavaScript

JavaScript es un lenguaje de scripting del lado del cliente basado en objetos interpretado por un navegador web. Se considera que JavaScript está basado en objetos porque se utiliza para trabajar con los objetos asociados con un documento de página web: la ventana del navegador, el documento en sí y elementos como formularios, imágenes y enlaces. Dado que JavaScript es interpretado por un navegador, se considera un lenguaje de programación del lado del cliente.

Este lenguaje es interpretado por el cliente. Esto significa que el navegador representará el código JavaScript, incrustado en el documento HTML. El trabajo del servidor es proporcionar el documento HTML. El trabajo del navegador web es interpretar el código en el archivo HTML y mostrar la página web en consecuencia. Debido a que todo el procesamiento lo realiza el cliente (en este caso, el navegador web), esto se conoce como **procesamiento del lado del cliente**. Hay lenguajes de programación que se ejecutan en el servidor, que se conocen como lenguajes de programación del lado del servidor. El procesamiento del lado del servidor puede implicar el envío de correo electrónico, el almacenamiento de artículos en una base de datos o el seguimiento de artículos en un carrito de compras.

4.4. PHP

PHP viene de las siglas en inglés *Hypertext Preprocessor* o traducido sería algo así como Procesador de Hipertexto. En realidad, más que buscar la etimología de este lenguaje, es darle un sentido aplicativo. PHP es en lenguaje de programación, que a diferencia de JavaScript, este funciona del lado del servidor, por lo que PHP permitirá no sólo darle más dinamismo a la página web en cuestión sino también, procesar y controlar la base de datos desde el servidor. Con PHP prácticamente se puede hacer una consulta a la base de datos, manipular el documento HTML, hacer peticiones tipo JSON, encriptar contraseñas, etc. Una de las ventajas de implementar PHP es que soporta la Programación Orientada a Objetos (POO) lo cual va a permitir generalizar el código y darle un mejor mantenimiento a la plataforma a desarrollar en un futuro. Además que se puede hacer uso de las APIs que ya están alojadas en la web para añadir funcionalidades como: crear archivos PDF con la información recabada de una base de datos, localización de mapas, envío de emails a través del protocolo SMTP (*Simple Mail Transport Protocol*), etc.

4.5. MySQL

MySQL es un sistema gestor de bases de datos (del inglés SGDB - *Database Management System*) desarrollado por Oracle basado en el lenguaje SQL que se puede encuadrar dentro de la categoría de los programas open-source. Aparte de las características que definen MySQL como programa open-source, existen aspectos que lo diferencian de otros productos como, por citar uno conocido, Access. Los atributos a los que hacemos referencia son:

- Posibilidad de crear y configurar usuarios, asignando a cada uno de ellos permisos diferentes.
- Facilidad de exportación e importación de datos, incluso de la base de datos completa.
- Posibilidad de ejecutar conjuntos de instrucciones guardadas en ficheros externos a la base de datos.

El factor principal que diferencia las bases de datos relacionales de otros dispositivos de almacenamiento digital radica en cómo se organizan los datos a un alto nivel. Las bases de datos como MySQL contienen registros en tablas múltiples, separadas y altamente codificadas, a diferencia de un único repositorio que lo abarca todo, o colecciones de documentos semiestructurados o no estructurados.

Esto permite que los RDBMS (del inglés *Relational Database Management System*) optimicen mejor acciones como la recuperación de datos, la actualización de información o acciones más complejas como las agregaciones. Se define un modelo lógico sobre todo el contenido de la base de datos, que describe, por ejemplo, los valores permitidos en columnas individuales, las características de las tablas y vistas, o cómo se relacionan los índices de dos tablas. Los modelos relacionales han seguido siendo populares por varias razones. Otorgan poder a los usuarios con lenguajes de programación declarativos e intuitivos, que esencialmente le dicen a la base de datos qué resultado se desea en un lenguaje similar, o al menos comprensible, en lugar de codificar meticulosamente cada paso del procedimiento que conduce a ese resultado.

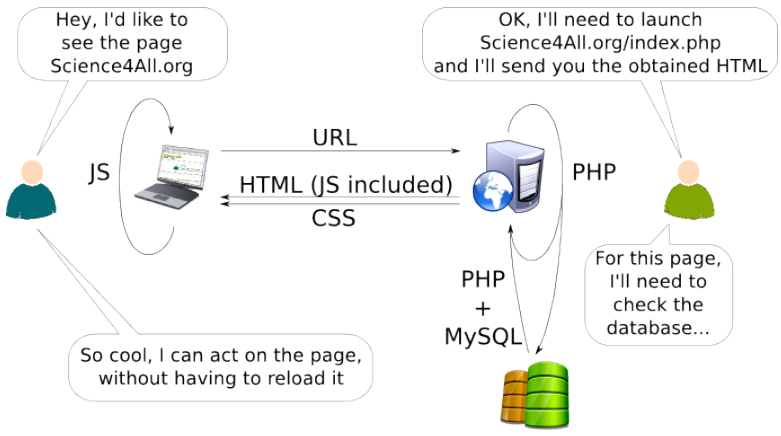


Figura 2: Proceso completo con las tecnologías presentadas para el desarrollo del proyecto. Fuente: Science4All. (2012). http://www.science4all.org/wp-content/uploads/2012/07/

5. Diseño Front-end con HTML5, CSS3 y JavaScript

Para empezar a diseñar la parte frontal de la plataforma, se pidió basarse en la línea de sitios web que se cuentan en la Facultad de Ingeniería. Es decir, utilizar la misma estructura para el menú, la misma gama de colores HTML y además de las presentaciones para redes sociales del departamento (en este caso). Sin embargo, respetando la línea gráfica de las plataformas de la Facultad, se hará la propia distribución de páginas de interés cuando los usuarios naveguen por dicha plataforma. El editor de código que se usó para escribir todo el proyecto fue **Visual Studio Code**, lo cuál además de su gran interfaz, posee herramientas de automatización para ahorrar tiempo usando atajos al ir desarrollando todo el proyecto.

Antes de escribir el código necesario para darle forma a la plataforma, es esencial mencionar las partes en las que estará divido todo el contenido del sitio web, estos son:

- **Header o encabezado**: contendrá una imagen de portada y el menú que presenta cada una de las páginas en las que se podrá navegar;
- **Body o cuerpo**: contenido principal de la plataforma;
  - **Article o secciones**: estará dividiendo algunos botones principales, imágenes informativas de las redes sociales del departamento;
  - **Sidebar o barra lateral**: Contendrá únicamente los plugins necesarios para mostrar las redes sociales en tiempo real.
- **Footer o pie de página:** contiene la información de contacto de la facultad y del departamento, así como algunos enlaces de interés.

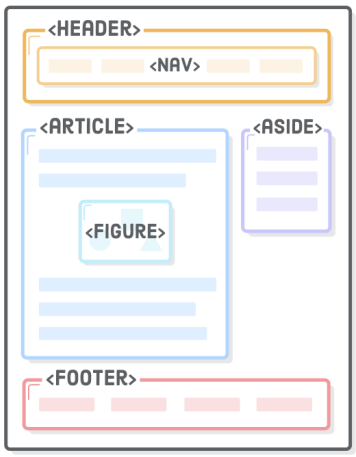


Figura 3: Estructura de una página web. Fuente: Delhi, W. (2019). Role of HTML, JavaScript, PHP in Web Development. Web Development Institute. https://www.web-development-institute.com/role-html-javascript-and-php-web-development/

(a) Menú principal de la Facultad de Ingeniería (b) Footer de la página de la Facultad de Ingeniería

Figura 4: Estructura de diseño en página de la Facultad de Ingeniería

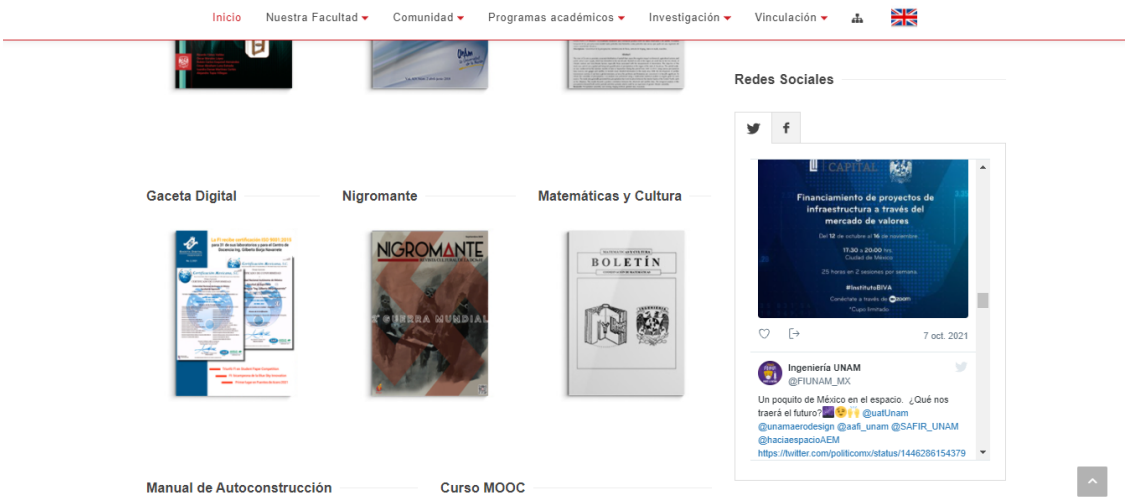


Figura 5: Sidebar de la página de la Facultad de Ingeniería.

Como se comentó anteriormente, la línea que se siguió para el diseño de cada una de las partes fue de las páginas de la Facultad de Ingeniería, esto para tener consistencia con los colores, formas y logotipos de la propia facultad. Sin embargo, cabe mencionar que las demás páginas que estarán incluidas en este proyecto tendrá aspectos propios de diseño acorde al Departamento de Telecomunicaciones.

5.1. Primera versión

La primer página que se empezó a desarrollar fue la del **inicio**. Esta página estará centrada en focalizar anuncios importantes a través de un carrete de imágenes que el administrador posteriormente podrá subir y publicar en la plataforma. También, presentará algunos botones que redirigirán al usuario para obtener más información con relación a la facultad y al departamento. Contendrá así mismo, el sidebar mencionado que visualiza las redes sociales del departamento, éstas serán dos: *facebook y twitter*. Finalmente, el footer o pie de página que contendrá la información de contacto del departamento. El código HTML se dividió como se muestra a continuación, a fines de resumir, no se muestra todo el contenido debido a su amplitud:

```html
<main>
    <section class = "contenedor">
        <div class = "carrete"></div>
    </section>

    <section class = "botones-main">
        <a href = "#"></a>
    </section>

    <div class = "contenido-grid__sidebar">
        <aside></aside>
    </div>

    <footer class = "contenedor"></footer>
</main>
```

Figura 6: Estructura resumida de la página principal de la plataforma.

Las clases *contenedor, carrete y contenido-grid___sidebar, botones-main* ayudarán a reciclar código cuando se este diseñando la parte gráfica con CSS. En esencia, dado que HTML es un lenguaje de marcado, nos permite estructurar exactamente los bloques que se mencionaron anteriormente para el diseño. Resta escribir las etiquetas propias para los botones, párrafos, encabezados, títulos, etc. y asignarles clases para darle apariencia con CSS.

```css
.contenedor{
    margin: 0 auto;
    /* estilos aquí */
}

.botones-main{
    background-color: #c90000;
    /* estilos aquí */
}

.contenido-grid__sidebar{
    display: grid;
    /* estilos aquí */
}
```

Figura 7: Estructura resumida del código de CSS para proporcionar estilos a los elementos HTML.

En resumen, el código de la figura 6 muestra la hoja de estilos de CSS para darle una buena apariencia a cada uno de los elementos descritos anteriormente en el HTML. Nuevamente, con fines de no extender mucho la información, se describe la parte esencial de estas hojas de estilos. Con CSS se podrá darle posición a los elementos, colores, animaciones sencillas, rotación, etc., requeridas para el público que visitará la página continuamente del departamento.

Como ya se ha definido la estructura HTML y CSS que tendrá cada una de las páginas, fue necesario hacer un modelo descriptivo de todas esas páginas así como las posibles subpáginas que podrían originarse. Esto se hizo para tener en cuenta lo que contendría el menú de la plataforma. El siguiente esquema representa cada una de las páginas que fueron propuestas por el Departamento de Telecomunicaciones:

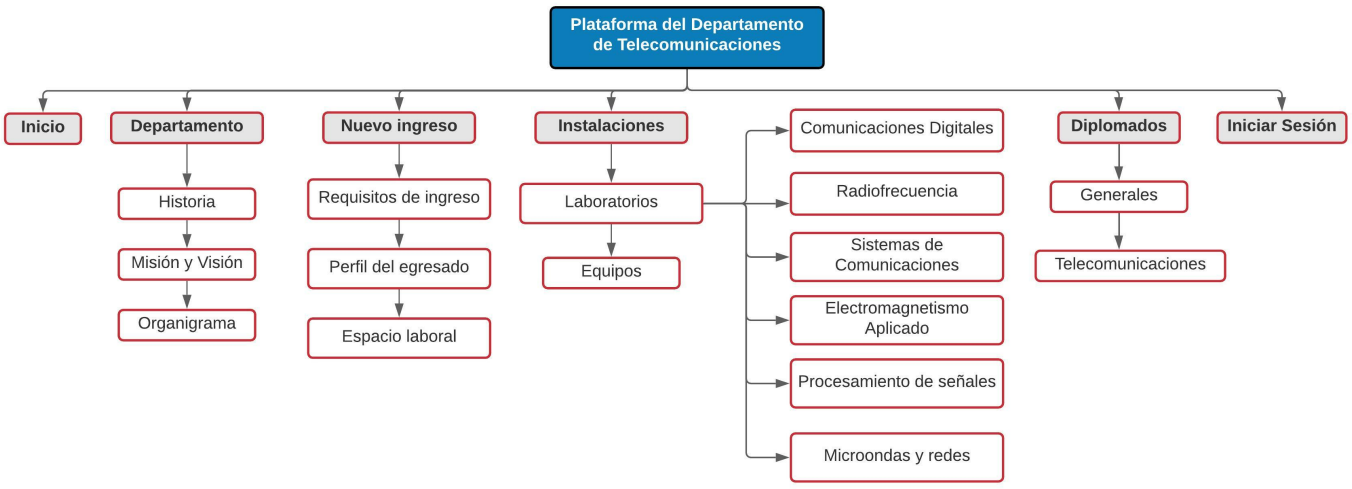


Figura 8: Esquema de páginas para la plataforma.

Estos pasos permitieron ahorrar tiempo antes de empezar a escribir código y empezar a pensar en herramientas de automatización para terminar lo más pronto posible el proyecto. Existen algunas herramientas que están basadas en código JavaScript, dado que la plataforma del departamento contendrá muchas páginas y demasiado código que hay que estar desarrollando, es un gran alivio tener en cuenta estas herramientas de automatización para ahorrar tiempo y espacio en el desarrollo del proyecto. Una es conocida como SASS-Gulp. SASS nos permite preprocesar muchos ficheros de estilo (CSS) con una sintaxis nueva para mejorar nuestro trabajo. Estos ficheros SASS, permiten hacer funciones, importar unos archivos en otros, declarar variables que constantemente se usarán, etc.

Para hacer uso de estas herramientas es necesario añadir al proyecto una carpeta llamada: **scss** que contendrá a su vez dos más: **base** y **layout**. Dentro de base estarán todas las variables a declarar para reutilizarlas en código CSS, funciones para implementar diseño responsivo, etc., mientras que en layout vendrá todas las hojas de estilo como se tal y como se describió en la figura 7.

De esta forma, todas las hojas de estilo que se escriban Gulp las juntará en un solo archivo **.css** y también, todos los códigos de JavaScript que se utilicen serán mezclados en un solo archivo **.js**, con la finalidad de que estarán comprimidos para que una vez cargados en el navegador web, este le sea fácil de descargarlos, esto le dará más *perfomance* a la plataforma.

```
const { src, dest, watch , parallel } = require('gulp');
const sass = require('gulp-dart-sass');
const autoprefixer = require('autoprefixer');
const postcss = require('gulp-postcss');
const sourcemaps = require('gulp-sourcemaps');
const cssnano = require('cssnano');
const concat = require('gulp-concat');
const rename = require('gulp-rename');
const cache = require('gulp-cache');

const paths = {
    scss: 'src/scss/**/*.scss',
    js: 'src/js/**/*.js',
    imagenes: 'src/img/**/*'
}

function css() {
    return src(paths.scss)
        .pipe(sourcemaps.init())
        .pipe(sass())
        .pipe(postcss([autoprefixer(), cssnano()]))
        .pipe(sourcemaps.write('.'))
        .pipe( dest('./public/build/css') );
}

function javascript() {
    return src(paths.js)
      .pipe(sourcemaps.init())
      .pipe(concat('bundle.js'))
      .pipe(sourcemaps.write('.'))
      .pipe(rename({ suffix: '.min' }))
      .pipe(dest('./public/build/js'))
}

function watchArchivos() {
    watch( paths.scss, css );
    watch( paths.js, javascript );
    watch( paths.imagenes, imagenes );
    watch( paths.imagenes, versionWebp );
}

exports.css = css;
exports.watchArchivos = watchArchivos;
exports.default = parallel( css, javascript, watchArchivos );
```

Figura 9: Código para el archivo gulpfile.js que permite automatizar tareas.

Dado que, el departamento requiere que ciertas páginas tenga un contenido y otro, se tuvieron que diseñar algunos bocetos de la estructura de cada una de ellas, siempre y cuando respete el estándar de diseño para páginas web presentada anteriormente.

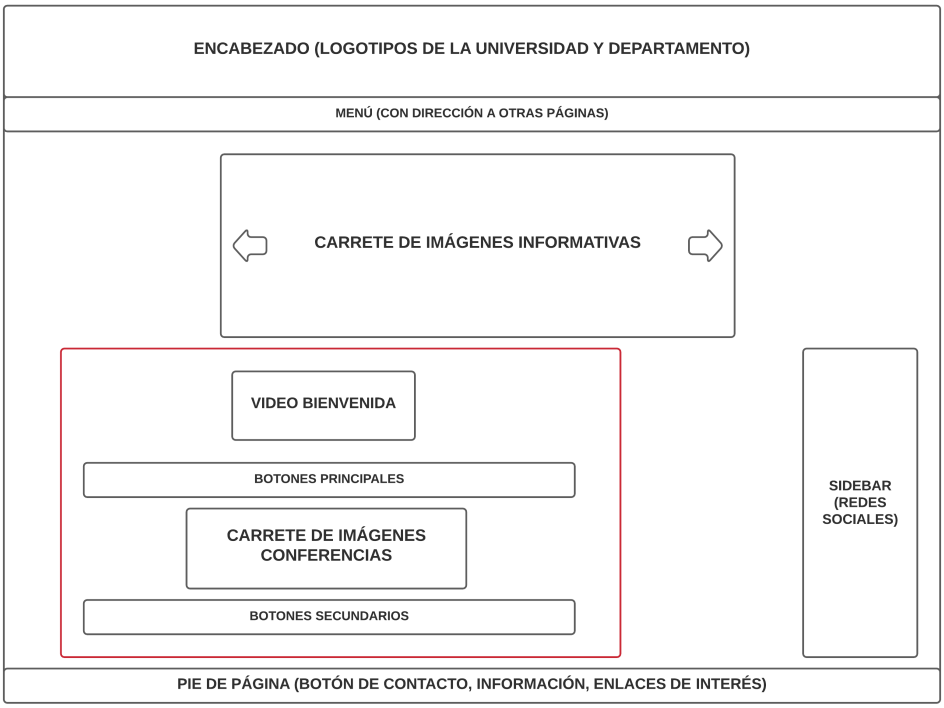


Figura 10: Estructura del documento HTML para el inicio de la plataforma.

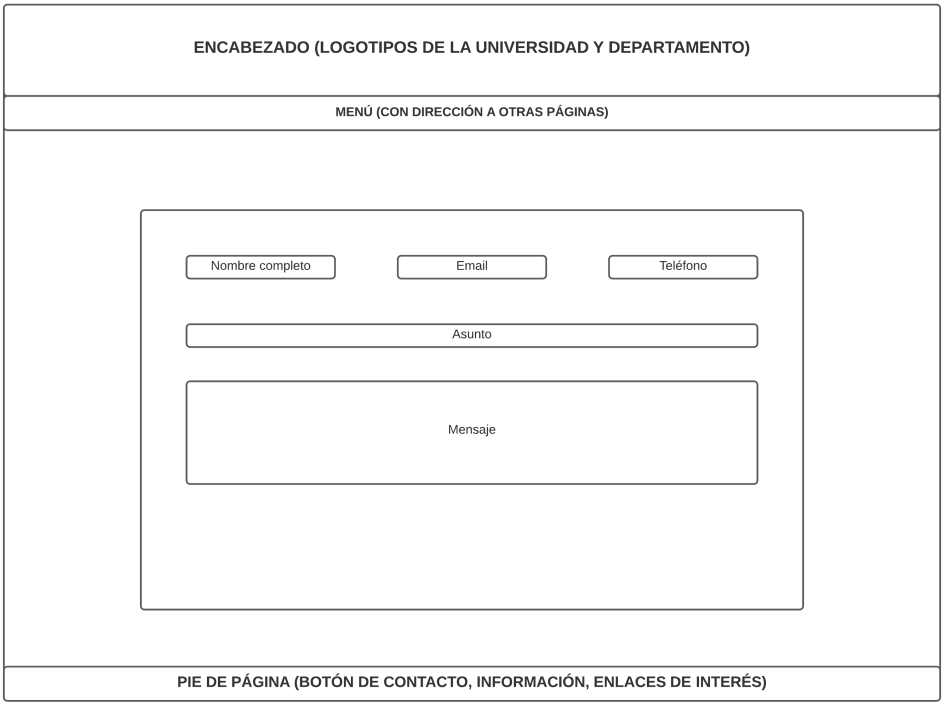


Figura 11: Estructura del documento HTML para el formulario de contacto.

En la figura 12 se muestra el cuerpo de las páginas secundarias, dado que éstas solo llevan una sección principal, solo se listaron los requerimientos para cada una de ellas.

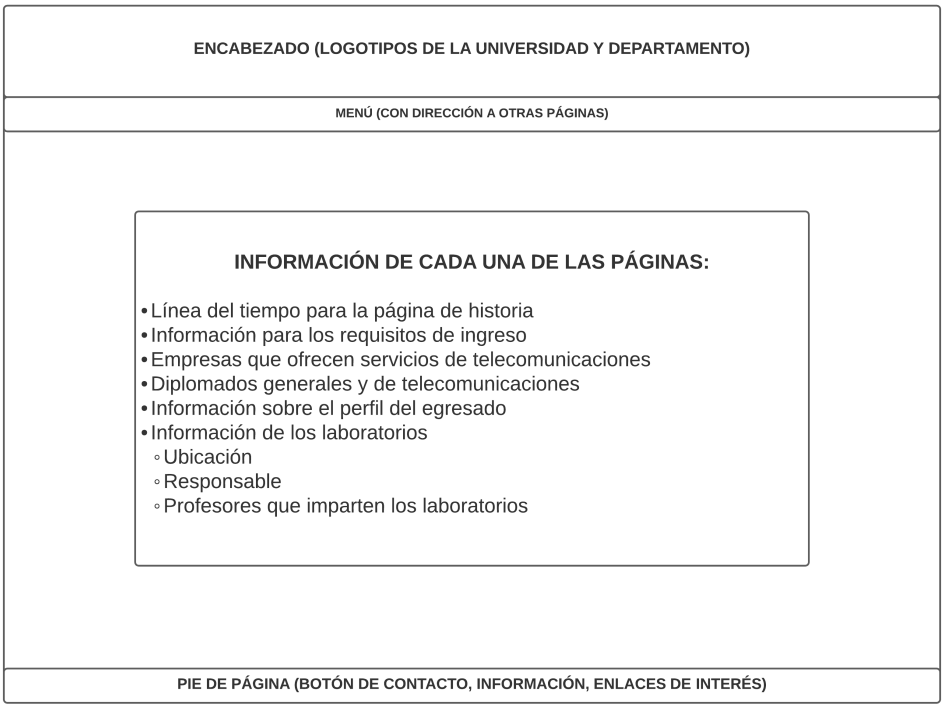


Figura 12: Estructura del documento HTML para las páginas secundarias de la plataforma.

Una vez escritas todas las páginas HTML y CSS, añadiendo un poco de JavaScript para ponerle dinámica a los elementos de la plataforma (por ejemplo, mostrar imágenes en el carrete principal, añadir las redes sociales utilizando las APIs de Facebook y Twitter). Algo muy importante es que, para visualizar los resultados de la primera versión hay que abrir una conexión **loopback** con el **servidor de Apache2**. Visual Studio Code ya proporciona esta funcionalidad. La primera versión tenía la siguiente apariencia:

Figura 13: Página principal de la plataforma para el Departamento de Telecomunicaciones.

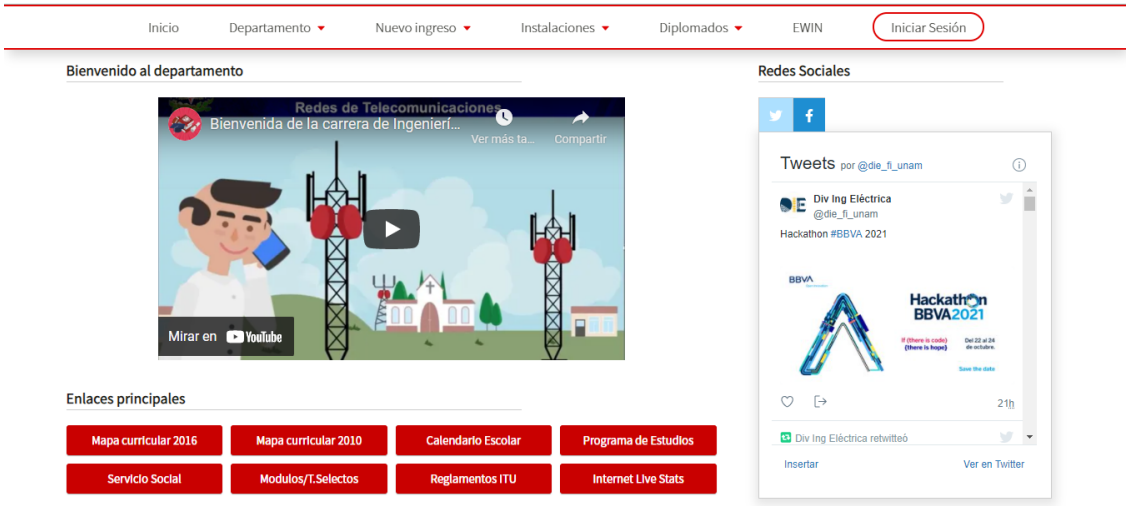
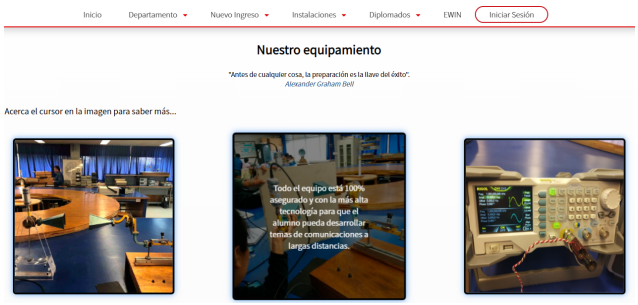


Figura 14: Secciones principales y Sidebar.

Figura 15: Footer.

Figura 16: Formulario de contacto.

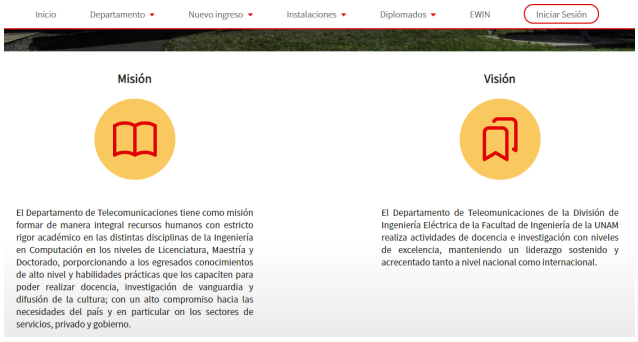
Figura 17: Página: Perfil del Egresado.



(a) Página: Equipo.

(b) Página: Historia.

Figura 18: Páginas diseñadas del departamento.



(a) Página: Misión y Visión.

(b) Página: Requisitos de ingreso.

Figura 19: Páginas diseñadas del departamento.

(a) Página: Diplomados de Telecomunicaciones.

(b) Página: Laboratorios.

Figura 20: Páginas diseñadas del departamento.

Finalmente, dado que la página oficial de la Facultad de Ingeniería tiene un menú interactivo que hace que cuando los usuarios hacen scroll por la plataforma, este menú queda fijo en la parte superior. Eso, se logró con una implementación de JavaScript.

```
function posicionarMenu() {

    const barra = document.querySelector('.container-navegacion');

    if(!barra){
        return;
    }
    // Registrar Intersection Observer
    const observer = new IntersectionObserver(function (entries) {
        if (entries[0].isIntersecting) {
            barra.classList.remove('fijo');
        } else {
            barra.classList.add('fijo');
        }
    });

    // Elemento a observar, en este caso el header.
    observer.observe(document.querySelector('.header'));
}
```

Figura 21: Código para fijar el menú en la parte superior del navegador.

El desarrollo de la hoja de estilos de CSS fue demasiado largo debido a la cantidad de apariencia que tienen los elementos HTML. En resumen, las propiedades más usadas en CSS fueron las siguientes:

- **background-color:** Permite darle color a los elementos;
- **display: flex/grid:** Posiciona los objetos HTML en el centro, arriba, izquierda, derecha, abajo;
- **font:** Font tiene muchas sub-propiedades, como el tamaño, aspecto y tipo de la fuente;
- **pseudo-selectores**: hover, focus, before, after, son algunos que más se ocuparon para darle dinámica y/o insertar elementos en el documento HTML.

5.2. Diseño responsivo para dispositivos móviles y tablets

Parte del proyecto es llevarlo a ser adaptable en los distintos dispositivos portables que hoy ya existen. La mayoría de las aplicaciones tienen la característica de ser flexibles en el aspecto visual en el cuál se están usando. Por ello, una vez desarrollada la primera versión del sitio o plataforma, es necesario implementar un patrón de diseño para dispositivos móviles y tablets. Ya que la plataforma puede ser inestable e inclusive algunos elementos que se ven perfectamente en el navegador de una computadora, simplemente no aparezcan. Esto debido a que los elementos están dentro de una resolución de pantalla definida.

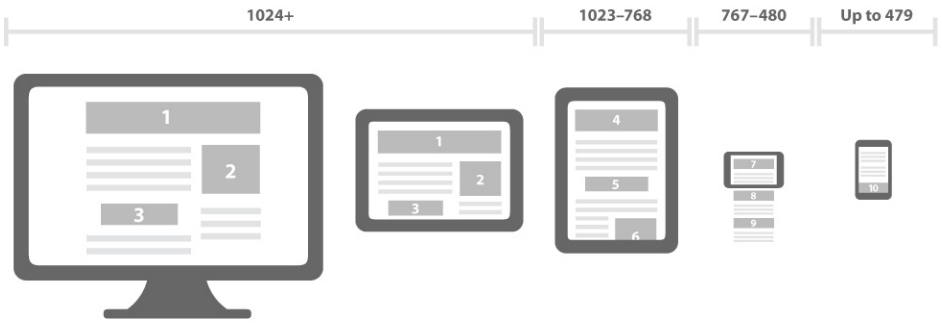


Figura 22: Resolución de dispositivos, desde computadoras de escritorio hasta móviles. Fuente: *Is your website mobile friendly? If not, you are losing customers. . .fast!* (2013). Media Consulta. http://www.media-consulta.com.mt/fileadmin/user_upload/mobile-friendly-sites/

Gracias a que desde un principio se implementaron herramientas de automatización, como se mencionó anteriormente, estas resoluciones mostradas en la figura 22 pueden ser perfectamente definidas en el ambiente de GULP. Para hacer uso de la definición de resolución de cada dispositivo, es necesario hacer uso de la técnica de **Media Queries** en CSS.

```
@mixin telefono {
    @media (min-width: #{$telefono}) {
        @content;
    }
}
@mixin tablet {
    @media (min-width: #{$tablet}) {
        @content;
    }
}
@mixin desktop {
    @media (min-width: #{$desktop}) {
        @content;
    }
}
```

Figura 23: Código para definir las distintas resoluciones en CSS con GULP.

De lo anterior, se puede observar que dentro de *@content* se escribe el código CSS como normalmente se hace para adaptar los elementos HTML ya desarrollados en la primera versión. Un punto importante es que, lo que ya se escribió en la versión 1 para cada uno de los elementos en el documento, no es necesario volverlos a escribir, esto gracias a la técnica de *media queries* implementada. Así que, solo es necesario escribir el código faltante para adaptar dicho elemento a la resolución del dispositivo.

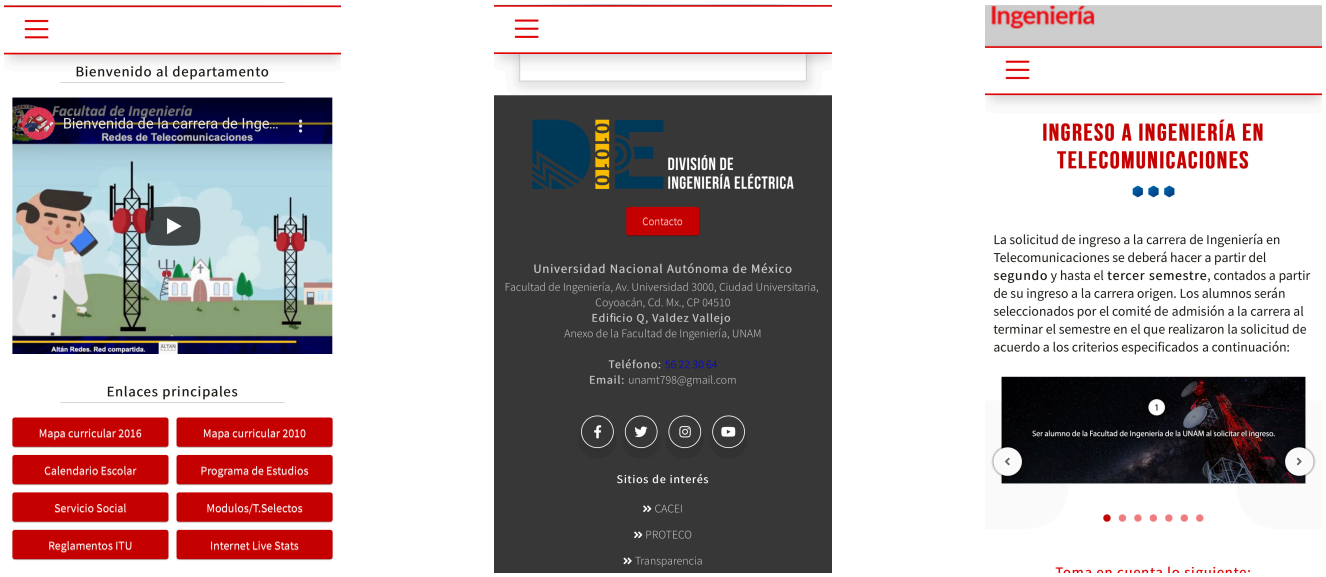
```
/* Familia de fuentes */
$fuente_principal: 'Noto Sans SC', sans-serif;
$fuente_secundaria: "Helvetica Neue", Helvetica, Arial, sans-serif;
$fuente_tablas: 'Quicksand', sans-serif;
$fuente_encabezados: 'Bebas Neue', cursive;

/* Tamaño de Media Queries (Dispositivos Moviles) */
$miniphone: 340px;
$telefono: 480px;
$tablet: 768px;
$desktop: 1024px;

/* Tamaño Fuentes */
$delgada: 300;
$regular: 400;
$mediana: 500;
$bold-light: 600;
$bold: 700;
```

Figura 24: Variables definidas como constantes con GULP.

De igual manera, las variables pueden ser declaradas como constantes con estas herramientas, aspecto que simplemente es muy complicado si sólo implementáramos CSS duro en el proyecto. La primera versión de la implementación de este diseño se muestra en las siguientes figuras.



(a) Sección principal. (b) Pie de página. (c) Requisitos de ingreso.

Figura 25: Páginas con diseño responsivo en dispositivos móviles.

La ventaja de los media queries es que se adaptan al código de JavaScript aplicado anteriormente para el menú, de esta forma, éste queda fijo al ir navengando desde el dispositivo. Sin embargo, se requirió código CSS adicional para poder ponerlo justo en la posición mostrada para dispositivos más pequeños.

Figura 26: Menú desplegable de la plataforma con diseño responsivo.

5.3. Segunda versión

Aunque la primera versión parecía estable, con el paso del tiempo y de las pruebas ejecutadas en los primeros meses del proyecto, se observaron algunos detalles que fueron corregidos posteriormente. Si bien, la parte de back-end aún no estaba implementada hasta cierto punto, los cambios que se realizaron fueron simplemente de front-end, es decir, de CSS, HTML y JavaScript. A continuación se exponen en una tabla aquellos problemas que fueron corregidos.

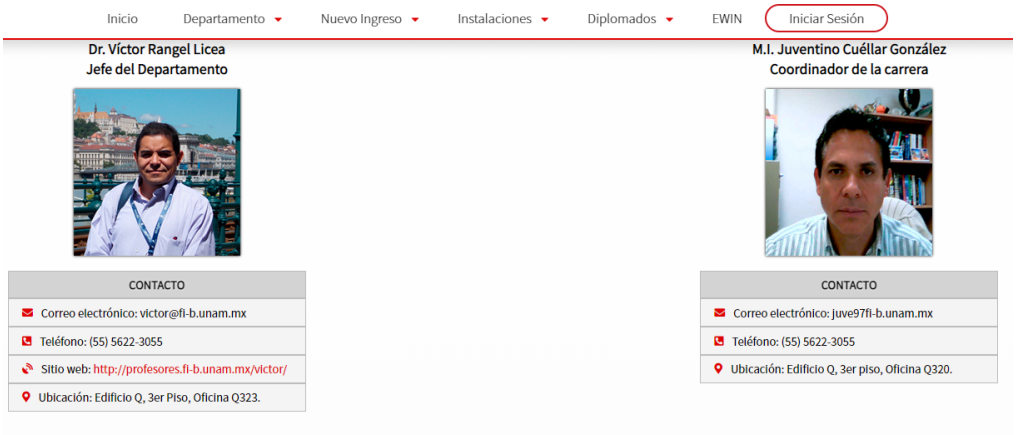
Dado que a este punto ya estaba desarrollado el patrón responsivo, también hubo que hacer modificación en los *@content* de cada resolución. Uno de los detalles que más sobresalió fue que en el página de organigrama, en los datos de cada profesor, el correo electrónico era mostrado en texto directamente, eso a futuro provocaría spam en el correo de cada uno de ellos, o que sean pasados por algún programa o algoritmo que recoja los correos electrónicos para fines no propios del departamento. Esto último fue solucionado con una API de Google para evitar el spam producido por bots.

Defecto	**Corrección**
Diseño inestable del formato para laboratorios	Cambio en la hoja de estilos de la sentencia display de **flex a grid**
Menú responsivo no se mostraba completo en dispositivos móviles	Implementación de la parte frontal con JavaScript utilizando la técnica de **Manipulación del DOM**.
Los correos de profesores en la página **organigrama** se mostraban sin protección anti-spam	Para evitar spam se añade la funcionalidad de un captcha anti-bots que previene dicho problema. Adición de la **API reCAPTCHA** v2.0 de Google.
Contenedor de todas las páginas se mostraba hasta las paredes laterales de la pantalla	Ajuste en las dimensiones del contenedor.
Mala gama de colores utilizados en algunas páginas	Utilización de los colores HTML adecuados, indicados por el departamento.

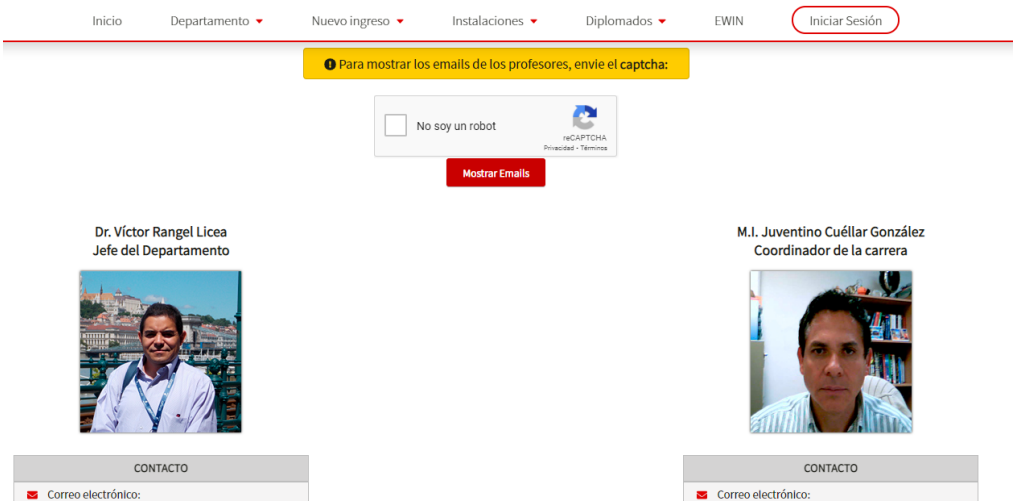
Cuadro 1: Tabla de defectos y correcciones de la versión 1.0

(a) Versión 1 del apartado de laboratorios.

(b) Versión 2 del apartado de laboratorios.



(a) Organigrama sin protección anti-bots.



(b) Organigrama utilizando la API Google

5.4. Vistas: sesiones de usuarios

La última versión de la página incluye todas las vistas que los usuarios de manera pública podrán ver, como la sección principal, organigrama, historia, laboratorios que se imparten en el departamento, etc., además de ello es importante desarrollar las vistas para los académicos y estudiantado que accederá a sus cuentas privadas para ver las calificaciones de laboratorio, que es prácticamente la esencia de este proyecto.

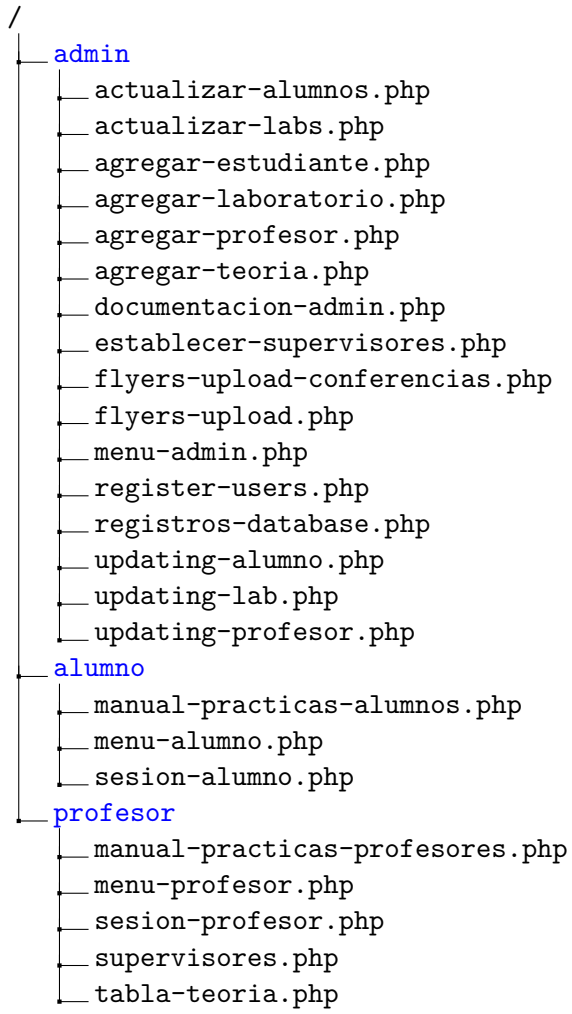


Figura 29: Árbol de directorios para las vistas

El anterior árbol de directorios muestra las carpetas y cada una de las vistas que contendrá cada una, todas estas páginas en formato ***.php** (posteriormente se abordará como se trabajó con este lenguaje), tienen lenguaje HTML, CSS y JavaScript. Hasta este momento, no es posible implementar tecnología de back-end, ya que la base de datos aún no estaba en creación, sin embargo, es necesario tener el pre-diseño para que sea más fácil desarrollar la parte con el servidor de la base de datos y el lenguaje de programación, que en este caso es PHP.

5.4.1. Administrador

Como parte del diseño front-end, la vista de administrador será la más compleja en cuanto a contenido, debido a que será la función que controlará prácticamente toda la plataforma. Las funciones que realizará, literalmente, están escritas en cada uno de los archivos que se presentaron en el árbol de directorios.

```
<div class="container-session">
    <?php include __DIR__ . '/menu-admin.php' ?>
    <main class="contenedor-sesion">
        <div class="topbar"></div>
        <div class="informacion-sesion">
            <table class="table-grades">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>No. Cuenta</th>
                        <th>Nombre/Apellido</th>
                        <th>Carrera</th>
                        <th>Correo electronico</th>
                        <th>Acciones</th>
                    </tr>
                </thead>

                <tbody>
                    <?php foreach ($registros as $registro) : ?>
                        <tr></tr>
                    <?php endforeach; ?>
                </tbody>
            </table>
        </div>
    </main>
</div>
```

Figura 30: Estructura general de las páginas del administrador.

Como se puede observar, en la estructura general, se creó una página .php especial para el menú, ya que todas las páginas lo incorporan, finalmente se hace la importación del archivo. Luego, se iterará cuando se tenga la base de datos, todos los registros de los alumnos, profesores, calificaciones, etc,. para el control de toda la plataforma. Las clases mostradas en cada uno de los div, permitirán darle estilos con CSS a cada uno de los elementos. Cabe resaltar, que la interfaz de las vistas en el inicio de sesión será original e independiente de las anteriores vistas, donde se requiere que la línea sea de las páginas de la Facultad de Ingeniería.

Figura 31: Panel principal del Administrador.

Además de poder registrar a los alumnos y profesores, también podrá visualizar la cantidad de cada uno de los registros en el panel superior. Adicionalmente, una sección de los laboratorios actuales en la base de datos.

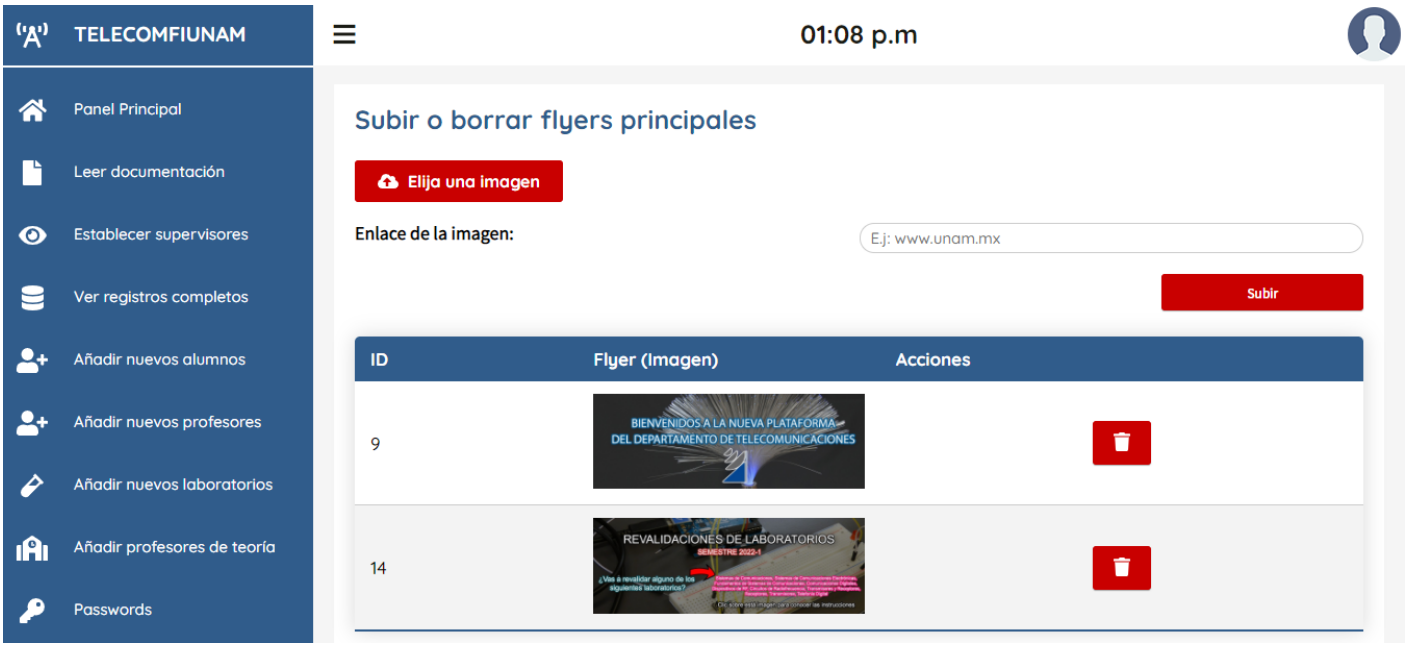


Figura 32: Sección para subir flyers al inicio de la página.

Figura 33: Sección para actualizar datos de los registros.

No se muestran todas las páginas porqué no todas fueron diseñadas hasta este punto, solo las principales para ir fijando los objetivos que tendrá cada una de las vistas. Los datos que están en las imágenes fueron añadidos manualmente para observar como se adaptaban las respectivas tablas de información.

5.4.2. Profesores

En la sesión de tipo profesor la base principal es la visualización de los datos de sus alumnos ya sea como **laboratorio** o **teoría**, a través de una tabla. Esta tabla se diseñó con *CSS* y **JavaScript**, para que los profesores puedan buscar rápidamente algún dato, se añadió una entrada de búsqueda, además de poder filtrar los datos por orden alfabético en cada uno de las columnas de la tabla.

Nuevamente, la parte lógica y dinámica estará implementada más adelante. Aunque, siempre se recomienda tener algunos datos para poder probar la parte frontal o la interfaz del usuario para no experimentar fallas más adelante en cuanto a visualización. El profesor de laboratorio una vez que pondere la calificación al estudiante este quedará registrada y no podrá ser editada. Otro detalle, las tablas muestran la cantidad de datos que el usuario elija, esto ya es a consideración y a gusto en este caso, de cada profesor.

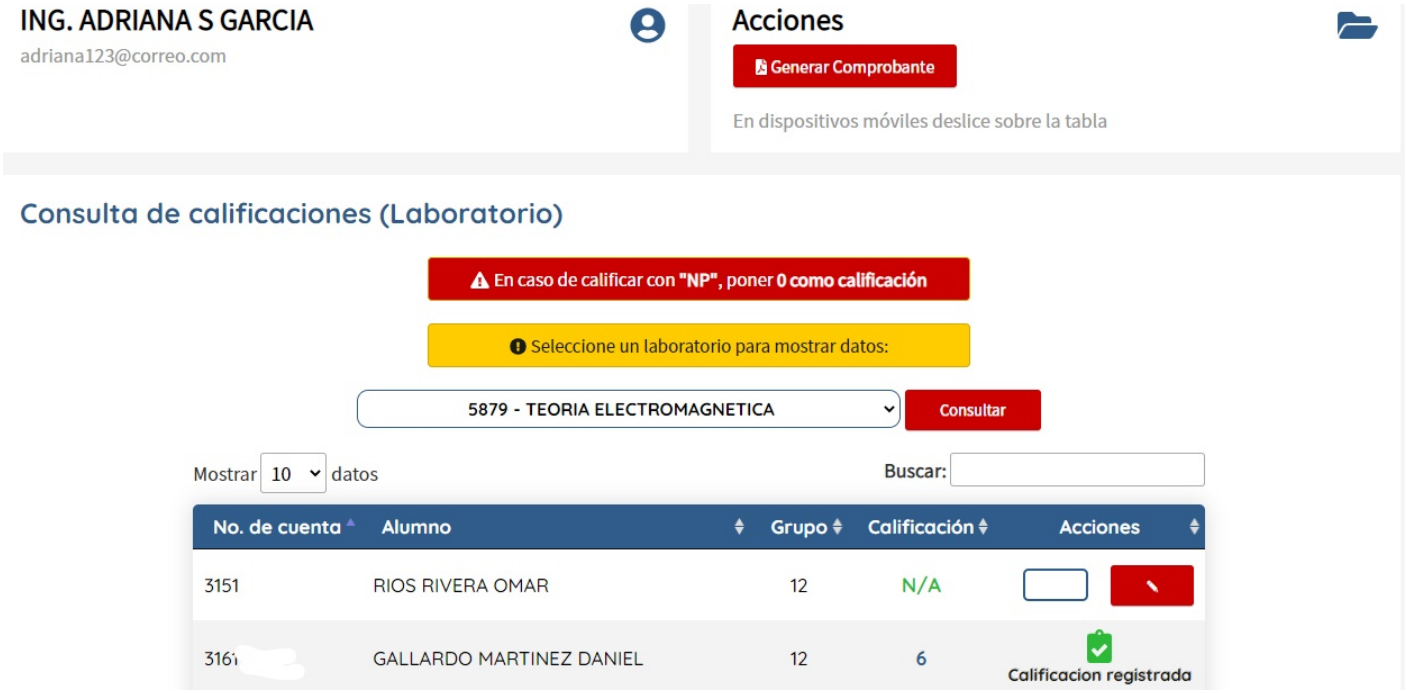


Figura 34: Sección para profesores de laboratorio.

Figura 35: Sección para profesores de teoría.

En el caso de que un profesor no solo sea de laboratorio sino que también de materias de teoría podrá observar ambas tablas mostradas anteriormente.

```
<div class="container-session">
    <?php include __DIR__ . '/menu-profesor.php' ?>
<main class="contenedor-sesion">
    <div class="topbar">
        <div class="toggle-session"></div>
    </div>

    <div class="card">
        <div class="information-box">
            <div class="numbers number-edit">Acciones</div>
            <div class="card-name">
                <div class="botones-opciones">
                    <form action="/profesor/generarpdf-profesor" method="POST">
                        <button type="submit" class="btn btn-red gpdf"></button>
                    </form>
                </div>
                En dispositivos móviles deslice sobre la tabla
            </div>
        </div>
    </div>
</div>

    <div class="details-session reset-contenedor">
        <div class="list-users">

        <?php if ($laboratorio) : ?>
            <div class="card-header">
                <h3 class="h3-reset h3-black">Consulta de calificaciones (Laboratorio)</h3>
            </div>
            <!--LABORATORIO TABLA-->
            <div class="tabla-seleccion">
                <p class="instructions-header instructions-header-red"></p>

                <form class="form-create" action="/profesor/sesion-profesor" method="POST">
                    <select class="filter-labs" name="labSelected" id="option-labs">
                        <option value="3">-- Seleccione un laboratorio --</option>
                        <?php foreach ($datosLab as $data) : ?>
                            <option></option>
                        <?php endforeach; ?>
                    </select>

                    <input type="hidden" value="selected" name="option-selected">
                    <input class="consultar btn-mv btn-red-mv" type="submit" value="Consultar">
                </form>

            </div>
    </main>
</div>
```

Figura 36: Estructura general de la página de profesores.

5.4.3. Alumnos

Sin duda, la parte más fácil es para el inicio de sesión de los alumnos. Una vez implementadas las tablas para profesores y el administrador, ese mismo diseño se puede reutilizar para mostrarle los datos necesarios a los alumnos, en este caso solamente los laboratorios que le corresponde (esta parte estará validándose con programación) además, una sección en el menú para que puedan descargar los manuales de laboratorio de interés.

Figura 37: Sección principal para alumnos.

Figura 38: Sección de manuales de laboratorio para alumnos.

Al dar clic en alguno de los manuales, los alumnos podrán descargarlos en formato .pdf. Esto podra controlarlo perfectamente el administrador de la plataforma.

Tanto en profesores como alumnos, en la parte superior encontrarán un botón que indica *generar comprobante*, especialmente este botón tiene como objetivo imprimir un **documento en formato PDF** para comprobar las calificaciones vistas en la plataforma, esto se añadió con fines de comprobación en dado caso que la base de datos no este sincronizada correctamente. Esta parte es puramente programable con algún lenguaje, que más adelante se detallará.

```
<div class="container-session">
    <?php include __DIR__ . '/menu-alumno.php'?>
    <main class="contenedor-sesion">
        <div class="topbar">
            <div class="toggle-session"></div>
        </div>

        <div class="cardbox two-columns">

            <div class="card">
                <div class="information-box"></div>
            </div>
        </div>

        <div class="details-session reset-contenedor">
            <div class="list-users">

                <div class="informacion-sesion">
                    <table class="table-grades">
                        <thead>
                            <tr>
                                <th>Clave de laboratorio</th>
                                <th>Laboratorio</th>
                                <th>Profesor</th>
                                <th>Grupo</th>
                                <th>Calificacion</th>
                            </tr>
                        </thead>
                        <tbody>
                            <?php foreach ($datosCalificacion as $dato) : ?>
                                <tr></tr>
                            <?php endforeach; ?>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </main>
</div>
```

Figura 39: Estructura general de la página de alumnos.

6. Diseño Back-end con PHP y MySQL

Para poder desarrollar la parte de inicio de sesión en la plataforma tanto para *profesores*, *alumnos* y *administrativos* es que se llega a lo que forma parte de las aplicaciones que se le conoce como **back-end** o bien, la parte dinámica en donde reside toda la interacción y manipulación de los datos de todos los registros que estarán almacenados en una base de datos.

Cuando se llega a la parte de back-end en una aplicación web lo que se tiene que tener en mente es que es necesario implementar toda la lógica en un lenguaje de programación sí o sí. En la decisión entre elegir Java o PHP como lenguaje base que se encargaría de toda la estructura de programación para lograr el **inicio de sesión**, se llegó a la conclusión de que **PHP** era el mejor en este caso debido a que las conexiones con MySQL eran más accesibles y se tenía un conocimiento más amplio en este lenguaje.

Aunque aún no se han definido las características con las cuales se va a trabajar para tener un mejor dinamismo en la plataforma, sobre todo que la interacción sea la correcta, es preciso señalar como es que estará funcionando la parte de PHP y MySQL ya en el servidor web. En la figura 40 podemos observar que la extracción de datos se hace a nivel de SQL, mientras que pasan por el servidor web en el que estará alojado la plataforma y finalmente, llegan a la **presentación** en el navegador del cliente y es visto por ellos. Sin embargo, se pueden enviar distintas solicitudes por parte de los usuarios y el proceso continúa de la misma forma. En esta parte del proyecto se asume, por obvias razones, que el protocolo que define las reglas por las cuales es transportada la información, es el **Protocolo de Transferencia de Hipertexto** (*Hypertext Transfer Protocol* por sus siglas en inglés).

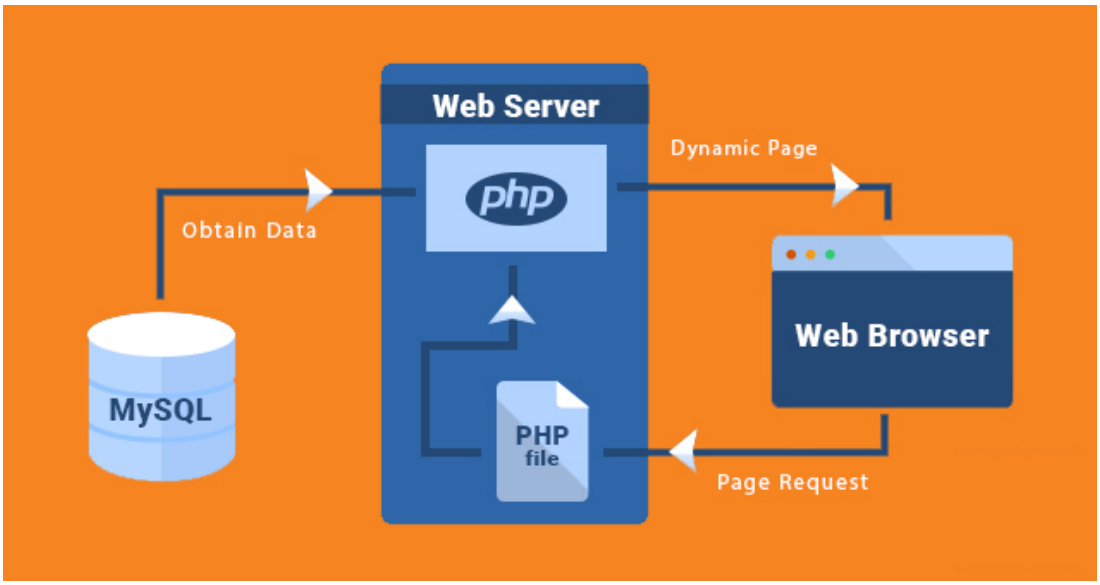


Figura 40: Estructura de envío de datos en el web server. Fuente: Nhi H. (2021, 9 octubre). MySQL server. Tino Group. https://wiki.tino.org/mysql-la-gi/

El proceso que se realiza más detalladamente contemplando las tecnologías con las que ya se ha estado trabajando hasta al momento, es decir **HTML, CSS y JavaScript** se muestra en la figura 41. El **protocolo** donde se lleva a cabo prácticamente todo la parte de las solicitudes del **cliente-servidor** es HTTP. Este protocolo, es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor.

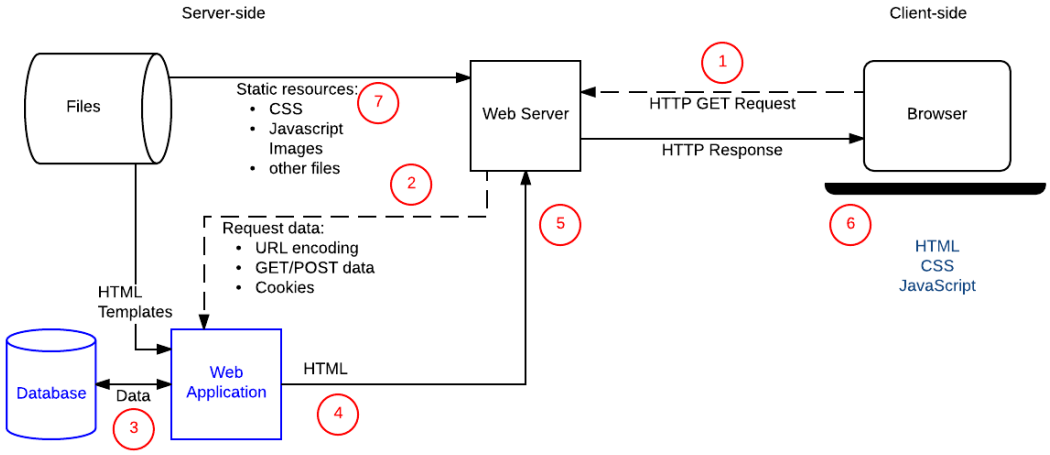


Figura 41: Proceso utilizando el protocolo HTTP. Fuente: Visión General Cliente-Servidor MDN. (2020). MDN Web Docs. https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview

El objetivo de mostrar este funcionamiento es solamente para tener en claro lo que se hará cuando un usuario de tipo **profesor, alumno o administrativo** accede a la plataforma y posteriormente interactúa con la base de datos. Todas las solicitudes que se hagan del lado del cliente son llevadas a cabo por un método desarrollado en todos los servidores web, este es conocido como **GET**. Este método reconoce las rutas de toda la plataforma (inclusive todas las que ya se han desarrollado en los anteriores capítulos), lanza la solicitud al servidor y este descarga todos los datos almacenados en la plataforma (archivos HTML, JavaScript, CSS, PHP, etc.). Existe otro método que se implementa siempre y cuando haya peticiones más estrictas al servidor, de hecho, este método es esencial que se use para poder registrar todos los datos en la base de datos, y es conocido como **POST**.

HTML, ya incluye dentro de sus etiquetas si se quiere implementar un método u otro cuando se hace una petición especial por parte del cliente. Esencialmente, esto se hace en formularios, registro de usuarios, en el propio inicio de sesión, entre otros. También, es necesario decir que, hay que establecer una manera en la que el servidor este comunicado con la base de datos sin que está recargue la página, esto para añadir más perfomance al proyecto, y esto solo se logra sincronizando el lenguaje de back-end con el de front-end, o JavaScript. Este método es implementado con una librería de JavaScript llamada **AJAX (Asynchronous JavaScript And XML)**, una forma de establecer comunicación entre servidores conectando las peticiones al mismo tiempo.

6.1. Base de datos y patrón de diseño MVC

Ya en la construcción de la base de datos, es necesario tener establecido cuáles van a ser las entradas de dicha base, es decir, que columnas tendrá cada tabla de la base de datos y cuál será la relación entre los datos de una y otra. Para esto se presenta el nombre de cada una de las tablas con sus respectivas columnas mismas que, complementan toda la base de datos del departamento.

- **Profesores**: id, nombre, apellido, correo y fecha de creación.
- **Alumnos**: id, nombre, apellido, número de cuenta, carrera, correo, fecha de creación.
- **Laboratorio**: id, nombre y clave.
- **Grupos**: id, grupo
- **Calificaciones (Relacional):** id, alumnoId, grupoId, laboratorioId, profesorId, calificación.
- **Sliders**: id, imagen, flyer, enlace, informativo.
- **Usuarios**: id, email, password, tipo, supervisor.

El modelo relacional es un ejemplo de modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos está estructurada en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo particular. Cada tipo de registro define un número fijo de campos o atributos. Las columnas de la tabla corresponden a los atributos de los tipos de registro. El modelo de datos relacionales es el modelo de datos más utilizado y la gran mayoría de los sistemas de bases de datos actuales se basan en el modelo relacional.

Es por ello que, cuando se haga toda la parte programable de la plataforma con **PHP** es que se implementa la programación orientada a objetos, precisamente para que los datos de los registros de la base de datos coincidan con los atributos de cada clase en el proyecto y de esa manera se extraigan los registros con su llave y valor correspondiente sin necesidad de crear más código innecesario.

```
-- -----------------------------------------------------
-- Table `telecomfi_database`.`alumnos`
-- -----------------------------------------------------
CREATE TABLE `telecomfi_database`.`alumnos` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nombreAlumno` VARCHAR(45) NULL DEFAULT NULL,
  `apellidoAlumno` VARCHAR(45) NULL DEFAULT NULL,
  `numeroCuenta` INT NULL DEFAULT NULL,
  `carrera` INT NULL DEFAULT NULL,
  `correoAlumno` VARCHAR(60) NULL DEFAULT NULL,
  `alumnoCreado` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `numeroCuenta_UNIQUE` (`numeroCuenta` ASC),
  UNIQUE INDEX `correoElectronico_UNIQUE` (`correoAlumno` ASC))

-- -----------------------------------------------------
-- Table `telecomfi_database`.`calificaciones`
-- -----------------------------------------------------
CREATE TABLE `telecomfi_database`.`calificaciones` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `alumnoId` INT NULL DEFAULT NULL,
  `laboratorioId` INT NULL DEFAULT NULL,
  `profesorSignedId` INT NULL DEFAULT NULL,
  `grupoId` INT NULL DEFAULT NULL,
  `calificacion` VARCHAR(2) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `alumnoId`
    FOREIGN KEY (`alumnoId`)
    REFERENCES `telecomfi_database`.`alumnos` (`id`),
  CONSTRAINT `grupoId`
    FOREIGN KEY (`grupoId`)
    REFERENCES `telecomfi_database`.`grupos` (`id`),
  CONSTRAINT `laboratorioId`
    FOREIGN KEY (`laboratorioId`)
    REFERENCES `telecomfi_database`.`laboratorios` (`id`),
  CONSTRAINT `profesorSignedId`
    FOREIGN KEY (`profesorSignedId`)
    REFERENCES `telecomfi_database`.`profesores` (`id`))
```

Figura 42: Creación de la tabla de calificaciones y alumnos.

En la figura 42 se muestra la creación de la tabla de alumnos y la de calificaciones. En este caso, la primera solo es para ejemplificar como se crearon las tablas primarias, y la segunda, para mostrar como la tabla de calificaciones une a todas las demás tablas respecto al **id**, de esta manera se podrán extraer los registros de cada alumno con todos sus datos de laboratorio, y viceversa para profesores ya una vez desarrollado el código con PHP, mostrando los resultados en HTML con los estilos establecidos en los primeros capítulos de este proyecto.

Dado que se utilizó MySQL Workbench para relacionar las tablas y ejecutar las consultas de SQL, se muestra a continuación un esquemático más sencillo de leer que muestra toda la base de datos del departamento:

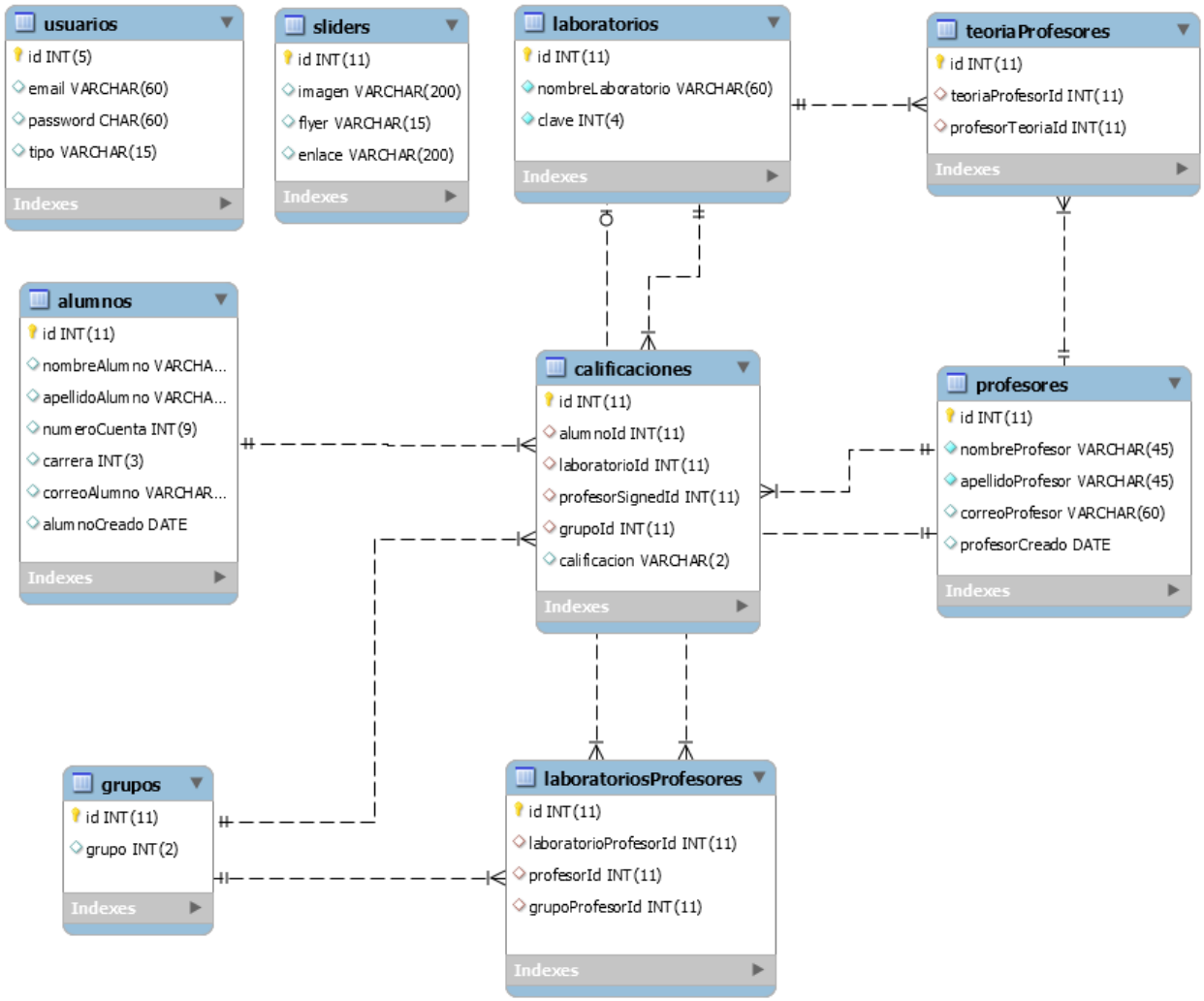


Figura 43: Diseño completo de la base de datos relacional de todo el departamento de telecomunicaciones.

En este punto del proyecto ya se tiene todo lo necesario para empezar a escribir código en el lenguaje de programación ya mencionado anteriormente. Dado que es un proyecto que conlleva muchas operaciones cliente/servidor, tener código que mezcle HTML, JavaScript, CSS y ahora PHP, sería simplemente un caos. La solución a ese problema es decretar un patrón de diseño, de los muchos que existen para el desarrollo de aplicaciones, que permita organizar mejor el proyecto. Esto además, beneficia al mantenimiento futuro que se tenga de la plataforma. Para ello, una vez realizada una búsqueda exhaustiva de todos los patrones de diseño que hay, se opto por usar MVC.

MVC - por sus siglas en inglés, Model - View - Controller (Modelo - Vista - Controlador) - es un patrón arquitectónico que nos permite desarrollar aplicaciones, manteniendo separada la lógica de negocios de las vistas, utilizando un ”controlador” como conector (o intermediario) entre ambas. En MVC, todo comienza con una petición del usuario. En este caso, puede ser el **profesor, el alumno o el administrador** de la plataforma.

En esta plataforma una petición podría ser ”Agregar un nuevo alumno” (del lado del administrador) o bien, ”calificar alumno X” (del lado del profesor):

- **¿Cómo realiza esta petición el usuario?** A través del navegador.
- **¿Cómo se identifica la petición?** Por medio de la URL ingresada por el usuario.

Este patrón trabaja con modelos, es decir, un modelo, en MVC, es una **clase**. Como patrón que guía nuestras arquitecturas, MVC nos obliga a escribir clases ”puras” para cada tipo de usuario o herramienta de la plataforma, que respeten el verdadero espíritu de la **programación orientada a objetos**. En este caso son varios los modelos que se construyeron debido a las necesidades del departamento, en la figura 45 se muestran cada una de ellas:

```php
class Modelo extends Main {
    public $nombreColumna;
    public function __construct($args = []){
        $this->nombreColumna = $args['nombreColumna'] ?? '';
    }
}
```

Figura 44: Construcción de un modelo en PHP.

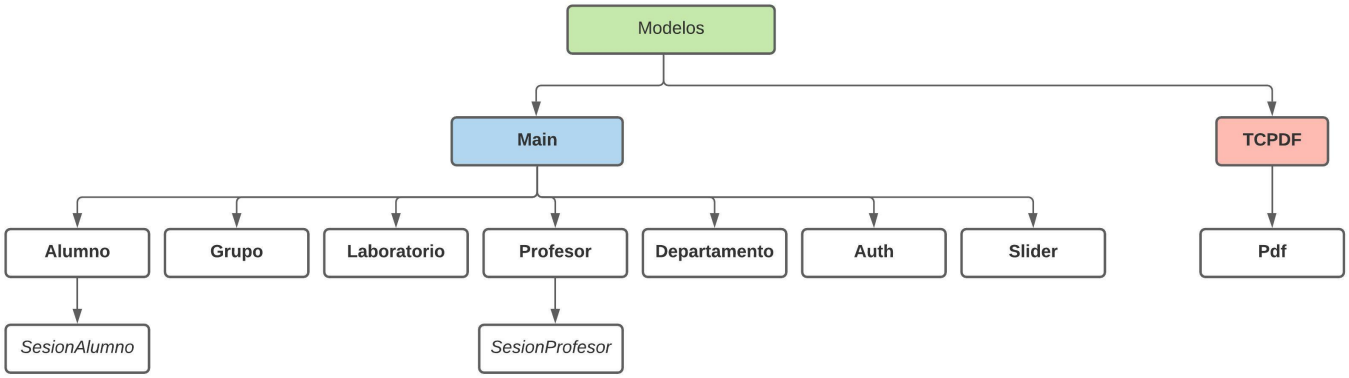


Figura 45: Modelos de la plataforma.

Los modelos están ideados conforme a las columnas en cada una de las tablas de la base de datos de este proyecto. Está es una recomendación que se hace en el diseño de aplicaciones y que conforma al patrón MVC. Es decir que cada **atributo** de cada clase o modelo contará con sus respectivas columnas conformadas en su tabla. Esto para que en el código realizado en PHP al momento de iterar en cada uno de los valores o datos de cada tabla pueda apuntar a su atributo extrayendo el valor correspondiente. Por lo que, en los modelos MVC se **escriben todas las consultas a SQL** necesarias para extraer datos de los profesores, alumnos o administrativos.

Gracias a que este proyecto se basa en POO, el modelo **Main** mostrado en la anterior figura pasa a ser la clase padre para las que están debajo de ella. Otra característica es que, en todos los demás modelos se ocuparán métodos generales como **extraer todos los datos, sanitizar valores, hacer la consulta con SQL y crear un objeto de la misma clase**, de otra forma había que crear cada una de las anteriores funciones mencionadas en cada modelo, lo cual es ineficiente en términos de tiempo y espacio de código.

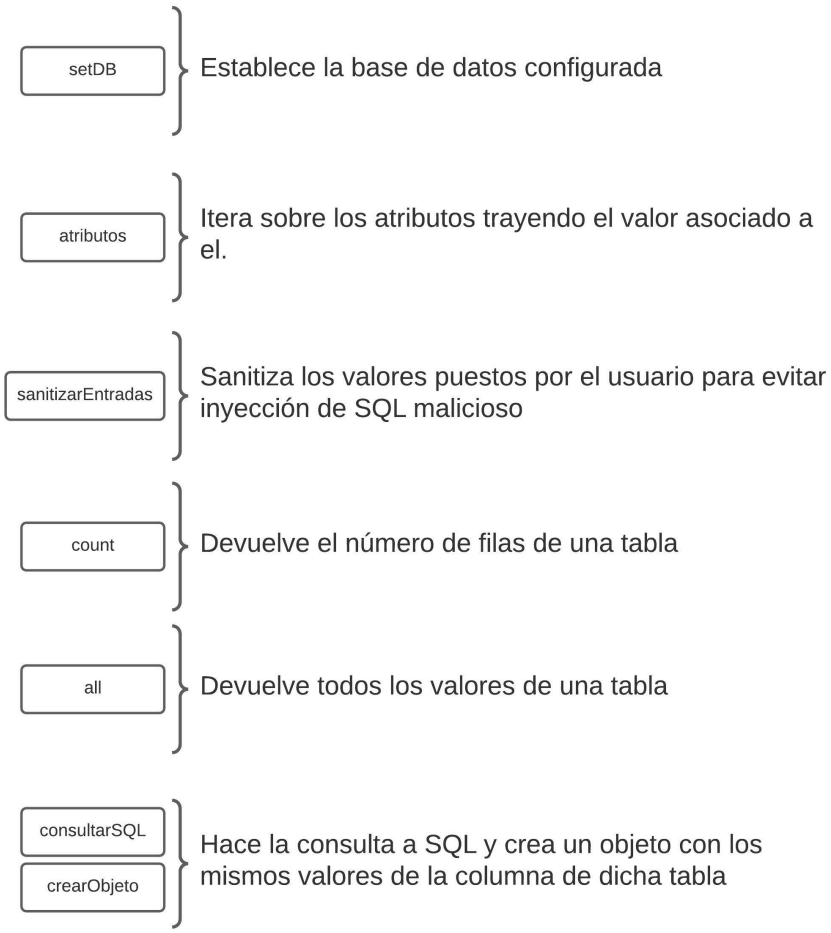


Figura 46: Funciones escritas en la clase/modelo: Main y que heredan todos los demás modelos.

Figura 47: Representación del modelo en PHP y su inherencia con la base de datos. Los atributos del modelo/-clase son las columnas de la tabla en la base de datos.

La primera parte del modelo MVC ya está realizada hasta este punto. Inclusive se podría indicar que la parte de las vistas también, porque de alguna manera en el capítulo "*Diseño Front-end con HTML5, CSS3 y JavaScript*" se realizó con las hojas de estilo correspondientes, archivos de marcado HTML y la parte dinámica con JavaScript. Por lo que, la última parte que es la del controlador faltaría definir. Aunque pareciese algo trivial, es la parte esencial y complicada de este modelo. En este sentido, es necesario poner en práctica ciertos conceptos de telecomunicaciones en el área de redes para entender este paradigma y aplicarlo correctamente.

Si bien, la palabra '*controlador*' hace referencia a aquello que toma las reglas, los requerimientos suficientes para que una cosa u otra pueda aplicarse, en el sentido del desarrollo de aplicaciones va más allá. No es solo el controlador el que decide que **vista** o que **modelo** se aplica dependiendo la **solicitud** que el usuario haya aplicado en ese momento, sino también un *intermediario*, que en el área de redes, específicamente hablando, se conoce como un **router** en un sistema de comunicación para que se pueda establecer una especie de saludo (o handshake) entre el modelo o vista y el controlador.

Imaginar que este modelo es una pequeña red que sincroniza lo que el usuario pide, se muestra en su navegador la página al cual se esta accediendo, pero también, están las URL's **registradas** en toda la plataforma que apuntan hacia cierta vista y que es visualizada en el navegador web. Estas URL's están dentro de todo el modelo y haciendo una inherencia con redes de telecomunicaciones, vendría siendo una **tabla de enrutamiento** que si bien, podríamos decir que es estática, es esta tabla la que especifica si la URL existe dentro de la plataforma, si lo es, enviará la petición realizada al controlador y finalmente, se encargará de hacer la etapa de decisión de que vista y modelo se ejecutará; de lo contrario, simplemente se envía un mensaje de error muy común y visto muchas veces, el error **404**.

A manera de explicar lo anterior con un diagrama, se muestra la figura 48, donde se puede ver claramente lo mencionado y además los casos de error o éxito si se lleva a cabo la petición HTTP. El router será un **archivo PHP exclusivo** que almacenará no sólo las rutas que todo el público puede ver, digamos si alguien accede a la plataforma una vez subida a un servidor, accede a través de el DNS asignado pero, no cualquiera puede ver la página de inicio de sesión de un alumno o profesor. Esto es muy importante, porque se debe cuidar la **integridad de datos** de todos los académicos, alumnos y administrativos que están en la base de datos, sino se hace de manera adecuada podría sufrirse algún ataque (Brute Force, DoS por sus siglas en inglés - Denial-of-Service, etc).

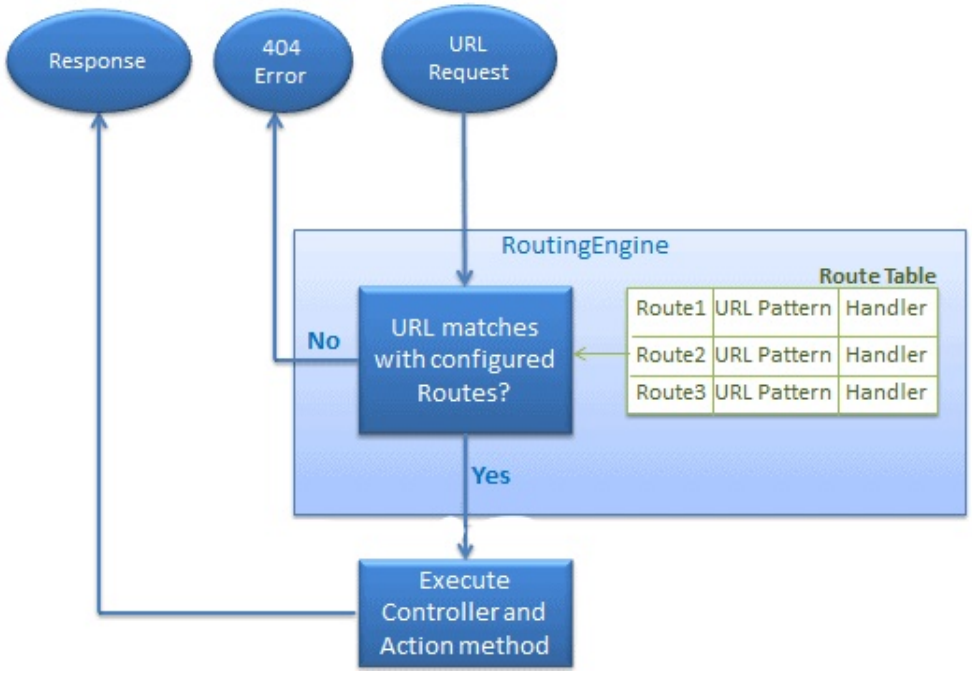


Figura 48: Consulta a las tablas de las URL's a través del método GET en un sitio web. Fuente: D. (2018, 5 octubre). Routing in MVC. Developer Corner Blog. http://dheerajpatial.blogspot.com/2018/09/routing-in-mvc.html

Como ya se había comentado, la parte de los modelos lleva a cabo toda la lógica de SQL que extrae los datos, las vistas representan todo lo que se ve en la interfaz de usuario o en el navegador web y en el controlador es donde se hace una petición a las clases construidas anteriormente ligadas a una cierta página de PHP asociada con su lenguaje HTML y sus propios estilos (vistas) para que se pueda completar de manera íntegra toda la plataforma.

En el diseño de proyectos que conllevan programación, es muy usual representar las clases, que en este caso son los modelos, en un diagrama conocido como **Lenguaje Unificado de Modelado - UML**, de esta forma quedará claro lo que se programará y como estará constituida toda la parte central del proyecto. En este sentido, el diagrama completo es mostrado en la figura 49 para representar lo antes expuesto.

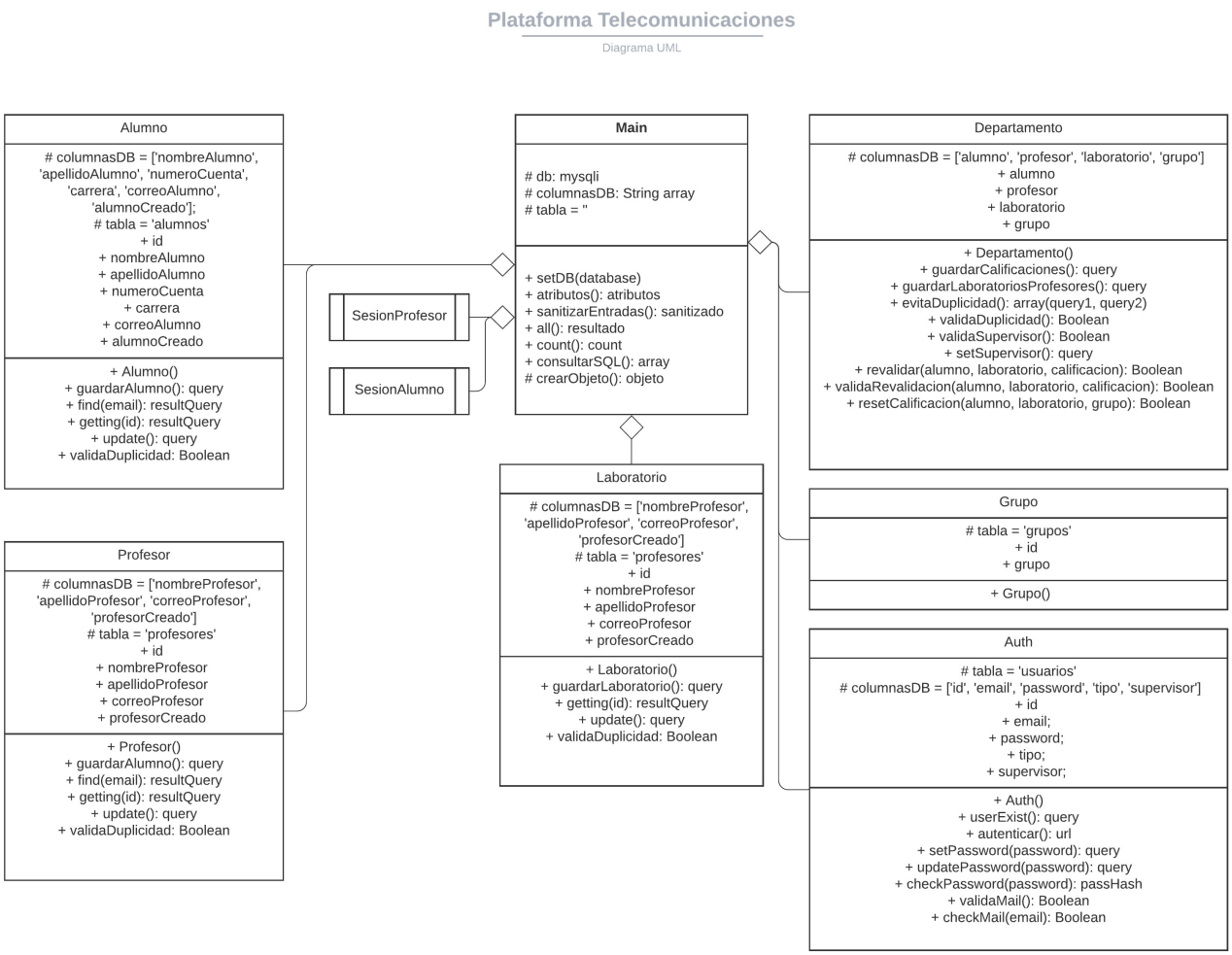


Figura 49: Diagrama de clases UML.

Sin embargo, no todas las clases o modelos se tienen que construir desde cero, afortunadamente existen pequeños modelos escritos por otras personas en internet que proporcionan características adicionales que se desean añadir a un sitio web. Algunas cosas extras que se requirieron por el departamento fueron las siguientes:

- Envío de contraseñas a los profesores y alumnos a través de correo electrónico.
- Generador de reportes PDF para los profesores de los laboratorios calificados.

Para la primera parte se ocupó una clase llamada **PHP Mailer** que se encarga precisamente de hacer envíos de mensajes vía correo electrónico, o bien, a través del **protocolo SMTP** (por sus siglas en inglés - Simple Mail Transfer Protocol), previamente configurado. La segunda parte fue solucionada con otra clase llamada **TCPDF**, esta es un poco distinta porque aquí se tiene que diseñar una plantilla PDF a través de coordenadas programadas en PHP, aunque pudiera sonar complicado, la realidad es que el detalle está en colocar correctamente los títulos y datos del reporte, ya que esta clase provee toda la configuración para generar el archivo .pdf automáticamente.

Finalmente, se construye el archivo **router.php**, contenedor de todas las URL's de la plataforma. En el arreglo llamado *protectedRoutes* están todos aquellos enlaces en los cuáles los usuarios públicos o externos no podrán acceder.

```php
class Router {

    public $routesGET = [];
    public $routesPOST = [];

    public function get($url, $fn){
        $this->routesGET[$url] = $fn;
    }

    public function post($url, $fn){
        $this->routesPOST[$url] = $fn;
    }

    public function verifyingRoutes(){

        session_start();
        $auth = $_SESSION['login'] ?? null;
        $protectedRoutes = ['...'];
        $url = $_SERVER['REQUEST_URI'];
        $data = explode("?", $url);

        $currentURL = $data[0] ?? '/';
        $method = $_SERVER['REQUEST_METHOD'];

        if($method === 'GET'){
            $fn = $this->routesGET[$currentURL] ?? null;
        } else{
            $fn = $this->routesPOST[$currentURL] ?? null;
        }

        if(in_array($currentURL, $protectedRoutes) && !$auth){
            header('Location: /');
        }
        if($fn){
            // URL existe y tiene una función asociada
            call_user_func($fn, $this);
        } else {
            header('Location: /');
            //echo "Página no encontrada.";
        }
    }
}
```

Figura 50: Código para el router de la plataforma

6.2. Envío de contraseñas a través del protocolo SMTP

Un punto importante que en las secciones anteriores se había discutido era que, la integridad de la base de datos debía ser segura. Implementar algoritmos de encriptación no es una cuestión tan fácil, sin embargo esta es una de las razones por las cuales también se escogió programar la plataforma con PHP. Una de las ventajas de usarlo es que provee dentro de sus funciones algunas formas de encriptación (funciones digestoras) para contraseñas seguras, como el caso de **BLOWFISH**, **MD5**, **SHA256**, **SHA512**, etc.

No obstante, utilizar **MD5** que por mucho tiempo fue el encargado de hashear contraseñas para aplicaciones web, de escritorio o móviles, hoy en día posee muchas vulnerabilidades. Una de las más cómodas, sencillas y bastante seguras de usar es el algoritmo implementado con **BLOWFISH**. Blowfish es un cifrado en bloque de 64 bits con una clave de longitud variable. El algoritmo consta de dos partes: expansión de claves y cifrado de datos. La expansión de claves convierte una clave de hasta 448 bits en varias matrices de subclaves por un total de 4168 bytes. **PASSWORD_BCRYPT** en PHP usa el algoritmo para crear el hash. Producirá un hash estándar compatible con *crypt()* utilizando el identificador "$2y$". El resultado siempre será un string de 60 caracteres, o false en caso de error.

```php
/* Encriptar el password proveniente de texto plano */
$passwordHash = password_hash($password, PASSWORD_BCRYPT);
/* Desencriptación */
$autenticado = password_verify($password, $usuario->password);
```

Figura 51: Encriptación y desencriptación de contraseñas con BCRYPT en PHP.

Una vez realizada la parte de encriptación de contraseñas es importante señalar que la cadena de caracteres generada por la función mencionada anteriormente de PHP se almacena en el campo "**password**" de la tabla de *usuarios*. De esta forma es imposible que alguien que administre la página y la base de datos en un futuro sepa la contraseña de cada uno de los profesores y/o alumnos registrados.

id	email	password	tipo	supervisor
1	[redacted]8@gmail.com	$2y$10$pSRdYWI0SlCZUECvI9/0X.GQWzdwX....	administrador	NULL
2	[redacted]:@gmail.com	$2y$10$4R0VLL8aEqycBmpCOaFhL.wnP3X8JKX...	profesor	accept
3	[redacted]98@gmail.com	$2y$10$jS41CaVa1/Y9kbsv9kKdQe8kQ7NZJYN...	alumno	NULL
4	[redacted]@hotmail.com	$2y$10$1tvecpCHGLNnLQCfDrNPh.oUNmNAZV...	profesor	NULL
5	[redacted]@hotmail.com	$2y$10$JnP3UKimwUZPHSa5gFOJnOgtinlB1aK7...	profesor	NULL

Figura 52: Contraseñas almacenadas en la tabla de usuarios.

Ahora bien, como parte del sistema de inicio de sesión es imprescindible que tanto profesores, alumnos y administrativos sepan la contraseña generada (sin encriptar). Para ello, es necesario que antes de que se pase por la función de encriptación de PHP, habrá una función que generará una cadena de 8 caracteres de longitud de forma aleatoria y ésta es la que será enviada por medio de correo electrónico (SMTP). Otro punto importante, la función que genera una cadena de longitud 8 puede recibir como parámetro precisamente el cambio de esta longitud si es que en un futuro se quisiera cambiar por una más grande o corta, recordando que entre mayor sea la longitud, menor será la probabilidad de que se repita. Aunque, si se llegase a repetir la cadena una vez que es pasada por la función **HASH** de PHP es prácticamente imposible que la cadena encriptada sea la misma que la de otro usuario.

```php
function generatePassword($length = 8){
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }

    return $randomString;
}
```

Figura 53: Función generadora de una contraseña segura y aleatoria.

SMTP, definido en *RFC 5321*, es el núcleo del correo electrónico de Internet. SMTP transfiere mensajes de los servidores de correo de los remitentes a los servidores de correo de los destinatarios. Para ilustrar mejor el funcionamiento de este protocolo, se muestra un diagrama en un escenario común de la literatura donde Alice quiere enviar un mensaje a Bob. En este caso, Alice vendría a figurar el servidor de hosting donde quedará almacenada la platforma y Bob serán cada uno de los usuarios a los cuales se les hará llegar su contraseña por esta vía:

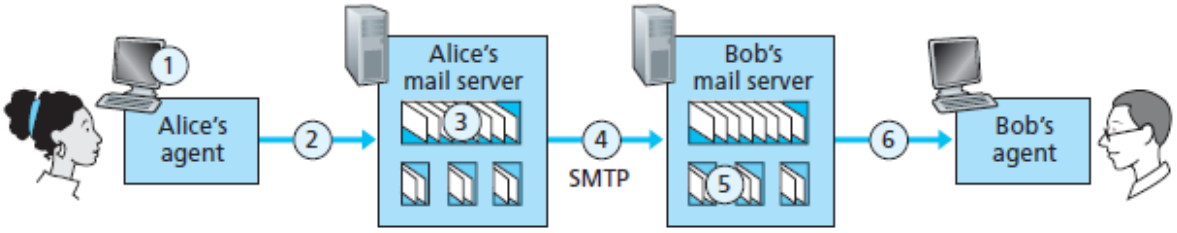


Figura 54: Protocolo SMTP en funcionamiento con el paradigma Alice-Bob. Fuente: Ross, I. (2013). SMTP. 6th ed., Computer Networking: A Top-Down Approach (6.a ed., pp. 121–124). Pearson.

Como se comentó anteriormente, no todos los modelos son escritos cuando se construye una aplicación, y para lograr el objetivo de envío de emails a través del protocolo ya explicado se utilizó la clase **PHP Mailer**. Este modelo ya proporciona los requerimientos básicos de la configuración para el envío exitoso de correos electrónicos, además proporciona una configuración de cifrado por TLS o SSL, según se requiera. En las especificaciones de PHP Mailer se requiere lo siguiente:

- **SMTPSecure:** TLS
- **HOST**: smtp.gmail.com
- **PORT**: 587

Dado que el envío de correos electrónicos requiere de un remitente, se especifica que el HOST por el cual se abre esta conexión a nivel de capa 4 del modelo OSI, específicamente de **TCP**, es a través del túnel de **GMAIL** dado que, el correo del departamento posee este servicio de correo. Si en algún momento se cambia este último parámetro, tendría que apuntar hacia un HOST distinto al de GMAIL, pudiendo ser cualquier otro servicio de correo electrónico. A pesar de considerar el HOST, otra configuración a destacar es el rango de puertos que se eligen cuando se envían correos por internet.

Existen algunos puertos muy comunes, como el puerto **25, 587, 465, o 2525**. Por ejemplo, el 25, el puerto SMTP estándar para mover mensajes entre servidores de correo, es a menudo bloqueado por los ISP y los proveedores de nube (incluyendo Google Cloud Platform, que es lo que utiliza Kinsta). Sin embargo, para **PHP Mailer** el funcionamiento correcto solo especifica el 587 y 465, eligiendo el primero para su configuración.

En última instancia, el cuerpo del mensaje incluirá la cadena que arrojará la función ya antes mencionada, además de algunos títulos y párrafos de bienvenida a la plataforma escritos con lenguaje de HTML. Quien será el encargado de hacerle llegar el password a cada usuario será el **administrador** a través de su particular inicio de sesión. Otra característica válida es que, si el correo electrónico no se encuentra en la base de datos, para evitar spam a usuarios que no pertenezcan, se hará una pequeña validación, rectificando el correo electrónico asociado a algún alumno o profesor.

6.3. Login de usuarios

Hasta este momento ya se tienen todos los datos registrados de cada tipo de usuario, esto es: **alumno, profesor o administrador**. Como parte del diseño de la plataforma es indispensable que estos usuarios inicien sesión dependiendo los datos y dichas validaciones que se harán posteriormente dentro del servidor.

Algunas validaciones importantes es que los usuarios públicos que navegan dentro de la plataforma no puedan ver ningún URL por medio del inicio de sesión. Muchos desarrolladores anteriormente no confiaban en un lenguaje como PHP debido a este factor atenuante. PHP si es un buen lenguaje en materia de seguridad, siempre y cuando se sepa implementar de manera correcta; el patrón de diseño MVC dentro de sus ya tantas características mencionadas anteriormente, es separar la lógica de las vistas públicas y privadas, de tal suerte que, cuando un usuario normal visita la página y muy audaz descubre alguna de las URLs con las cuales se inicia sesión, estas prevalecerán protegidas.

Si bien, podría pensarse que si un usuario común ingresa la URL en el navegador, suponga, de los profesores, el servidor marcaría muchos errores de PHP debido a que los datos no han sido emparejados. Estos errores revelan información importante de las rutas de los archivos contenidos en el proyecto, lo cual se traduce en una revelación importante que podría provocar o darle entrada a ataques de sesión o hackeos. Es por ello que, al proteger las rutas con otro modelo exclusivo para este objetivo, es que podemos evitar estas vulnerabilidades.

Como se ve en el pequeño fragmento de código de la figura 55, algo tan sencillo como identificar el tipo de sesión auxiliándose de la variable global de PHP **$__SESSION**, podemos evitar a personas que no estén en la base de datos o quieran acceder por medio de peticiones tipo GET a las URL, sean denegadas o redirigidas a la página principal de la plataforma. Las referencias hacia las URL se envían en formato **JSON** para que sean entendidas por el lenguaje de JavaScript y pueda mostrarle las correspondientes vistas ya antes mostradas en este proyecto al tipo de usuario.

```
if($_SESSION['tipo'] === 'alumno'){
    $ref = ["Location"=>"/alumno/sesion-alumno"];
} else if ($_SESSION['tipo'] === 'profesor') {
    $ref = ["Location"=>"/profesor/sesion-profesor"];
} else if ($_SESSION['tipo'] === 'administrador') {
    $ref = ["Location"=>"/admin"];
} else {
    $ref = ["Location"=>"/"];
}
```

Figura 55: Reconocimiento del tipo de sesión.

Aunque se restringe el acceso por medio de las anteriores condiciones, es importante mantener las rutas protegidas. En el patrón MVC se comentó que, existe una especie de tabla de enrutamiento que almacena todas las URL's de la plataforma. Justamente, en este modelo llamado **Router** es donde separamos las rutas privadas de las públicas y de esta manera se almacenan en un colección de datos protegidos. Para esquematizar esta parte, que es realmente la esencia del inicio de sesión implementando el patrón ya mencionado, se ilustra lo siguiente a detalle, en donde la petición empieza por solicitar la vista, ésta a su vez pregunta al controlador para que resuelva y finalmente, si existe algún modelo asociado con peticiones SQL, se mandará a llamar dicha clase. En este caso, no se ilustra el **router**, este es solo un intermediario entre el controlador y los modelos para saber a quien darle acceso a ciertas páginas de la plataforma.

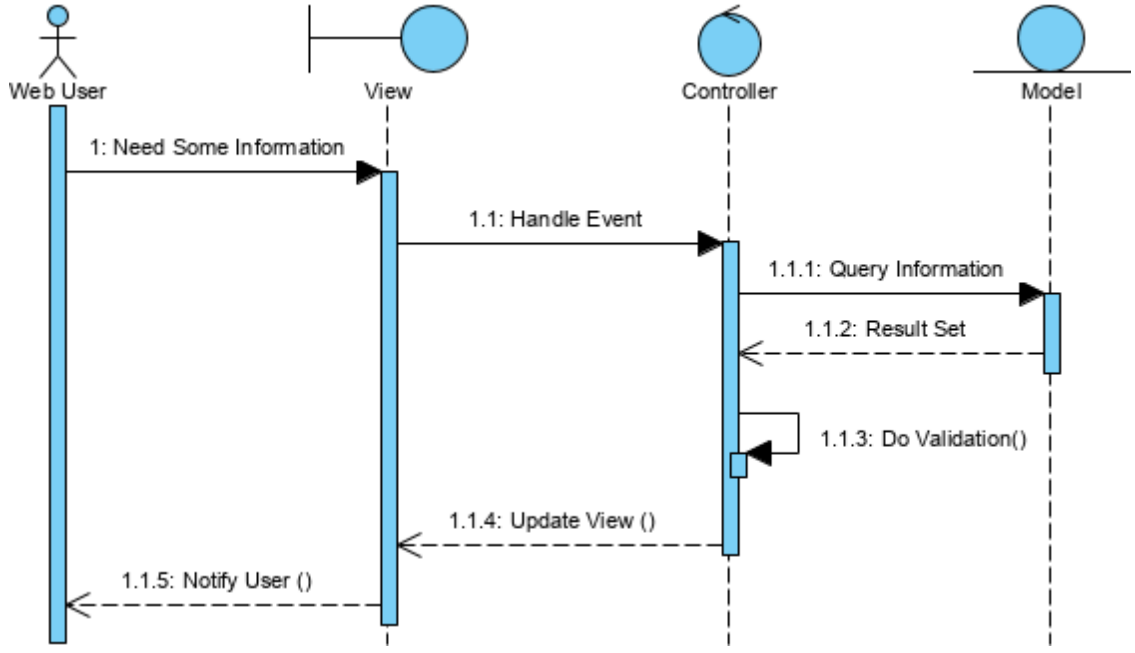


Figura 56: Esquema de inicio de sesión en patrón de diseño MVC. Fuente: What is Model-View and Control? (2019). Visual Paradigm. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-model-view-control-mvc/

6.4. Conexión de la base de datos con el servidor

La parte final de este proyecto viene dada por la conexión a la base de datos. Todas las pruebas que se hicieron con datos no reales se hicieron en el **localhost** de tal manera que, cualquier error de tipo back-end que se presentara en todas las posibles funciones ya desarrolladas, es aquí donde se solucionaron.

Es muy importante antes de lanzar el sitio o plataforma a un host o servidor en la nube, tener seguridad de que no habrá errores principalmente en el inicio de sesión de los usuarios. Para ello, se hace una replica de la base de datos que estará realmente alojada en Azure o AWS en la computadora de prueba a través de la dirección IP local, es decir al localhost.

Dado que, algunas clases de PHP que ya estan preestablecidas en el lenguaje, utilizan el paradigma orientado a objetos, se utilizó una de ellas para hacer la conexión con el gestor MySQL. Esta clase se llama *MySQLi*, permite hacer la conexión especificando los parámetros necesarios de la base de datos como: **host, username, password, name_database**, prácticamente lo mismo que cuando se hizo la conexión con el protocolo SMTP a través de PHPMailer. Finalmente, se estableció el método de manera general, es decir, en un archivo aparte (no requiere un modelo), para que allí resida la configuración general de la base de datos.

```php
function conectarDB() : mysqli {
    $db = new mysqli('host', 'username', 'password', 'nameDatabase');

    if(!$db){
        echo "Error no se pudo conectar a la base de datos.";

        exit;
    }

    return $db;
}
```

Figura 57: Configuración de la conexión de la base de datos con PHP y MySQLi

Con lo anterior establecido y una vez hechas las pruebas correspondientes en los 3 usuarios: **administrador, profesor y alumno**, se procede a la investigación y configuración del servidor que alojará la plataforma. Estos servicios, proporcionan diferentes características dependiendo del almacenamiento, procesamiento del servidor, etc. Dentr de los modelos de administración de la nube, se encuentran tres grandes divisiones:

- IaaS (Infrastructure as a Service)
- PaaS (Platform as a Service)
- SaaS (Software as a Service)

De las tres anteriores, la que más conviene en este tipo de escenarios y a fines de este proyecto es la segunda opción, PaaS. Este modelo se enfoca a brindar servicios en la nube para la configuración de almacenamiento y de base de datos. En el siguiente capítulo, se explicará a detalle la configuración que se realizó en cada uno de estos servicios, enfocado a alojar la plataforma ya en el ambiente global de internet.

7. Servidores de alojamiento web

En este punto se ha llegado casi a la finalización del proyecto. Sin embargo, es necesario aún hacer más pruebas para asegurarse de que los usuarios ven correctamente toda la información antes expuesta. Para lograr este objetivo, hay que configurar correctamente la plataforma de Azure, crear un servidor y administrar las direcciones IP que se requieren para autorizar el acceso solo a administradores.

Primero, lo que se hizo fue crear un servidor en la nube (virtualizado) proporcionado por Microsoft para almacenar la base de datos que anteriormente se ha implementado.

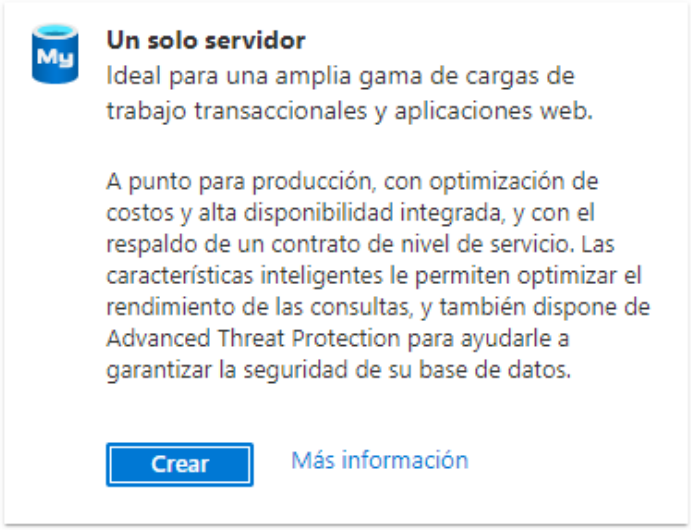


Figura 58: Tipo de servidor a crear en Azure.

Dado que este será la primer prueba de la plataforma ya a nivel público, con 1 servidor basta. Posteriormente, es necesario establecer el nombre, ubicación, almacenamiento del servidor:

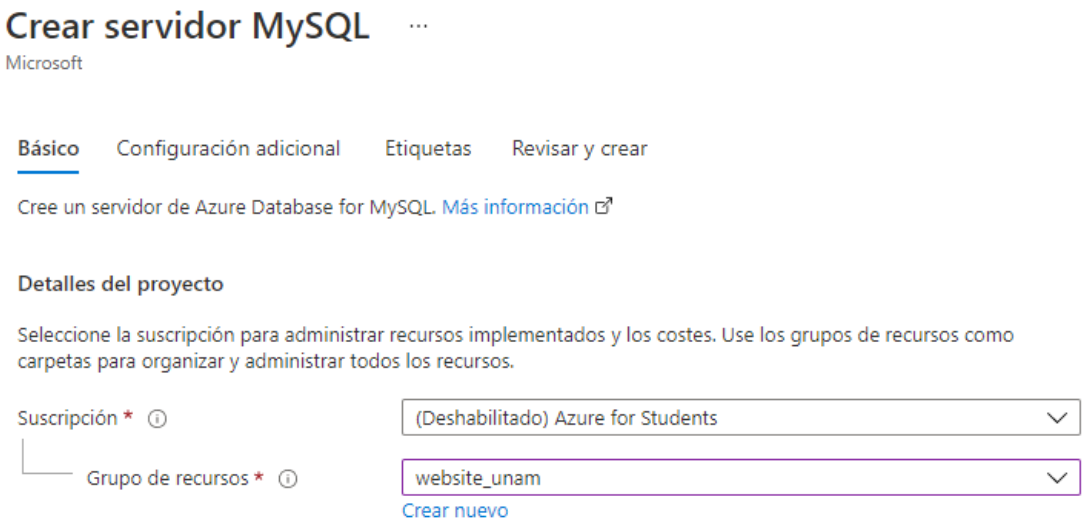


Figura 59: Estableciendo parámetros básicos del servidor.

Algunos detalles más, como la localización virtual de este servidor y el tipo de almacenamiento (al añadir más datos adicionales como copias de seguridad y más núcleos virtuales, se añaden costos extras fuera del crédito estudiantil):

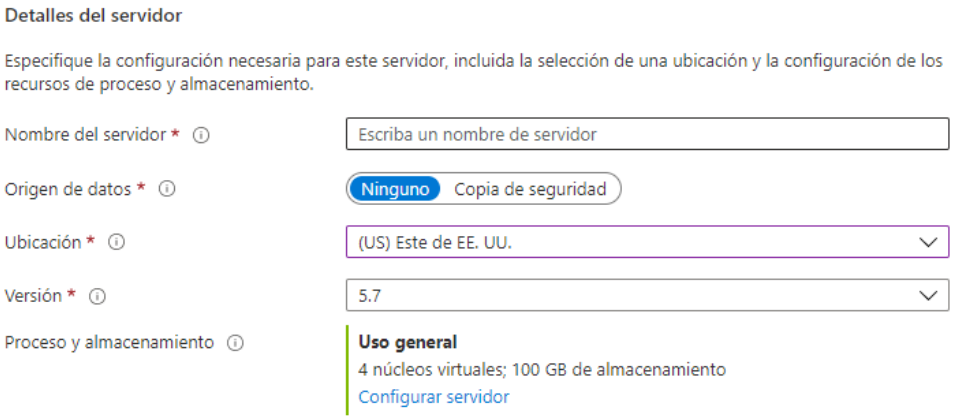


Figura 60: Detalles del servidor.

Finalmente, las credenciales del administrador del servidor:

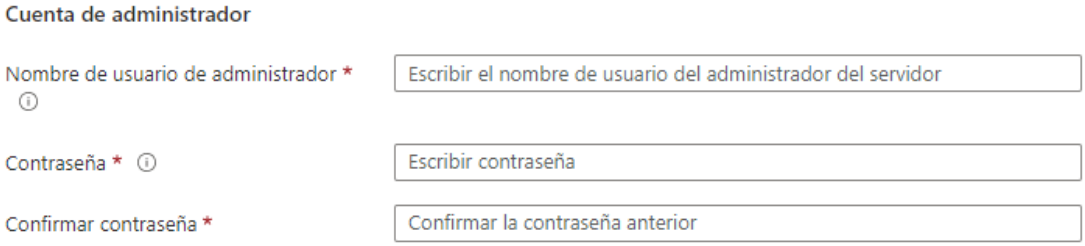


Figura 61: Credenciales de autenticación.

Una vez creado el servidor, ya se puede especificar el tipo de recurso que almacenará, en este caso será una base de datos del gestor **MySQL** que es donde está originalmente implementada. Para ello, se crea la base de datos en Azure:

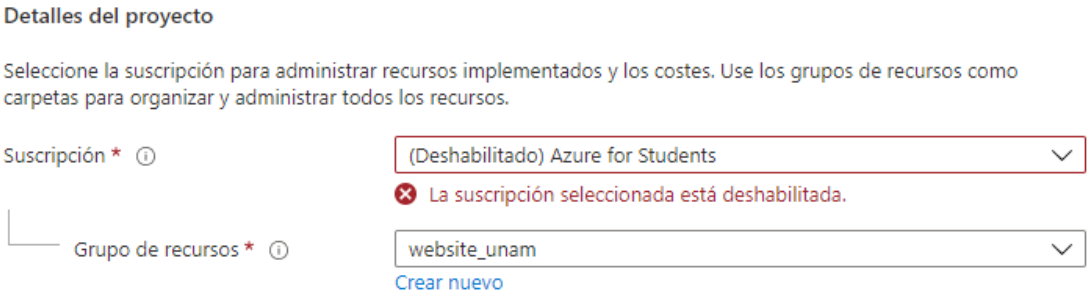


Figura 62: Configuración básica de la base de datos, especificando el recurso utilizado.

Como ya se ha creado el servidor, se especifica la base de datos que estará alojada en la plataforma:

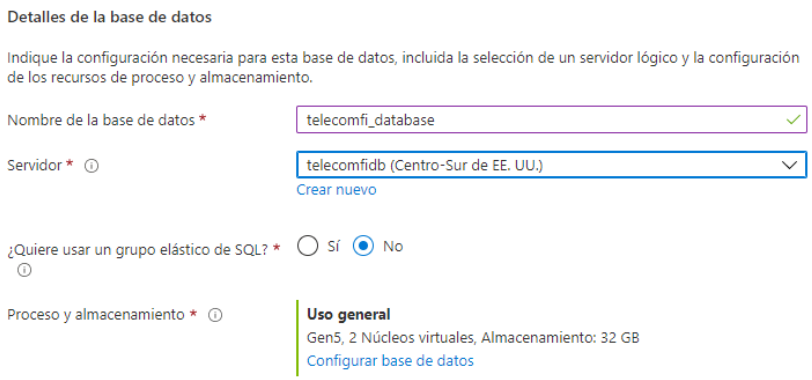


Figura 63: Especificando el servidor relacionado con la base de datos.

A diferencia de el servidor también se requiere un almacenamiento, en este caso el gratuito proporcionado por Azure son de 32 GB, lo cual son perfectamente adaptados para la primer prueba de la plataforma, sin pagar costos extras.

Dado que Azure cuenta con firewalls virtuales, lo que quiere decir que no todas las personas acceden a la zona protegida por estos dispositivos lógicos, hay que configurar que direcciones IP están permitidas y cuáles no, para evitar que la información baje de manera insegura hasta la plataforma.

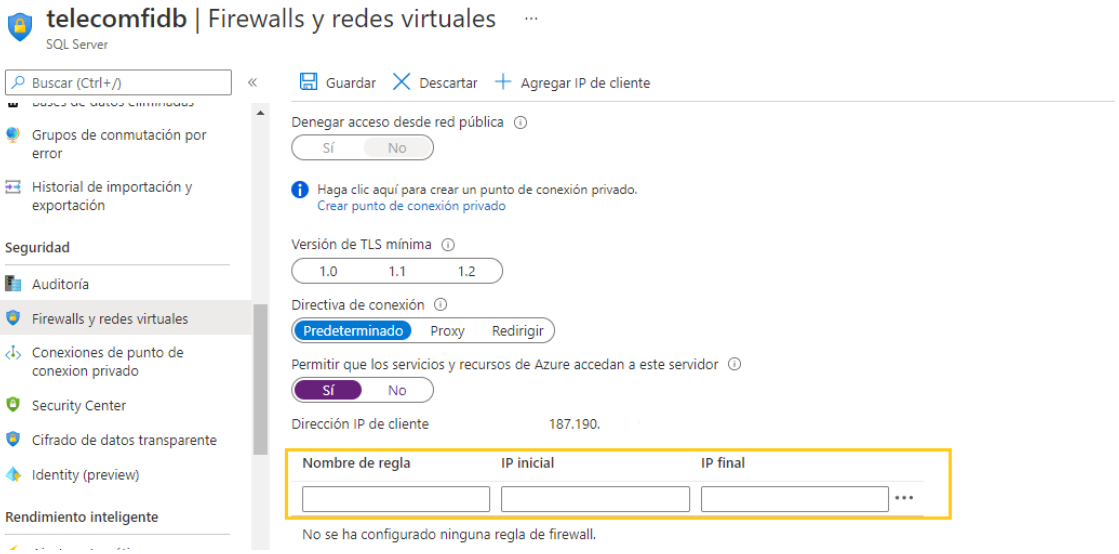


Figura 64: Añadiendo y configurando las direcciones IP permitidas para el servidor.

En los campos de registro de IPs es donde se irán llenando. Un aspecto importante que aclarar es la forma en la que se accede al servidor. Una cosa es que los usuarios entren y registren una dirección IP pública asignada por su ISP (*Internet Servicer Provider*) del lado del cliente, y otra aquellas que tienen privilegios de poder configurar, editar y añadirle elementos al servidor creado. Esta última parte es la que se configuró con la dirección IP ,en este caso la propia, y la de los administrativos correspondientes.

Otro punto a destacar es la resolución de la dirección IP y el servidor DNS de la plataforma de Heroku App (que posteriormente se explicará), si no se activa esta dirección una vez generada, el firewall de Azure se verá en la necesidad de bloquear la ruta y no dejará visualizar nada. Por eso, es muy importante configurar esta parte, que si no se tiene en cuenta puede ocasionar problemas de funcionamiento o en el peor de los casos no se podrá establecer ninguna conexión a la base de datos.

Finalmente, una vez llevado a cabo toda la configuración del servidor, y una vez creado la base de datos con MySQL y Azure, éste proporcionará los requisitos de host, password, nombre de la base de datos que son esenciales para poder hacer la conexión con PHP, como fue expuesto en el código de la figura 57.

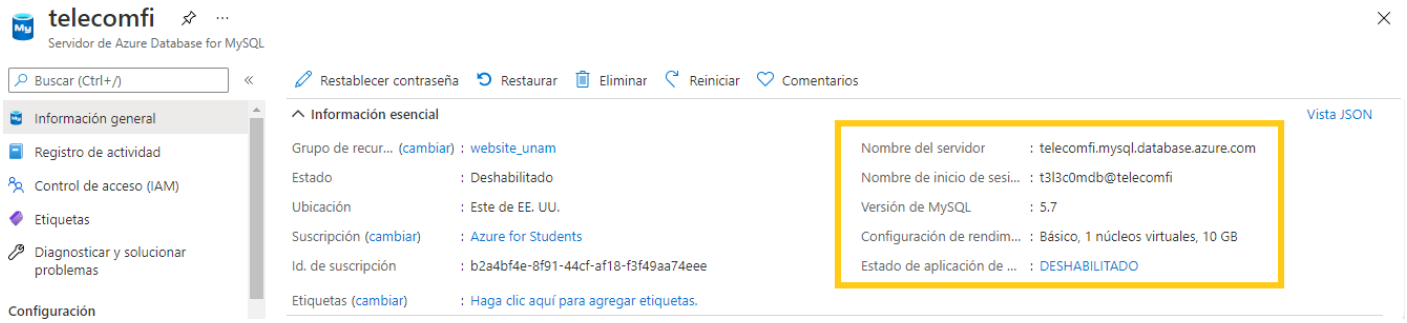


Figura 65: Datos de conexión al gestor de MySQL en Azure.

El dato que se pone en el host para PHP es el **nombre del servidor** y como nombre de usuario el que está justo debajo de esa información. El password anteriormente se especificó y por fines de seguridad Azure no la muestra. Con estos datos es posible hacer la conexión ya con la plataforma y la base de datos, teniendo en cuenta que el establecimiento de los parámetros en el servidor creado en este caso, de manera virtual por la nube de Azure, debe ser bien instaurado de tal suerte que, se tengan resultados esperados como lo fue en las pruebas con el **localhost**.

Como siguiente parte en el desarrollo de este proyecto, es indispensable subir la plataforma con toda la carpeta del proyecto y programación de **PHP** necesaria para su funcionamiento a un servidor de alojamiento; en este caso por fines de costos, se eligió Heroku App, este servicio recae en el nivel de PaaS, dado que los servicios que proporciona son a nivel desarrollador. Heroku proporcionará la configuración automática para establecer un servidor DNS y dar un dominio al sitio web.

Como bien se mencionó anteriormente, este servicio de alojamiento provee el DNS público en este caso, de manera gratuita ya que adicionalmente puede haber costos extras por un dominio customizado. Heroku funciona con el servidor muy popular en el área de aplicaciones web conocido como **Apache**.

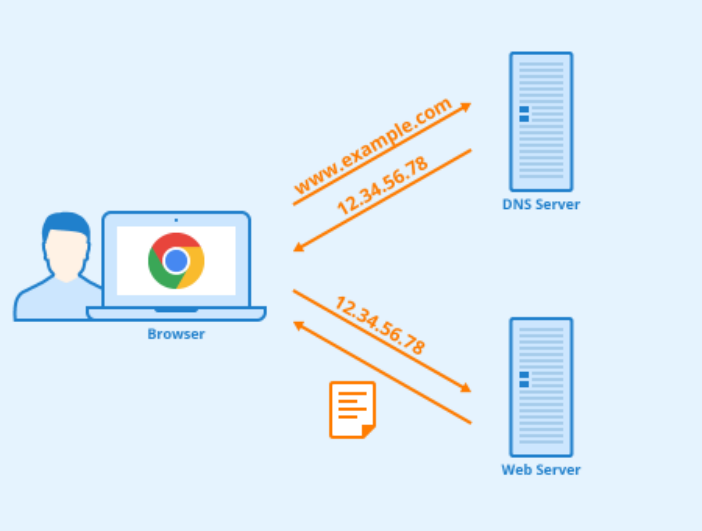


Figura 66: Proceso de una petición DNS desde un browser. Fuente: Salom, J. (2021, 8 marzo). Cómo filtrar contenido no apropiado mediante el servicio DNS. La mar de seguros. https://lamardeseguros.com/controlpatental/como-filtrar-contenido-no-apropiado-mediante-el-servicio-dns/

La característica del servicio gratuito de Heroku viene dada por que al dominio le añaden parte del nombre de su servicio, debido a la gratuidad del mismo:

www.example.herokuapp.com

Por lo cual, esta es una desventaja, pero por temas de costos es necesario dejarlo implementado con el paquete gratuito proporcionado por Heroku. Algunas de sus ventajas más importantes es que permite poner el sitio web en modo mantenimiento, y proporciona la configuración automática con algún controlador de versiones como lo puede ser **GitHub**, de esta manera cada vez que se haga un cambio en la web de cualquier tipo, se actualice a través de comandos del controlador y estos sean subidos a Heroku automáticamente sin tener que subir los archivos que fueron modificados uno por uno.

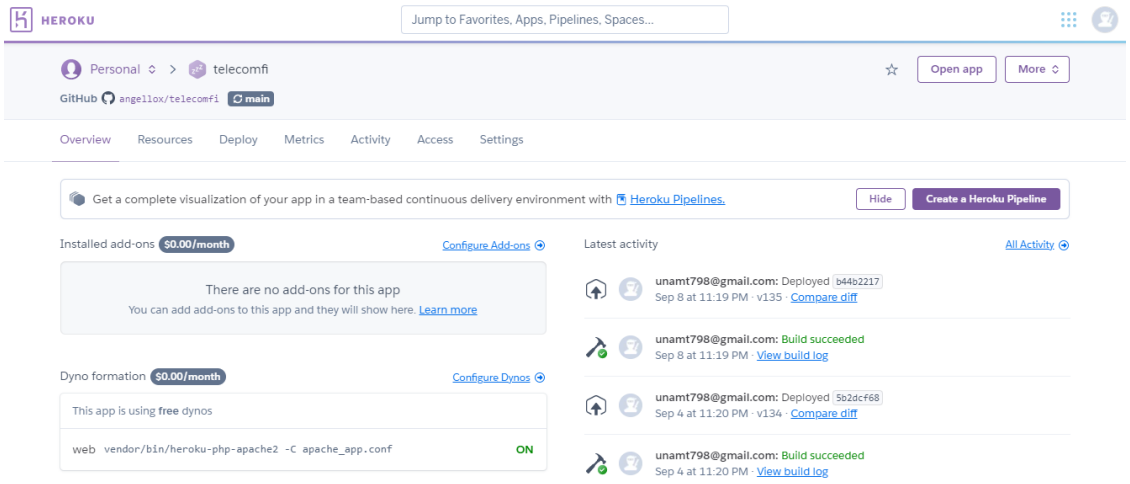


Figura 67: Dashboard del servicio de Heroku.

7.1. Deployment con GitHub

Finalizando el proyecto y para futuras modificaciones de la plataforma, se implementó un controlador de versiones con la plataforma de **GitHub**, que permite actualizar los archivos que son modificados del proyecto y automáticamente, subirlos a la plataforma de hosting, en este caso, Heroku. La automatización de este tipo de controladores permite ahorrar tiempo y gestionar los archivos del proyecto de una forma eficaz.

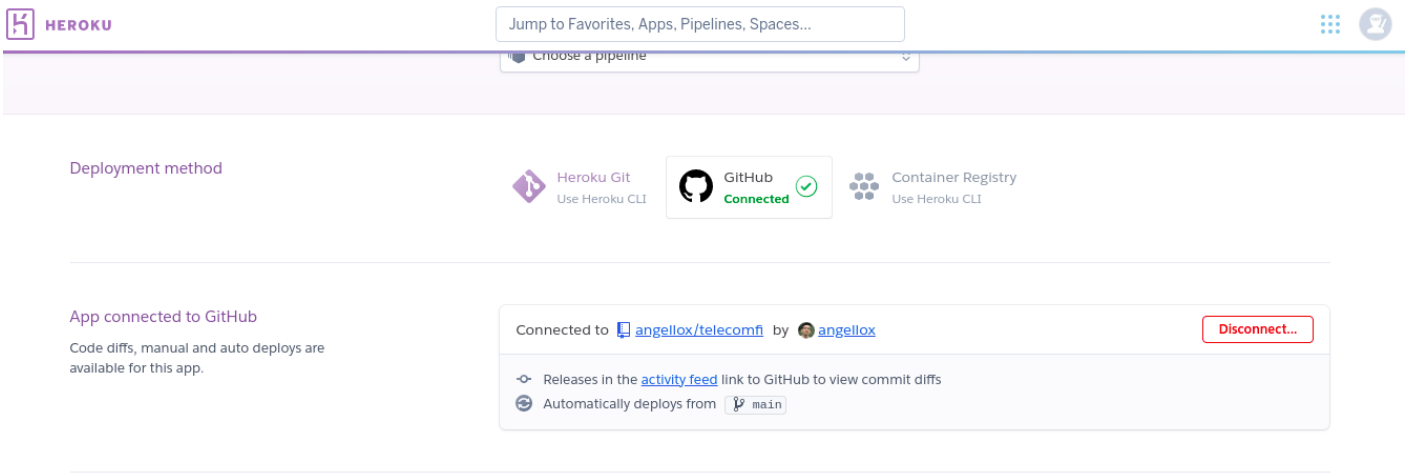


Figura 68: Vinculando GitHub con Heroku para el deployment

Para ello, se requiere de una instalación de GitHub a nivel de linea de comandos, además de que el soporte está bien sustentado en Windows, Mac OS y Linux. También, fue necesario vincular Heroku con GitHub para lograr este objetivo. Los comandos que a continuación se muestran, permiten desplegar el proyecto hacia Heroku y que automáticamente a través de GitHub, se hagan las configuración pertinentes.

```
>> git init
>> git add .
```

Figura 69: Comandos desde el CLI de GitHub para inicializar el proyecto desde la computadora local.

```
>> git commit -m "cualquier comentario"
>> git branch -M main
>> git remote add origin URL_DEL_PROYECTO
>> git push -u origin main
```

Figura 70: Comandos desde el CLI de GitHub para desplegar archivos modificados del proyecto a Heroku.

No esta demás indicar que, el repositorio de GitHub está privado, de esa forma nadie puede ver el contenido, excepto los administradores del proyecto.

8. Resultados obtenidos

El resultado final de todo el proyecto puede perfectamente visualizarse a través del siguiente link. Dominio el cuál es proporcionado por DGTIC - UNAM (antes estuvo alojado en Heroku):

https://labstelecom.unam.mx/

Con fines de representar lo obtenido, se mostrará a continuación, capturas de pantalla de la plataforma y algunas de sus páginas:

Figura 71: Página de inicio

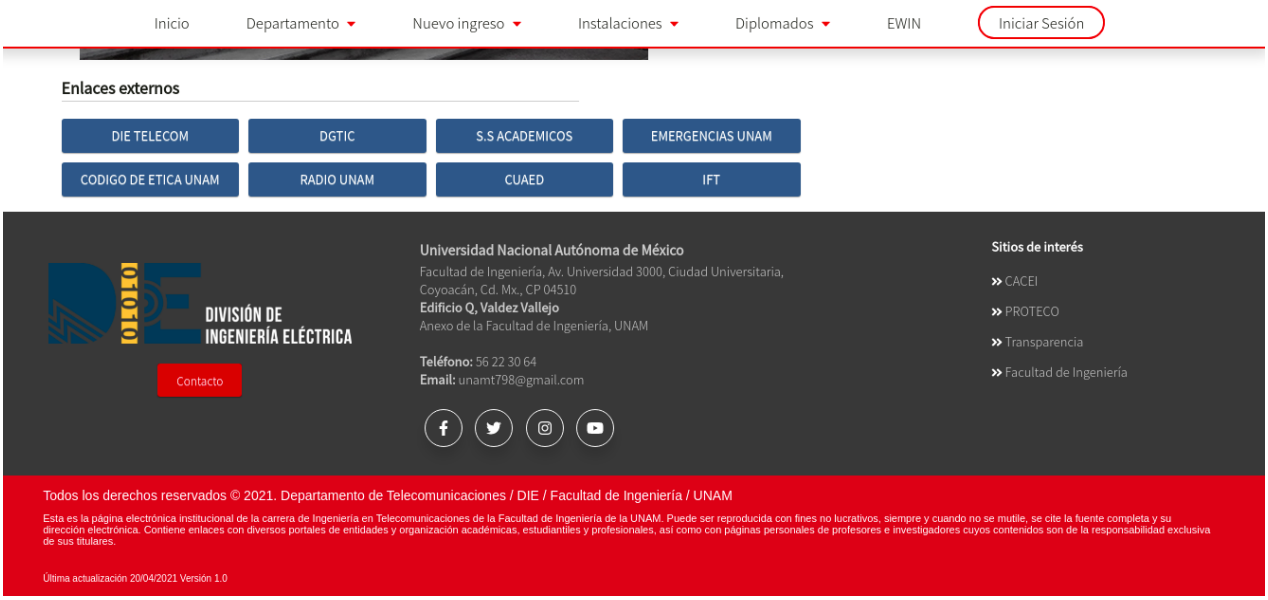


Figura 72: Pie de página de la página de inicio.

En las anteriores figuras, se observa que existen diversos enlaces de interés en los cuales, los usuarios pueden dar clic y redirigir a ciertas páginas. Así como, un pie de página que muestra un botón de contacto hacia un formulario que puede ser llenado en caso de solicitar más información al departamento; finalmente sus redes sociales y algunos sitios más de interés.

Figura 73: Página que muestra la historia del departamento.

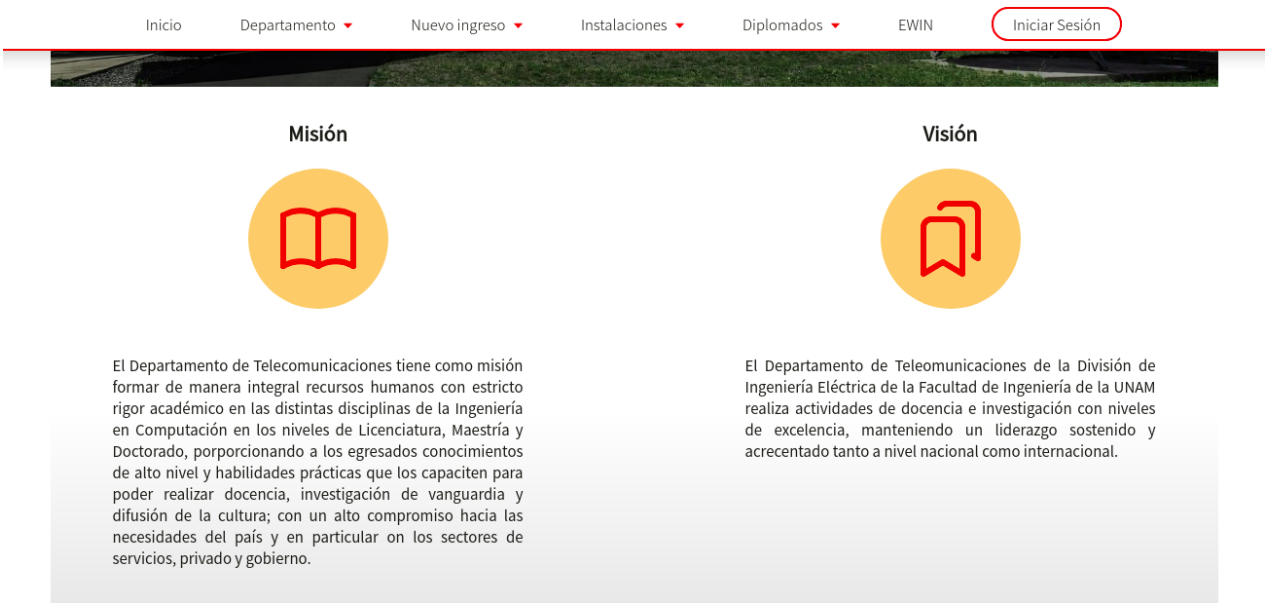


Figura 74: Página de la misión y visión de la carrera de Ingeniería en Telecomunicaciones

Algunas páginas se pueden visitar libremente para saber más del departamento, como las que se muestran actualmente, en donde se puede consultar más información del departamento.

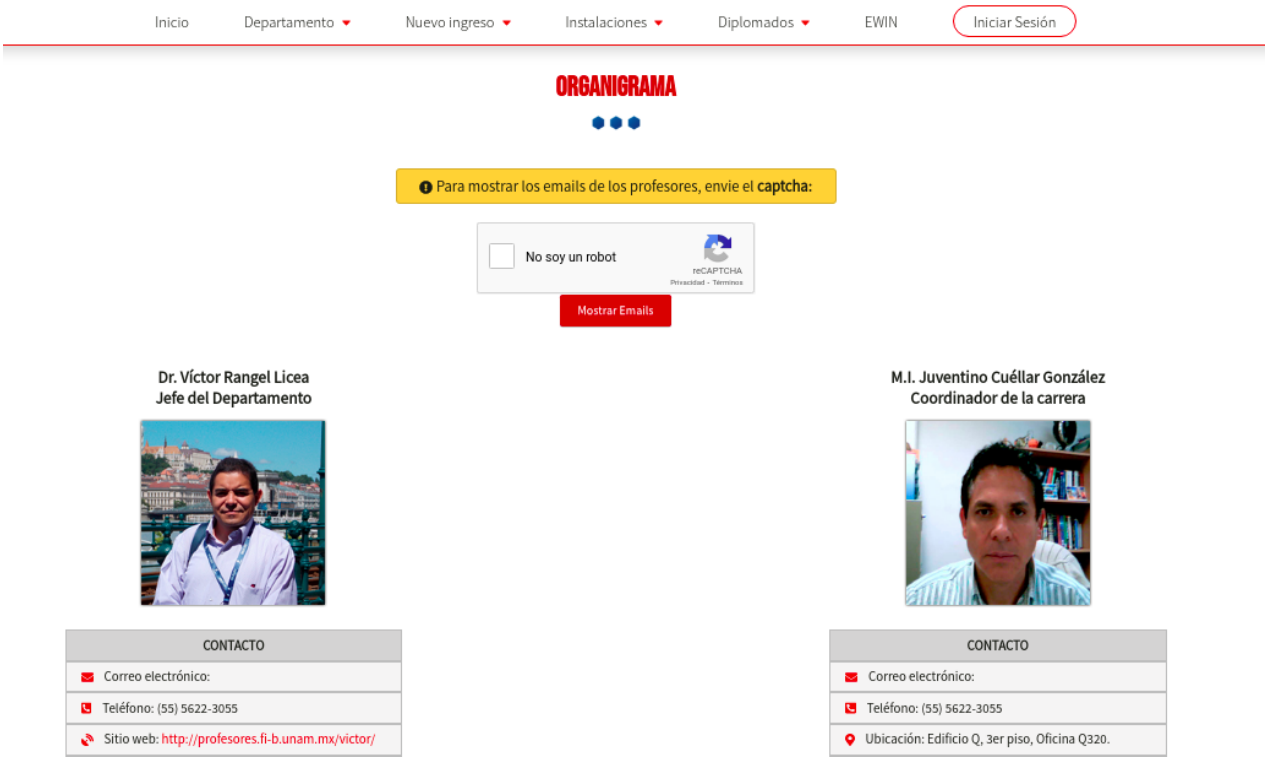


Figura 75: Página del organigrama que compone al Departamento de Telecomunicaciones.

En la siguiente figura, se observa una de las páginas de los laboratorios que pueden ser encontrados en el subapartado del menú de instalaciones. Cada uno de los laboratorios contiene el mismo diseño pero, diferente contenido.

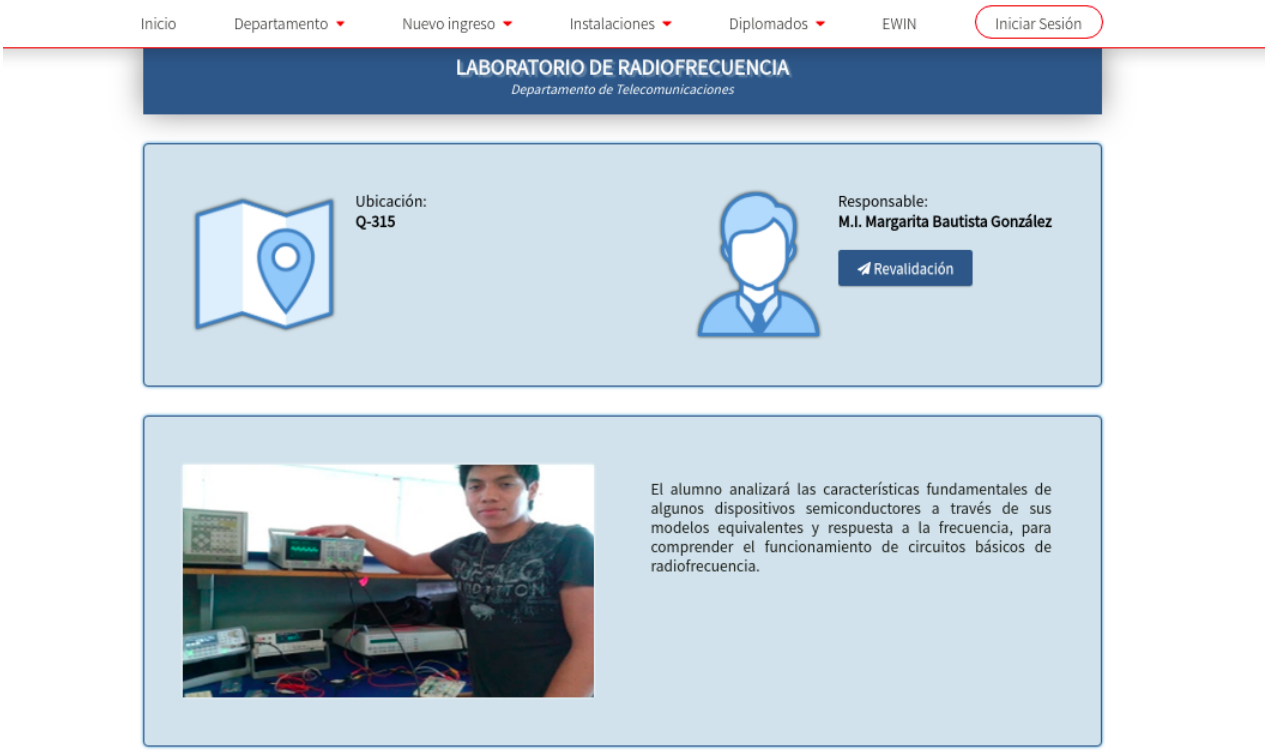


Figura 76: Página de uno de los laboratorios que imparte el departamento.

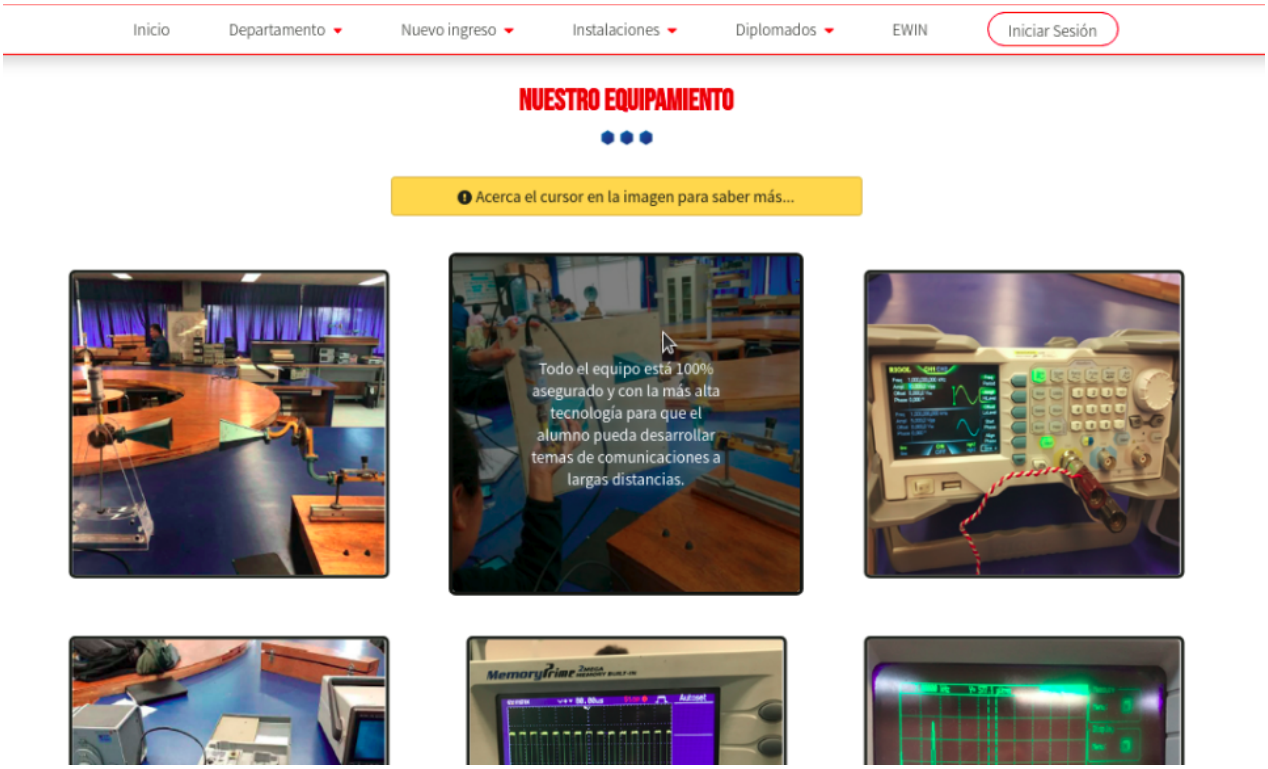


Figura 77: Página de los materiales que se encuentran en los laboratorios de telecomunicaciones.

También, existe un subenlace hacia otra gama de páginas que corresponden a los diplomados que se imparten en telecomunicaciones. Sin embargo, esta página aún está en construcción por parte de los administrativo, allí es donde se subirán algunos flyers de los diplomados, fechas e inscripciones que se desean realizar a éstos.

Figura 78: Página de diplomados de Telecomunicaciones.

Cuando los académicos, administrativos y alumnos dan clic en el botón de inicio de sesión, verán en sus pantallas un pequeño formulario para escribir su correo electrónico y contraseña, mismo que se les hizo llegar por medio de correo electrónico por parte del administrador de la página.

Figura 79: Página de inicio de sesión.

Cuando un alumno inicia sesión, verá en su cuenta, los respectivos datos que bajan del servidor de Azure y la base de datos, en una interfaz que fue mostrada en el capítulo 5, donde observará sus respectivas calificaciones:

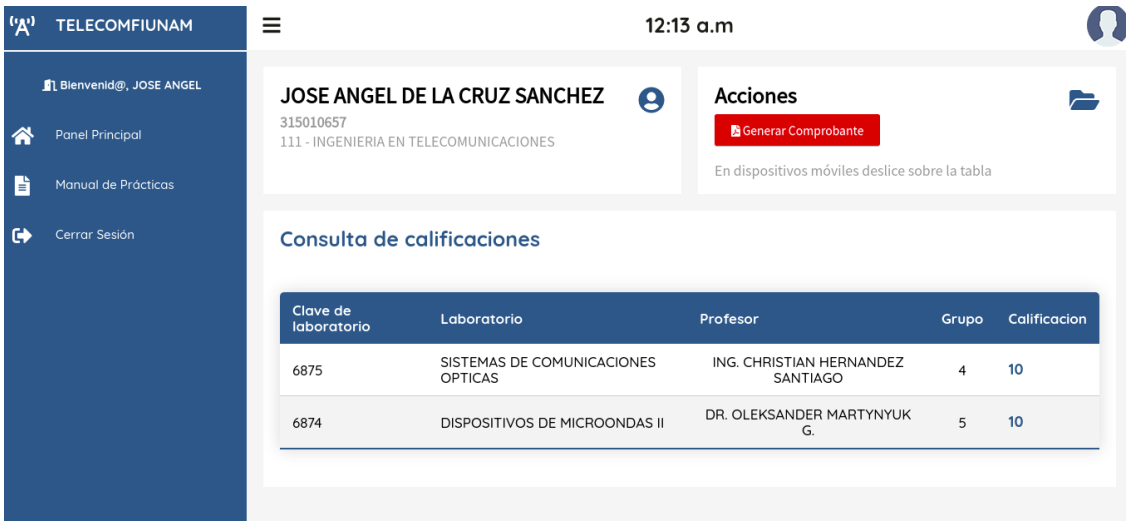


Figura 80: Sesión de un alumno.

De igual forma, el alumno puede dirigirse a manual de prácticas si quiere descargar alguno de los disponibles, que el propio administrador puede ir actualizando conforme se vaya solicitando por parte de los laboratorios. Estos manuales son descargados en formato **.pdf**

Otro inicio de sesión importante, es el del administrador. En la siguiente figura podrá observarse la gama de opciones que se tienen, cada una de ellas fueron las que se programaron funcionalmente con PHP.

Figura 81: Sesión del administrador.

En la opción de **passwords** el administrador es capaz de enviar el password al alumno o profesor una vez que introduzca el correo electrónico. De igual forma, como se mencionó anteriormente, si el correo electrónico no existe en la base de datos, éste no se envía a ningún lado. Si existe y ya posee una contraseña se enviará una alerta de que el password se sobre-escribirá, esto tiene que tener autorización del alumno o profesor en caso de que haya olvidado su contraseña. Finalmente, si es un nuevo usuario, podrá ser enviado, siempre y cuando se cumpla la primera condición:

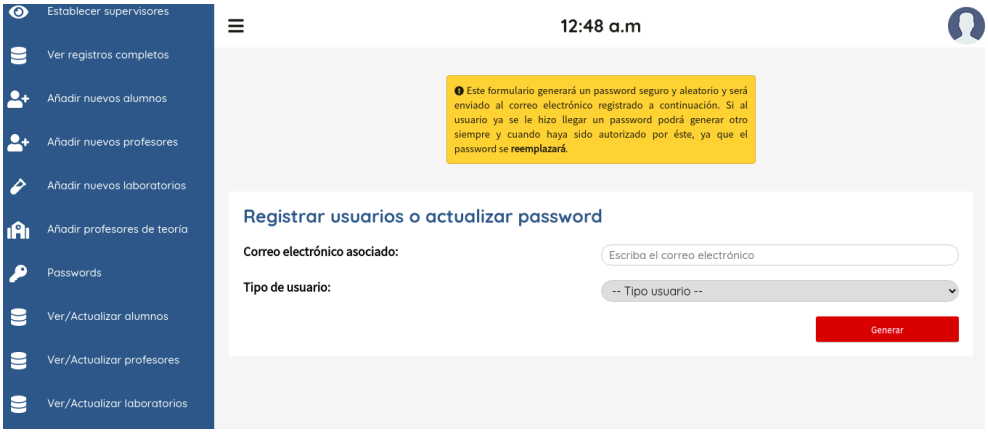


Figura 82: Sesión del administrador.

Entre otras opciones, puede subir flyers al apartado del inicio o de conferencias, es decir, puede subir una imagen que contenga datos a mostrar y un link (opcional) en caso de que se desea que los usuarios al dar clic sean redirigidos a alguna página. Las otras opciones como: **añadir alumnos, laboratorios, profesores** son literalmente, formularios donde se piden datos de los respectivos campos a añadir, y que son inyectados a la base de datos.

Y finalmente, el inicio de sesión de un profesor se muestra a continuación. El profesor tendrá la tabla de sus alumnos de laboratorio y en dado caso de ser profesor de teoría también, podrá visualizar otra tabla con los datos correspondientes. Para ahorrar tiempo a los profesores, las tablas contienen un campo de búsqueda y filtrado (esto fue programado con JavaScript) para buscar alumnos rápidamente, así como una sección de los diversos laboratorios que un profesor pueda tener:

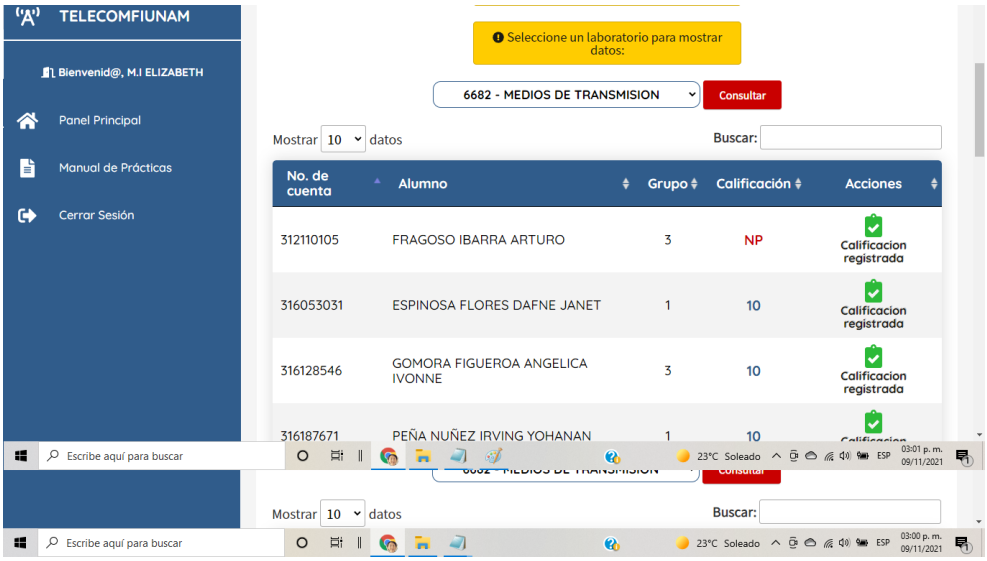
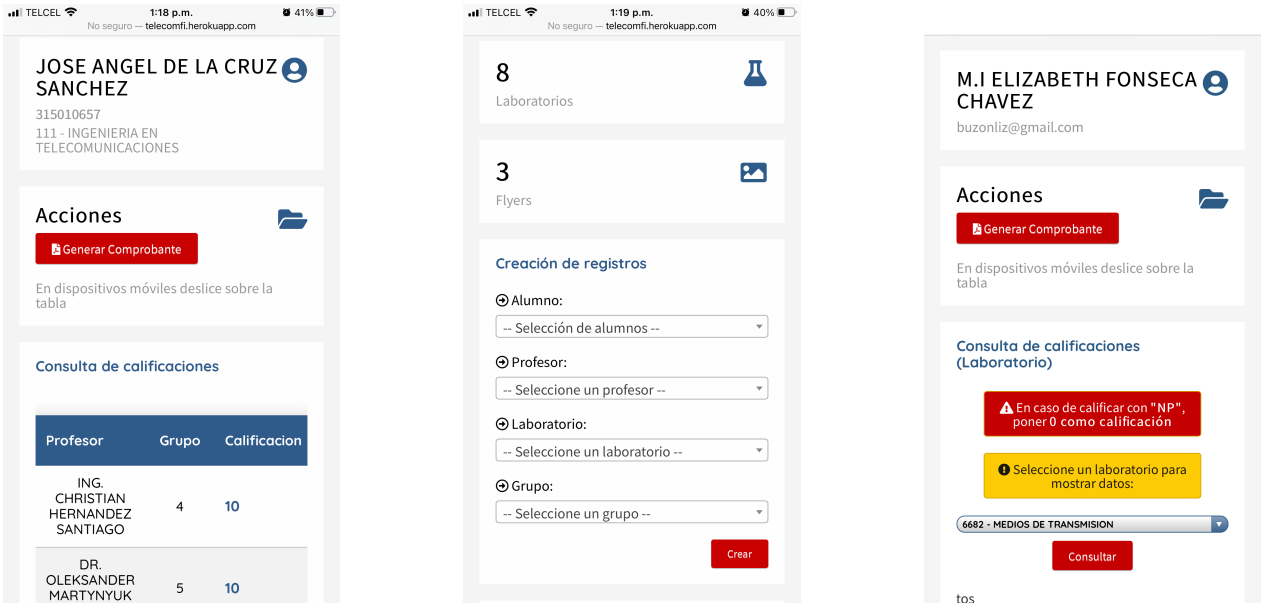


Figura 83: Sesión del administrador.

Todas las páginas pueden ser observadas en dispositivos móviles o tabletas, dado que, todo fue implementando con el diseño responsivo y los media queries que se explicaron en el capítulo 5. Algunos ejemplos son mostrados en las siguientes figuras:



(a) Sesión alumno (b) Sesión administrador (c) Sesión profesor

Figura 84: Inicio de sesión en dispositivos móviles (Diseño Responsivo)

9. Conclusiones

El paradigma del desarrollo de aplicaciones ya sea web, escritorio o móviles requiere de una planificación ardua y sistemática para poder tener resultados satisfactorios. El desarrollo de este proyecto no solo permitió investigar tecnologías de desarrollo en programación para aplicaciones web, sino aplicar conceptos de telecomunicaciones para el correcto funcionamiento del envío de información a través de servidores, en este caso, servidores web. Fue necesario también, tener conocimiento de tecnologías que están emergiendo hoy en día en el mundo de las redes, es decir, la nube.

La nube entre sus tantas características permitió almacenar prácticamente todo el proyecto y que fuera visto por todos a través de internet y de los protocolos HTTP de una manera fácil y rápida. Otro aspecto importante, fue la parte de back-end del sitio o de la plataforma. Al principio todo parecía realmente fácil, el manejo de la estructura HTMl, CSS para darle estructura visual al proyecto, sin embargo, cuando el departamento empezó a listar todo lo que se requería, era necesario no solo quedarse allí, sino migrar a lenguajes avanzados que permitieran desplegar animaciones, funciones, conexiones, etc. Es aquí donde los conceptos de programación que fueron formados a lo largo de mi trayectoria escolar, fueron aplicados.

Uno de los principales objetivos que se tuvieron siempre en mente cuando se desarrolló este proyecto, era que la plataforma serviría como un medio de comunicación entre alumnos y profesores, específicamente, para tener una coordinación entre las calificaciones del laboratorio al que pertenece cada uno, lo cual es importante, porque es la base principal. Si bien, el proyecto contendrá muchas páginas de presentación, la parte esencial y más grande radicaba saber en cómo mostrar cada uno de los datos y que concordaran con la base de información que cada profesor y alumno tuviera en ese momento.

Más aún, el hecho de que el proyecto no era algo trivial, sino algo aplicado ya con datos que no iban a hacer 10, 20 o 30, requería de un diseño o patrón especifico que tuviera un orden y que fuera fácil de manejar a lo largo del desarrollo de la plataforma. Junto con PHP, MySQL y el patrón de diseño MVC se logró satisfactoriamente esta parte. No solo el patrón de diseño permitió aplicar conceptos como *orientado a objetos* o programación funcional, sino también, tener claro cada una de las acciones que se requerían hacer cuando el usuario iniciara sesión, por ejemplo. De igual manera, el diseño de la base de datos relacional, sería mucho más sencilla y práctica, que si todo el proyecto no tuviera una base firme y sistemática.

Por otro lado, la base de datos fue un tema complicado en el sentido del manejo de la información. Esta información será transportada por internet, una red de redes, donde todos alrededor del mundo comparten datos en cualquier momento, y es importante mantener una línea en el conocimiento de seguridad de la información. No solo es aplicar conceptos de programación y quedarse con la idea de que los datos fueron visualizados correctamente, hay que tener en cuenta como los datos llegaron a ser visualizados. Es aquí donde una investigación profunda de la composición de un lenguaje, como lo es PHP, permitió saber que existen herramientas de seguridad que aseguran la integridad de los datos con la base de datos, en donde sea que resida en algún servidor en la nube. Sanitizar los datos una vez que el usuario introduce en formularios, manejar protocolos de seguridad a nivel de capa 4 del modelo OSI, encriptación y hasheo de contraseñas, son temas cruciales, al final del día es lo que importa para que una aplicación sea robusta y confidencial en todo momento.

Por lo tanto, si bien el proyecto contiene mucho de varios campos de la ingeniería, como lo es el buen diseño de una aplicación real, también facultó la idea de la resolución de problemas, una vez el proyecto es lanzado, no todas las funciones son desplegadas correctamente como en las pruebas locales si. Uno de los mayores problemas es cuando aparecieron errores en la conexión con el servidor y la base de datos, definir en donde está el problema, es una de los pasos que como ingeniero te llevan a desarrollarte, en este caso, dada la buena orientación de los profesores y todos los temas vistos a lo largo de este proyecto, los problemas se resolvieron en tiempo y forma.

10. Referencias

Referencias

[1] Web Development and Design Foundations with HTML5, Global Edition. (2021). Pearson.

[2] Puig, J. C. (2013). CSS3 y Javascript avanzado. Universitat Oberta de Catalunya.

[3] Silverio, S. (2019, 28 abril). What Happens When I Type ”holbertonschool.com” into My Browser and Press Enter? Medium. https://medium.com/hackernoon/what-happens-when-i-type-holbertonschool-com-into-my-browser-and-press-enter-9a9a643d1aee

[4] Almishal, Abdulelah & Youssef, Ahmed. (2014). Cloud Service Providers: A Comparative Study. International Journal of Computer Applications & Information Technology. 5. 2278-7720.

[5] Cloud Services - Amazon Web Services (AWS). (2019). Amazon Web Services, Inc. https://aws.amazon.com/

[6] Cloud Computing Services. (2021). Microsoft Azure. https://azure.microsoft.com/en-us/

[7] Cloud Application Platform | Heroku. (2020). Heroku Cloud Application Platform. https://www.heroku.com/

[8] Role of HTML, JavaScript, and PHP in Web Development. (2019, 26 julio). http://www.web-development-institute.com/role-html-javascript-and-php-web-development

[9] reCAPTCHA v2. (2019). Google Developers. https://developers.google.com/recaptcha/docs/display

[10] Generalidades del protocolo HTTP - HTTP | MDN. (2021, 5 septiembre). Mozilla Developers. https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[11] Routing in MVC. (2019). TutorialsTeacher. https://www.tutorialsteacher.com/mvc/routing-in-mvc

[12] Schneier, B. (2015b). Definition of Blowfish. En Applied Cryptography: Protocols, Algorithms and Source Code in C (20th Anniversary ed., pp. 282–285). Wiley.

[13] Foreign Key Constraint. (2021). CockroachDB. https://www.cockroachlabs.com/docs/stable/foreign-key.html

[14] Ross, I. (2013). SMTP. 6th ed., Computer Networking: A Top-Down Approach (6.a ed., pp. 121–124). Pearson.