UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Mi labor como consultor de tecnologías de información.

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Eduardo Daniel Flores González

ASESOR DE INFORME

Ing. Erica Guzmán Vargas

Ciudad Universitaria, CD. MX., 2021

Contenido

Introducción

Los avances en tecnología y el desarrollo de nuevas herramientas de software facilitan cada vez más diversas actividades del ámbito profesional; el uso de sistemas computacionales para la gestión de procesos y automatización de actividades es cada vez más común.

Para el ingeniero en computación, las oportunidades que brinda el campo laboral mexicano, se encuentran en el desarrollo de sitios y aplicaciones web, sistemas de escritorio, gestión de sistemas de bases de datos, configuración y administración de redes, desarrollo de aplicaciones móviles, cómputo en la nube, sistemas de seguridad y más recientemente, sistemas con inteligencia artificial, como soluciones a las necesidades de MIPYMES y grandes empresas.

Sin importar el giro comercial que tengan las organizaciones, el uso de sistemas computacionales hoy en día (2020) es sumamente necesario, pues representa una oportunidad de mejora, tanto en el área administrativa, como en la operativa, además de representar una disminución considerable de costos de operación cuando se implementan herramientas y procesos de automatización. Por tanto, la figura del profesional de tecnologías de la información (TI) desempeña un rol muy importante en la operación diaria de las compañías, brindando asesoría técnica y administrativa, dando mantenimiento a los sistemas actuales, así como diseñando y desarrollando nuevos, según el negocio lo requiera.

Objetivo

El presente informe tiene la finalidad de exponer de manera breve las actividades que he realizado a lo largo de mi desempeño profesional hasta el día de hoy, como consultor e ingeniero de software en diversas empresas; proporcionando un contexto general acerca de las actividades realizadas, las responsabilidades que he tenido en el desempeño de mi profesión, los principales retos a los que me he enfrentado y las soluciones que implementé en cada caso.

Justificación

A lo largo de mi carrera profesional, he colaborado en diferentes organizaciones y empresas, tanto en el sector gubernamental como en el privado, como desarrollador de sistemas, consultor e ingeniero de software. Realizando actividades como:

- Análisis y levantamiento de requerimientos y necesidades del cliente.
- Análisis, diagnóstico y mantenimiento de sistemas actuales.
- Desarrollo de nuevas funcionalidades, mejoras y sistemas nuevos.
- Documentación de las modificaciones y desarrollos realizados.

Cada nuevo proyecto demanda siempre un esfuerzo que puede ser similar o no a lo realizado anteriormente; ya que, a pesar de que a grandes rasgos las actividades a realizar y los componentes de cada sistema son los mismos en todo proyecto de software, la variedad y desarrollo de nuevas tecnologías hacen que cada proyecto, cada empresa y

cada cliente tengan sus particularidades. En el presente trabajo presento algunas de las situaciones que he experimentado en mi desempeño profesional.

Descripción del puesto de un consultor

A lo largo de mi experiencia profesional me he desempeñado como desarrollador, consultor e ingeniero de software, he trabajado con equipos multidisciplinarios en cada uno de los proyectos en los que he participado, ya que en la ejecución de un proyecto de software intervienen muchos perfiles profesionales, en primera instancia se encuentran el equipo de ventas, que es quien contacta al cliente y cierra con él la contratación de los servicios de la empresa para el desarrollo del sistema, una vez concretada la venta del proyecto, el equipo de administración de proyectos se encarga de la gestión del equipo de profesionales que llevarán a cabo el desarrollo del producto.

Dentro de un proyecto de software, es importante contar con la participación de diferentes perfiles, idealmente se debe contar con un equipo de diseño, tanto de interfaz como de experiencia de usuario, que se encargarán del diseño de las vistas y la navegación del usuario a través del sistema, respectivamente; un equipo de pruebas, que se encargará de realizar las validaciones necesarias para comprobar que lo que se está construyendo sea lo que se especificó y tal cómo se especificó. Y por último, se encuentra el perfil del consultor de software; este es el perfil de quienes se encargarán de construir el producto, por lo que su papel dentro del proyecto es esencial (al igual que los anteriores), el equipo de ingenieros y consultores de software es el responsable de determinar la arquitectura

del sistema a desarrollar, las tecnologías a utilizar para su desarrollo, así como las herramientas y estrategias necesarias para garantizar la seguridad de la información que se aloje dentro del sistema.

Objetivos del puesto

- Analizar el estado actual de los sistemas existentes.
- Analizar nuevos requerimientos para la implementación de mejoras, modificaciones y módulos adicionales a los sistemas existentes.
- Generar propuestas de mejora a los sistemas existentes y desarrollos pendientes (en caso de haberlos).
- Implementar nuevas funcionalidades.
- Mantener actualizado los sistemas conforme a las nuevas tecnologías disponibles en el mercado.
- Identificar necesidades y áreas de oportunidad dentro de la operación de las empresas para generar una estrategia de mejora.
- Realizar el levantamiento, evaluación y análisis de necesidades y requerimientos para el desarrollo de un sistema nuevo.
- Mantener actualizada la documentación de los sistemas actuales, así como generar la documentación correspondiente a los nuevos desarrollos.

Responsabilidades del puesto

- Revisión y estimación de funcionalidades y nuevos requerimientos solicitados por el cliente, así como las mejoras y las modificaciones necesarias que se detecten dentro de los sistemas.
- Desarrollo y mantenimiento de los sistemas y aplicaciones asignados.
- Desarrollo y ejecución de pruebas unitarias dentro del código.
- Participación en el diseño e implementación de la arquitectura del sistema.
- Generación de nuevos sistemas o funcionalidades dependiendo de las necesidades del cliente y del negocio.

¿Qué es un consultor?

La consultoría es un servicio profesional de asesoramiento que se brinda a un usuario o cliente, ya sea que se trate de un profesional en una determinada área, una empresa, organización o un órgano gubernamental.

"Tiene su origen en la revolución industrial, luego de iniciarse la organización científica del trabajo, por Taylor Gilbreth, Gantt y Emerson (considerados los pioneros en este tema), como consecuencia de la necesidad de las empresas por optimizar sus procesos productivos"[1].

[1] Lavín Ivan, La consultoría a través del tiempo. ***Ver referencias al final del documento.***

Con el paso del tiempo su demanda se incrementó debido a la creciente complejidad y competitividad de los negocios del mundo moderno. Algunos factores que han favorecido su fortalecimiento han sido el crecimiento de las tecnologías y la constante aparición de nuevos conocimientos y técnicas que deben ser adquiridos con rapidez por las empresas, para no perder competitividad.

En este sentido, un consultor es aquel individuo que posee un determinado nivel de conocimientos y experiencia tal, que puede ser considerado como un experto en una o más áreas dentro del campo laboral. Por lo que, debe contar con las siguientes cualidades:

- Tener experiencia previa en el área de conocimiento en cuestión.
- Tener referencias que le permiten comparar diferentes escenarios.
- Habilidad para identificar problemas rápidamente.
- Capacidad de proponer soluciones rápidas y sencillas a los problemas identificados.
- Ser empático.
- Ser un buen comunicador.

El consultor es quien, gracias a la experiencia que posee y los puntos anteriormente mencionados, brinda asesoría al empleador que contrata sus servicios. Debido a que el empleador no siempre tiene conocimiento del o los problemas reales que necesitan ser resueltos, la experiencia y capacidad de análisis del consultor juega un papel importante, pues si no se tiene un diagnóstico adecuado, el proceso de consultoría pude tener consecuencias negativas, tanto para el consultor como para el contratante.

Las empresas.

En mi desempeño profesional he trabajado con diferentes clientes, ya que una característica muy particular de las consultorías es precisamente la variedad de clientes con los que colabora y proyectos en los que participa. Al estar dentro de una consultoría, los proyectos pueden ser con empresas tanto del sector público como del sector privado, siendo diferentes tanto en la administración de los proyectos, como en la confidencialidad y tratamiento de la información que se maneja. Algunas de las empresas en las que he trabajado son:

- Kata Software – 2019 - Actualidad
- Gobierno de Baja California - 2019
- Grupo ARCA - 2019
- CBX – 2019
- Grupo ProtG – 2018, 2019
- Consejo de la Judicatura Federal - 2018
- PEMEX – 2018
- Kiewit – 2017
- CHUBB – 2017
- SPP sistemas de prepago - 2016
- SPF (Secretaría de protección Federal) – 2015

En general, las empresas en las que he colaborado, tienen giros diferentes: que van desde dependencias gubernamentales en los ramos de seguridad y energético hasta empresas

privadas en los ramos energético, asegurador y de construcción. Todas ellas con necesidades diferentes pero que dentro de su operación trabajan con grandes cantidades de información y realizan operaciones que pueden facilitarse con el uso de tecnologías y sistemas.

Esta es el área de oportunidad que la ingeniería en computación tiene dentro de las empresas, ya que no está restringida a un ramo en particular, cualquier organización que maneje información y que realice operaciones administrativas que pueden automatizarse representan oportunidades perfectas para la labor de consultoría e ingeniería de software; pues con la maduración de los procesos y actividades dentro de las organizaciones, la definición de parámetros y necesidades se vuelve más clara, facilitando así nuestra intervención en el diseño de soluciones tecnológicas.

Antecedentes.

Aunque similares en algunos aspectos (arquitectura y tecnologías), los proyectos en los que he colaborado han sido diferentes (funcionalidad, codificación y organización) y por consiguiente me he encontrado con diversas situaciones que a su manera en particular representan un reto y situaciones diferentes.

En algunas ocasiones los sistemas desarrollados son propios de la empresa, lo que permite tener una mayor fuente de información en los casos en que no se cuenta con la documentación necesaria para entender el sistema, además de una mayor rapidez en la ejecución de modificaciones y una disminución en la curva de aprendizaje.

En otras ocasiones los sistemas desarrollados fueron realizados por terceros, ya sea otra consultora, fábrica de software, o por desarrolladores independientes. En estos casos es de vital importancia contar con la adecuada documentación del sistema, manuales, diagramas de arquitectura, controles de cambios, etc. Desafortunadamente esto no siempre es una realidad, pues buena parte de los desarrollos que se realizan tienen documentación inconclusa o poco clara, lo que dificulta el rápido entendimiento del sistema y por consiguiente incrementa el tiempo necesario para realizar detección de oportunidades de mejora, implementación de funcionalidades adicionales y el mantenimiento del sistema en cuestión.

En ambas situaciones llega a presentarse el caso de que quienes desarrollaron los sistemas no tienen el perfil de ingeniero en computación o ingeniero en sistemas, sino que pertenecen a otra disciplina pero que fueron asignados a la creación de las herramientas. El impacto que tiene esto en el código es que no necesariamente cumple los lineamientos necesarios para garantizar su escalabilidad y mantenimiento. Es decir, el código no fue desarrollado de tal forma que cumpla con buenas prácticas de desarrollo, por ejemplo:

1. Mantener el código lo más simple posible
2. Estandarizar el nombramiento de clases, atributos, métodos, variables, etc…
3. Realizar comentarios en el código, que faciliten su comprensión
4. Mantener la correcta alineación de código en las funciones, métodos, etc…
5. Utilizar validaciones en datos de entrada
6. Agrupar líneas de código comunes en clases o métodos

Dado lo anterior, mi labor como consultor, junto con los equipos con los que he colaborado, ha sido analizar los sistemas actuales, desde la organización, la relación entre los módulos que componen cada parte del sistema, la estructura de la base de datos en que se aloja la información, además de la implementación que nuevas funcionalidades que el negocio requería para su operación.

Participación profesional

Un sistema está compuesto por varias partes o módulos conectados entre sí, los cuales en conjunto permiten realizar las operaciones para la cual fue diseñado, estos módulos comprenden la base de datos en la que se almacenará la información con la que trabajará el sistema, la capa de servicios (back end), que utilizarán la información de la base de datos para leerla, modificarla y realizar operaciones con ella para poder generar información nueva, y la capa de vistas (front end), la cual comprende las ventanas y vistas con las que interactuará el usuario cuando trabaje con el sistema.

Debido a lo anterior, las actividades que he desempeñado durante mi carrera profesional comprenden diferentes áreas del desarrollo de sistemas, como las siguientes:

- Arquitectura del sistema
- Migración de ambientes
- Base de datos
- Desarrollo de servicios
- Desarrollo de vistas

En cada una de estas áreas me he visto en la necesidad de estar en constante capacitación, debido a la naturaleza de cada proyecto y a la evolución continua de las tecnologías empleadas en el desarrollo de sistemas. Así mismo cada área tiene consideraciones que deben tenerse presentes para poder llevar a cabo las actividades, a continuación, presento algunas de las situaciones que he experimentado y las acciones que tomé para poder manejarlas.

Arquitectura y migración de ambientes

La arquitectura de un sistema es la base sobre la cual está construido, tanto en la parte del hardware como del software. En este punto he trabajado principalmente con el análisis, migración, diseño e implementación de arquitecturas en la nube, utilizando la plataforma de Microsoft Azure, la cual es el centro de administración del servicio de nube de Microsoft.

La razón principal para el uso de los servicios de cómputo en la nube de Microsoft, es debido a que, las empresas en las que he colaborado, son “Gold Partners” de Microsoft, es decir, son empresas que cumplen los requisitos solicitados por Microsoft para poder pertenecer a esta categoría de socios, algunos de los cuales son:

- Aprobar los exámenes de certificación necesarios y la validación de conocimientos.
- Cumplir con los requerimientos de desempeño (estos varían dependiendo del tipo de socio al que se desea pertenecer).
- Cubrir la tarifa anual de competencia de oro.

Esta distinción valida la experiencia técnica de la organización, ya que, tanto para obtener el distintivo de “socio oro (Gold Partner)”, como para mantenerlo activo, la empresa debe contar con personal certificado en tecnologías Microsoft y renovar estas certificaciones periódicamente.

Sin embargo, existen otras empresas que ofrecen servicios de cómputo en la nube, como Amazon (con Amazon Web Services: “AWS”), e IBM (con IBM Cloud Solutions), etc… Las cuales, de igual forma, ofrecen herramientas para desarrollar diferentes proyectos y servicios utilizando la nube. Tanto para arquitecturas híbridas (parte local y parte en la nube) o arquitecturas 100% en la nube.

Para realizar el diseño de una arquitectura en la nube, deben conocerse adecuadamente los requerimientos del cliente, el comportamiento y las cargas de trabajo que la arquitectura debe soportar, y por supuesto, la seguridad de la arquitectura y de la información.

Varias de las actividades que he desarrollado se centraron en la migración de ambientes locales hacia la nube; es decir, migrar aplicaciones web desde el servidor del cliente a los centros de datos de Microsoft, migrar la información de las bases de datos locales hacia la nube y configurar, en los casos en que fue necesario, el dominio de la nueva aplicación. Las dificultades más comunes que surgen al momento de trabajar con arquitecturas en la nube son:

- Dimensionar adecuadamente el tamaño de la base de datos a implementar.

- Dimensionar adecuadamente el tamaño y los recursos que serán necesarios para la arquitectura diseñada.
- Comunicación adecuada entre los componentes de la arquitectura.
- Resiliencia y alta disponibilidad de la aplicación, es decir, que la aplicación hospedada en la nube pueda hacer frente a errores y “caídas” del servicio.

Las dificultades anteriores son puntos que deben tomarse en cuenta al momento de realizar el diseño de la arquitectura. Generalmente una arquitectura en la nube tiene la siguiente estructura.

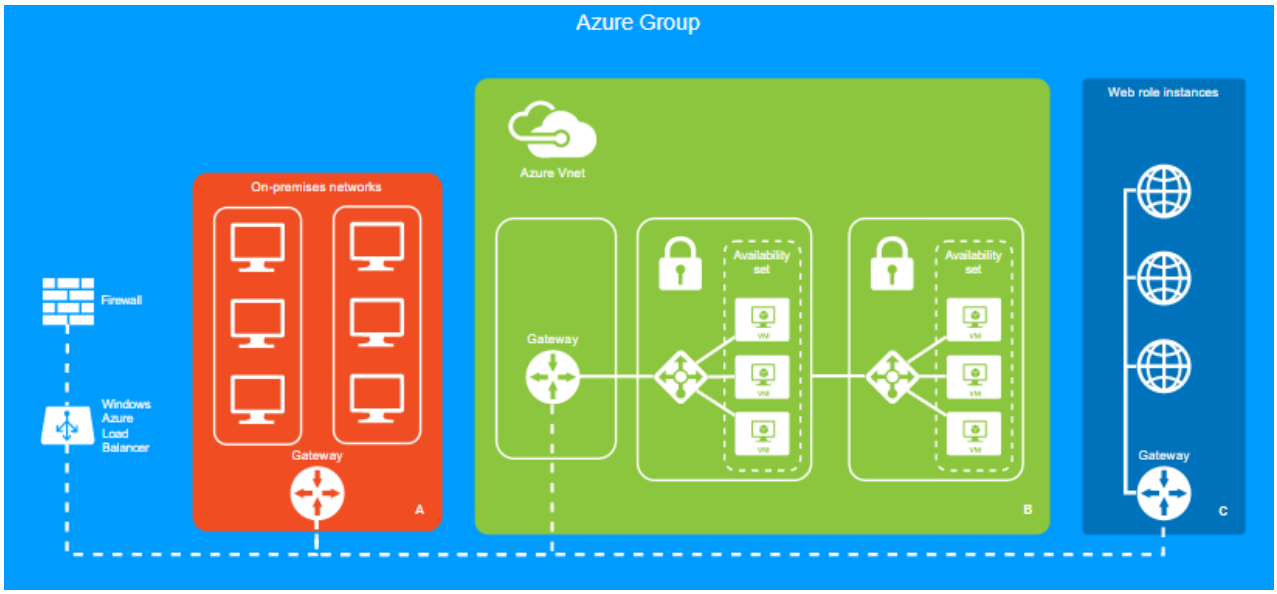

Ilustración 1: Diagrama de arquitectura en la nube.
Fuente: Creación propia, draw.io Diagrams software

Para poder mitigar y aún más importante, prevenir los incidentes relacionados al dimensionamiento de los recursos al momento del diseño de la arquitectura, el mismo portal de Azure es una gran ayuda, ya que es posible crear elementos como máquinas

virtuales y bases de datos con un tamaño inicial, y posteriormente, dependiendo del desempeño y funcionalidad de los elementos provisionados, redimensionar cualquiera de estos con mucha facilidad, y aunque el dimensionamiento impacta en el costo de facturación de Azure, esta funcionalidad permite trabajar únicamente con lo que se está ocupando en ese momento, permitiendo así optimizar costos.

En una ocasión, trabajando con una empresa dedicada al ramo asegurador y al momento de realizar el análisis del ambiente que querían migrar a la nube, me percaté que esta empresa tenía una máquina virtual donde almacenaban tanto la información de la base de datos como de la aplicación web como tal, a medida en que fui conociendo aún más el ambiente, me comentaron que constantemente tenían intentos de intrusión a su página web y en un par de ocasiones habían sido víctimas de hackeo. Con esto en mente, al momento de realizar el diseño de la arquitectura, opté por crear dos máquinas virtuales, una para la base de datos y otra para la aplicación web. Esto con la finalidad de aislar la información de la base de datos y restringir el acceso a la misma únicamente desde la dirección IP de la máquina virtual que contenía la aplicación. Dada la sensibilidad de la información que maneja esta empresa, es indispensable que la información dentro de la base de datos se encuentre bien resguardada, por este motivo, creé dos redes virtuales y dentro de una de ellas realicé el provisionamiento de la máquina virtual que almacenaría la base de datos, eliminé la dirección IP pública que Microsoft Azure asigna a la máquina virtual al momento de su creación, y configuré el grupo de seguridad de la red virtual para aceptar conexiones entrantes únicamente desde la dirección IP privada de la máquina virtual que almacenaría la aplicación web. Con esto garantizaría que la base de datos

podría accederse únicamente desde la aplicación web, y aunque la máquina virtual de aplicación se viera comprometida por alguna razón, la información seguiría disponible.

En cuanto a la máquina virtual que manejaría la aplicación; esta se implementó en la segunda red virtual; de igual forma, configuré el grupo de seguridad para garantizar la comunicación con la máquina de base de datos, pero en este caso también se generaron reglas para permitir su comunicación a través de la dirección IP pública, esto último con la finalidad de que pudiera accederse a ella desde programas como escritorio remoto (Remote Desktop) de Windows, y poder realizar modificaciones y mantenimiento del código de la aplicación.

Para poder disminuir el riesgo de ataques y brindar una capa adicional de seguridad a la aplicación de la empresa, se utilizó el servicio de Cloudflare, para realizar un filtrado de tráfico desde internet hasta el sitio web de la empresa. Cloudflare es un software que actúa como una gran red de puntos de acceso a un mismo sitio, evitando así la propensión del sitio a ataques DDoS[2], además de brindar estadísticas de acceso que permiten al administrador del portal, revisar el comportamiento del tráfico a su sitio a lo largo del día. Para realizar esta configuración, fue necesario realizar el registro de los DNS[3] de la aplicación en el portal de Cloudflare, esto con la finalidad de realizar el filtrado de tráfico web a través de este servicio.

[2] Ataque informático en el que se envían varias solicitudes a un recurso web, con la intención de desbordar la capacidad del sitio web para administrar varias solicitudes y así evitar que funcione correctamente.
[3] Domain Name System, es un servicio distribuido globalmente, que convierte los nombres legibles por las personas (www.ejemplo.com) de un sitio web, en las direcciones IP de cada uno.

La arquitectura mostrada en la ilustración 1, garantiza que la información y aplicación de la empresa se encuentre siempre disponible, ya que el diseño se encuentra realizado en alta disponibilidad. Es decir, la arquitectura se encuentra diseñada de tal manera que a pesar de una falla en la aplicación o en los servicios de Microsoft, la aplicación siga operando con normalidad; desafortunadamente, un diseño en alta disponibilidad implica en varios casos, tener una “réplica” de la arquitectura trabajando en paralelo, como puede verse en la ilustración 2.

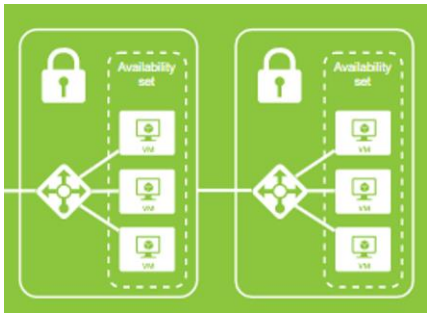

Ilustración 2: Máquinas virtuales de servicio en alta disponibilidad, los balanceadores de carga se encargan de distribuir las peticiones.
Fuente: Creación propia, draw.io Diagrams software

Así, en caso de que alguna de las máquinas virtuales dentro de los espacios de disponibilidad (identificados con líneas puntedas), tuviera algún problema y dejara de responder, el balanceador de carga de cada zona, dirigiría el tráfico a las máquinas que se encuentran disponibles, manteniendo así disponible el servicio.

Dado que esta opción significa un costo más elevado para los clientes, muchos de ellos no ven la importancia de este tipo de diseño, sin embargo, este tipo de arquitecturas

pueden ser de mucha utilidad en caso de alguna falla, tanto en la infraestructura provista por Microsoft, como en los sistemas del cliente como tal.

Al respecto de este último punto, en septiembre de 2018, Microsoft sufrió la caída parcial de uno de sus centros de datos, en la región Centro – Sur de Estados Unidos, causando así la interrupción de todo servicio o sistema alojado en el hardware de ese centro. Situaciones como esta son la razón por la que es importante contemplar alta disponibilidad (preferentemente con redundancia geográfica) en el diseño de arquitecturas, sin importar si son o no en la nube.

El diseño de la arquitectura siempre dependerá del tipo de negocio, los recursos y niveles de servicio que tenga cada cliente. Por ejemplo, en el caso de que el cliente cuente con infraestructura propia, podría evaluarse la forma de utilizarla para generar una arquitectura híbrida, en la que, se realicen respaldos locales de los datos de la organización, para estar preparados ante una posible falla en el servicio. O para migrar los servicios con más demanda a la nube, y mantener en la infraestructura local, los servicios emergentes (en tanto la demanda llega al punto deseado)

Base de datos, desarrollo de servicios y sistemas: El software.

El papel de consultor es brindar asesoría y recomendaciones que permitan mejorar el desempeño de la empresa en el área en la cual se brinda la consultoría, y en mi caso, como consultor e ingeniero de sistemas, en el ámbito del software, mis actividades han tenido como finalidad el mejoramiento de los sistemas informáticos.

Para poder hacer esto, lo primero en cada ocasión es comprender cómo es que funciona la compañía, observar el comportamiento y la finalidad de cada componente de los sistemas ya implementados. Poniendo especial atención tanto en las debilidades como en las fortalezas de los mismos, con la finalidad de generar soluciones efectivas a los problemas que se detecten durante el análisis.

Al momento de realizar el análisis de los sistemas, es importante prestar atención a los siguientes puntos:

Arquitectura del sistema

La arquitectura del sistema en términos de software, se refiere a la identificación o definición (en el caso de sistemas nuevos), de las herramientas con las que se encuentra construido el sistema, las comunicaciones entre los módulos que componen el sistema como tal y los patrones de arquitectura que se emplearon para el diseño de la herramienta que se está analizando.

En los proyectos en los que he participado, el uso del patrón de arquitectura modelo-vista-controlador (MVC), fue con el que más trabajé en un inicio, debido a que, la mayoría de los proyectos, ya se encontraban en una etapa avanzada. Lo que implicó necesariamente seguir con la arquitectura ya propuesta. Mientras que, en proyectos más recientes, he trabajado con el patrón Modelo-Vista Vista-Modelo (MVVM).

Sin embargo, existen otros patrones de arquitectura que pueden utilizarse, dependiendo el proyecto en el que se esté trabajando, por mencionar algunos tenemos los siguientes:

1. Cliente Servidor:

   Compuesto por dos participantes (un servidor y múltiples clientes), en el cual, los clientes solicitan servicios, mientras que el servidor se mantiene “atento” a las solicitudes de los clientes, a fin de proveerles los servicios necesarios.

   Ejemplos: Aplicaciones de correo electrónico, documentos compartidos, etc…

2. Bus de evento:

   Este patrón se utiliza para manejar eventos. Consta de 4 componentes principales: La fuente del evento, la escucha del evento, el canal y el bus de evento. En este patrón, las fuentes de evento publican mensajes en los canales particulares de un bus de eventos, mientras que los oyentes, suscritos a estos canales, reciben las notificaciones, una vez que estás son publicadas por las fuentes.

   Ejemplo: Servicios de notificación móvil

El patrón Modelo – Vista - Controlador, permite tener una estructura definida en el proyecto y facilita así la labor de desarrollo, ya que separa adecuadamente las vistas, el código de manipulación de información y la capa de acceso a datos. Este modelo permite que sea sencilla la realización de pruebas unitarias y el proceso de depuración del código. Para lograr una implementación de este tipo, primero es necesario asegurar que el código sea modular, es decir, que se encuentre constituido por funciones que pudieran ser reutilizables y que sean lo más independientes posible.

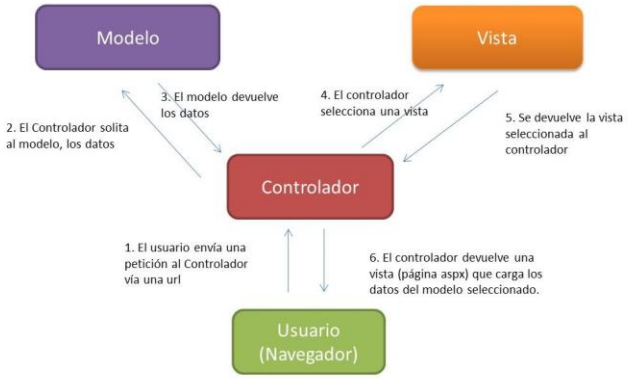

Ilustración 3: Patrón de arquitectura: modelo, vista, controlador (MVC).
Fuente: https://medium.com[4]

Como puede apreciarse en la ilustración 3, el patrón MVC define una línea de comunicación específica entre los diferentes componentes de su arquitectura. Esta línea de comunicación evita que la vista (los elementos visibles por el usuario del sistema) manipule la información que vive dentro de la base de datos; ya que es el controlador quien se encarga de esta tarea. El controlador entonces es el mediador de las peticiones que realiza el usuario desde la capa de vista y la recuperación de información desde la base de datos, que realiza el modelo, para después, manipular esta información y devolverla a la vista para su despliegue.

Este funcionamiento, permite desarrollar sistemas de una manera segura, pues en el controlador pueden implementarse las medidas necesarias para prevenir intrusiones a la información de la base de datos.

[4] David, *Entendiendo MVC y sus problemas.* ***Ver referencias al final del documento.***

En proyectos más recientes, he trabajado con Angular, un marco de trabajo para JavaScript, que permite generar componentes que interactúen entre sí, cada uno conformado por tres archivos:

1. Archivo HTML, donde se trabaja con la vista al usuario.
2. Un archivo CSS, para manejar los estilos de la vista.
3. Un archivo TS (typescript), código tipado para Javascript, que permite vincular la vista con el modelo de datos, mediante la propiedad “two-way data binding”.

Por este último punto (two-way data binding), Angular no trabaja con un patrón MVC, sino que, al permitirnos identificar en la vista el modelo de datos que se está ocupando, la vista y el modelo son dependientes entre sí. Una modificación en el modelo nos permite modificar la vista (contrario a como ocurre en el patrón MVC, en el que la vista y el modelo de datos son independientes).

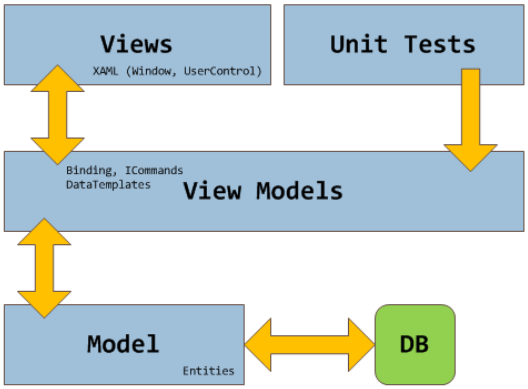

Ilustración 4: Patrón de arquitectura: modelo, vista, vista, modelo (MVVM)
Fuente: https://openwebinars.net/[5]

[5] Montero Ortega, J.M, *La arquitectura MVVM y sus componentes.* ***Ver referencias al final del documento.***

Este modelo es conocido como Modelo – Vista Vista -Modelo (MVVM) (ver ilustración 4), y aunque similar al patrón MVC, en el patrón MVVM, el acoplamiento entre la vista y el modelo de datos es débil, por lo que un modelo puede ocuparse en varias vistas, haciendo más fácil el trabajo paralelo entre diseñadores y desarrolladores. Y permitiendo que los cambios en la vista afecta en menos a la capa de modelo, a diferencia del patrón MVC, en donde la vista afecta la plantilla del modelo de datos que usa el controlador.

Es importante definir (en el caso de sistemas nuevos) e identificar (en el caso de sistemas existentes), las herramientas con las que se construirá el sistema. Es decir, ¿Qué software se ocupa u ocupará para la base de datos? ¿Qué software se ocupa u ocupará para realizar los servicios (back end) del sistema? Y ¿Qué software se utiliza o utilizará para construir las vistas (front end)? todas estas cuestiones siempre han sido constantes en los proyectos en los que he trabajado y son la parte medular de mi labor profesional.

Base de datos

Todo sistema necesita datos para poder trabajar, es decir, necesita información para llevar a cabo las actividades para las que fue diseñado. Por ejemplo, un sistema de administración escolar, necesita (entre otras cosas), la información de los alumnos, profesores y asignaturas que existen en un determinado colegio.

Por lo anterior, es muy importante tomar en cuenta la forma en la que se almacenará y manipulará la información dentro de un sistema, al momento de planear su construcción; para esto se utilizan bases de datos.

La base de datos, es el lugar que contendrá la información y que definirá la forma en la que esta podrá ser consultada y manipulada; comúnmente se utilizan administradores de datos como SQL server, MySQL, Mongo DB, etc…, para almacenar los datos con los que trabajará el sistema.

La forma de conectarse con la base de datos desde algún proyecto de software, es desde la cadena de conexión, ya que esta contiene la información del servidor, la base de datos, y las credenciales para poder interactuar con la información dentro de la base.

Algunos de los problemas que con mayor frecuencia he encontrado en la capa de base de datos de los proyectos han sido:

- La duplicidad de información.
- Información basura en la base de datos.
- Información basura en las bases de datos del entorno de producción.

En cuanto a la duplicidad de información y la presencia de información basura dentro de la base de datos productivas, ambas situaciones suelen producirse principalmente cuando se realizan pruebas dentro de la base de datos de un entorno productivo, o cuando no se realizan depuraciones de las bases de datos una vez que se realiza alguna corrección dentro del funcionamiento del sistema.

Para los casos en que se encuentra información basura en la base de datos, lo que he realizado es evaluar qué tipo de información es la que no debería existir dentro de la base y construir las consultas necesarias para poder identificar correctamente esta información, para posteriormente eliminarla; este proceso necesariamente involucra el conocimiento de las reglas de negocio para poder identificar las relaciones adecuadas y por consiguiente la información que es realmente útil para la empresa.

En el caso de información duplicada dentro de la base, igualmente se diseñan las consultas necesarias para identificar la información duplicada dentro de las tablas de la base de datos. Un ejemplo de una consulta para identificar la información duplicada puede encontrarse en la ilustración 5

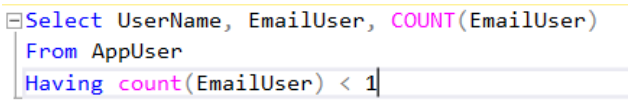

Ilustración 5: Ejemplo de consulta para encontrar registros con el mismo Email (registros duplicados)
Fuente: Elaboración propia, SQL server

Tomando en cuenta en el ejemplo anterior, que el email registrado en la base de datos puede pertenecer únicamente a un alumno. Una vez identificada la información que no debería estar dentro de la base de datos, se procede a su eliminación, al igual que en el caso de la información basura.

Front - End del sistema: Vistas y diseño web

El front-end, o las ventanas de interacción del usuario con el sistema, son quizá la parte más valiosa para el usuario final de la aplicación, y por ello requieren de la colaboración de varios perfiles para su realización, en este punto intervienen diseñadores y desarrolladores.

A pesar de ser una tarea de carácter técnico dentro del sistema, no se puede omitir la participación de diseñadores de interfaz y de experiencia de usuario, esto debido a que, como consultor, puedo desarrollar funcionalidades que faciliten tareas al usuario, el sistema puede ser adecuado a sus necesidades, rápido y seguir las mejores prácticas de codificación (en lo referente al software con el que se construya), sin embrago, si la interfaz y la experiencia de manipulación no son del agrado del usuario (el sistema no es fácil de usar o no despierta la curiosidad del usuario), este no utilizará el sistema, o tendrá muchas dificultades para hacerlo.

Es justo en este punto en el que he tenido la mayor cantidad de problemas, puesto que, en varios proyectos, no se tiene un diseño sobre el cual realizar la construcción de las vistas del sistema. Esto ha significado en muchas ocasiones retraso en la entrega de módulos cuya funcionalidad está lista, pero las vistas aún no, ya sea por cambios que solicita el cliente, o porque como desarrollador, elaboré las vistas como creí conveniente, pero al entregarlas el usuario, él esperaba una distribución de componentes en pantalla e interacción diferentes.

Para evitar estos inconvenientes, es necesario involucrar al equipo de diseño desde el principio del proyecto, y antes de iniciar la construcción del sistema como tal. Aunque esto no significa que la participación del equipo de diseño termine aquí, puesto que, en caso de surgir cambios por parte del cliente o inconvenientes en el desarrollo de las pantallas, se deberán realizar los ajustes necesarios en el diseño, para mantener la documentación en orden y para llevar un control adecuado de cambios.

Del lado de desarrollo, puede haber conflictos en la construcción de las pantallas, en el caso en que alguno de los elementos considerados en las vistas al momento de realizar el diseño, no se encuentran disponibles o no tienen la funcionalidad deseada. Esto se debe a que algunas de las herramientas utilizadas para el maquetado (diseño) de las pantallas de la aplicación, no necesariamente se encuentra actualizada con los controles que pueden utilizarse. Con esto me refiero a que, por ejemplo: El diseñador contempló un elemento de tipo tabla (grid), con una funcionalidad de ordenamiento y búsqueda en las columnas, accesible desde el encabezado de las mismas. Sin embargo, puede que alguna de estas funcionalidades no se encuentre disponible para el elemento de tipo tabla que el desarrollador utilice dentro de la aplicación.

Este inconveniente puede solucionarse si tanto desarrollador como diseñador, utilizan documentación oficial de elementos de construcción para las vistas. Por ejemplo, en el caso de que el proyecto utilice Angular, se puede recurrir a la documentación de Angular Material (https://material.angular.io/components/categories) para elegir los elementos que compondrán la vista.

Back - End del sistema: Servicios

Los servicios o “back-end” componen la parte lógica de un sistema; es aquí donde se implementan las reglas de negocio y se trabaja con la información proveniente de la base de datos. También es en este punto donde me he encontrado con la mayor cantidad de situaciones problemáticas y retadoras, por ejemplo:

1.- Las consultas, inserciones y modificaciones de la información a la base de datos se encuentran escritos en código duro (cadenas de texto) dentro de los archivos de código fuente de la aplicación, esto dificulta enormemente poder corregir una consulta, ya que es necesario correr la aplicación estableciendo puntos de interrupción que permitan extraer la consulta completa para analizar la causa del error o si es necesario modificarla para que recupere, inserte o elimine la información correcta. Además de hacer que el tamaño de los archivos de código sea innecesariamente grande.

2.- No se cuenta con catálogos que hagan más sencilla la manipulación de valores específicos para algunas variables; ciertos valores son asignados directamente dentro del código, por separado. Por lo cual, en caso de que se trate de una variable o valor que se ocupe en diferentes archivos fuente dentro del proyecto, si se tiene la necesidad de modificar este valor, es necesario modificar todos los archivos en los que es empleado.

3.- Uso inadecuado de sentencias de control: En varios proyectos, he notado que varios archivos contienen sentencias “if - else” que podían ser omitidas sin alterar la funcionalidad de la aplicación. La proliferación de este tipo de sentencias, cuando no son necesarias, representa una demanda ligeramente mayor para el desempeño de la aplicación en un

ambiente productivo, y más aún cuando son varias y se encuentran en varios archivos de la solución.

Del mismo modo, hay casos en que el manejo de sentencias de iteración, por ejemplo, sentencias de iteración del tipo “For”, pueden impactar negativamente si no son utilizadas adecuadamente, por ejemplo, al trabajar con una funcionalidad de actualización de datos en uno de los proyectos en una planta de energía. Una de las sentencias “for”, era ejecutada sobre cada uno de los registros de un elemento “DataTable”, lo que tenía como consecuencia que el tiempo de actualización de la información se incrementa enormemente (elevando el tiempo necesario para completar la actualización de 2 a 30 minutos). Ya que, la sentencia “for”, era ejecutada en cada celda de la tabla, para actualizar la información. Una mejor opción, era primero actualizar la información general y después insertarla en la tabla. Para evitar repetir la iteración del “for” en cada registro.

4.- Es común encontrar proyectos que no cuentan con un adecuado manejo de excepciones, por lo que, al ocurrir un error dentro del código, la información que se guarda en los registros (logs) destinados a los errores, en varias ocasiones es diferente al error generado. Por ejemplo: Teniendo dos funciones, donde la primera tiene un bloque “try catch” genérico y la segunda no. Si la primera hace el llamado a la segunda, y esta última genera un error, el registro de errores recibe la información especificada en el bloque de la primera función, y no el detalle del error generado en la segunda función.

5.- La falta de validaciones es otra situación común en varios sistemas, y esto puede ser fuente tanto de información relacionada erróneamente en la base de datos como en errores durante la ejecución del sistema, un ejemplo es la conversión o “cast” de valores

de las variables. Ya que, al intentar realizar una operación con un tipo de dato incorrecto, el sistema puede detenerse. Por ejemplo, al momento de leer la información de la base de datos, si el valor recuperado era DBNULL (no existe en base de datos) o NULL (referencia inválida), al momento de querer convertir el valor y asignarlo a una variable de tipo cadena, ocurre una excepción.

6.- A pesar de que cada una de las implementaciones y correcciones de errores dentro de los sistemas son idealmente probadas por un equipo de pruebas, el desarrollo de pruebas unitarias dentro del código, es responsabilidad de nosotros como desarrolladores e ingenieros de software.

Una prueba unitaria es un bloque de código que contiene la funcionalidad especifica de una acción en concreto, y su finalidad es la de comprobar que los métodos y la lógica implementados funcionan adecuadamente, estas pruebas contienen la información necesaria de los escenarios con los que trabajará la aplicación en un entorno productivo. De esta forma, si se agregan funcionalidades adicionales al código, bastará con modificar las funciones de prueba para validar estas nuevas funcionalidades antes de que pasen al equipo de pruebas.

Es común encontrar proyectos que no cuentan con pruebas unitarias que validen la funcionalidad con los posibles casos o escenarios que se puedan presentar, esto sobre todo en proyectos que han sido desarrollados por varios equipos a lo largo del tiempo y en proyectos que han sido realizados con muy poco tiempo para su implementación (esta suele ser una de las razones principales, necesitar un proyecto “para ayer”).

Ambientes del sistema

Debido a lo ya anteriormente presentado, un proyecto de software tiene diferentes fases en su construcción; después de concluir el desarrollo de algún módulo o funcionalidad, se pasa a la fase de pruebas y finalmente al cliente para que el usuario final pueda hacer uso del sistema. Para permitir que el flujo a través de esta serie de estados ocurra de la mejor forma posible, es importante considerar lo siguiente:

Un proyecto de software debe contar con tres ambientes, entendiendo un ambiente como una versión estable del sistema que se utiliza para un fin específico, es decir, un proyecto de software debe contar con varios ambientes, para llevar a cabo en cada uno las tareas necesarias para garantizar el correcto funcionamiento del sistema. Los ambientes que deben existir son:

- Desarrollo
- QA (Quality Assurance)
- Producción

Cada uno de ellos cumple con un fin específico dentro del ciclo de desarrollo.

El ambiente de desarrollo es aquel en el que los desarrolladores, pueden comprobar la funcionalidad del código que van construyendo, en este ambiente pueden hacer y deshacer cuantas veces necesiten alguna funcionalidad para asegurarse de que cumple con las especificaciones necesarias, comúnmente el acceso a este ambiente se realiza desde un servidor local instalado en la computadora del desarrollador.

El ambiente de QA, es el ambiente en el que los miembros del equipo de pruebas pueden ejecutar el plan de pruebas que se desarrolló al inicio del proceso de construcción; en este ambiente el sistema debe tener la funcionalidad que verá el cliente, es decir, debe cumplir con los flujos de negocio necesarios para llevar a cabo tareas específicas. Es en este ambiente en el que idealmente se deben detectar la mayor cantidad de errores posibles en el sistema; estos a su vez deben ser reportados y transmitidos al equipo de desarrollo para su corrección. Una vez que el equipo de desarrollo realiza las correcciones pertinentes, se actualiza el ambiente de QA con los nuevos cambios y se ejecutan nuevamente las pruebas hasta comprobar que el error ha sido corregido.

En lo que respecta al ambiente de producción, es un ambiente que idealmente está libre de errores, este ambiente es el que se entrega al cliente como resultado final de la construcción del sistema. Las actualizaciones al código de este son considerablemente menores que las realizadas en los dos ambientes previamente mencionados; esto debido a que la primer publicación (actualización de código) en este ambiente se realiza una vez que el sistema se encuentra terminado, o cuando se tiene un avance (previamente acordado), es decir, si en el calendario de entregas definido al inicio del proyecto, se establecen liberaciones parciales a medida que las funcionalidades van terminando de construirse, se realizarán tantas publicaciones en producción como ambientes o funcionalidades se hayan acordado en las entregas.

A pesar de que es importante contar con estos tres ambientes, me he encontrado con proyectos que no sólo no tienen esa estructura de publicación de cambios, sino que no cuentan con los tres ambientes; siendo el ambiente de QA el que en todos los casos que

he visto, no se toma en cuenta. Por esta razón, las actualizaciones de código y corrección de errores se realizan en el entorno de desarrollo y se prueban en el ambiente de producción, lo cual no es para nada aconsejable, ya que esto significa que el equipo de QA y el cliente están compartiendo el mismo ambiente y por lo tanto, todo error que el equipo de pruebas detecte lo verá el cliente, pero más importante aún, todo error que se encuentre en el código, afecta directamente las operaciones que el cliente o el usuario final realice dentro del sistema.

Depuración y optimización de código

En cada uno de los proyectos en los que he participado, siempre han existido mejoras que pueden hacerse al código. Estas mejoras bien pueden deberse a que alguna de las reglas de negocio se ha modificado y la forma original en que se implementó la funcionalidad del código ya no resulta ser la mejor forma de cumplir con lo requerido, o debido a la experiencia que proyecto tras proyecto se va adquiriendo y que puede aplicarse a proyectos nuevos.

Para dar solución a las áreas de oportunidad dentro del código de cada proyecto, y tomando en cuenta lo expuesto anteriormente, comúnmente comienzo con el análisis del sistema, comparando reglas de negocio y viendo el código que cumple cada una de ellas. Posteriormente verifico la programación y en caso de ser necesario, elimino las sentencias de control que no son necesarias y corrijo aquellos bloques de código sin referencia alguna o que se encuentran comentados dentro de la solución (en este último caso después de

asegurarme que el código comentado puede eliminarse sin problema, ya que en ocasiones este tipo de código corresponde a implementaciones previstas a fututo, o para atender algún error potencial), así como complementar las sentencias de manejo de excepciones dentro de los módulos y archivos de código.

En los casos en que la documentación dentro del código no es suficiente o no se tiene, procedo a revisar la información existente y la funcionalidad de los módulos para documentar en el encabezado de las funciones, la funcionalidad principal de cada una, así como los parámetros esperados en cada caso y a qué se refiere cada uno. Tener adecuadamente documentado el código facilita su lectura, mantenimiento y modificación a futuro, pues hace más comprensible el flujo de información dentro del sistema, y así es más fácil comparar la estructura con las reglas de negocio.

Dentro de la estructura de los archivos de código, es importante que cada uno contenga sólo las referencias a librerías (usings) necesarias para la correcta ejecución del código, ya que esto permite identificar de manera rápida los componentes referenciados dentro de un archivo.

Conclusiones

Mi labor profesional tiene un gran impacto dentro del ámbito en el que me desarrollo; no sólo en el sentido de realizar bien las labores que desempeño dentro de las organizaciones en las que colaboro, sino también en el nivel de responsabilidad que recae sobre el cargo al cual he centrado mi desarrollo profesional.

Como consultor de tecnologías de la información, los clientes acuden a mí no sólo para realizar el desarrollo de un sistema, desarrollar una aplicación o una página web. Acuden a mi buscando orientación respecto de alguna implementación, opciones de seguridad o simplemente respecto de las oportunidades de mejora sobre sus sistemas. Este último punto resulta ser el más complicado, puesto que, para llevarlo a cabo, es necesario contar con un panorama bastante amplio de conocimiento, tener una buena capacidad de análisis y tener la habilidad para proponer soluciones rápidas y efectivas.

Además del aspecto técnico, las habilidades interpersonales, la comunicación, empatía, y negociación juegan un papel muy importante en el trato diario con los clientes, pues sólo así es posible comprender correctamente las necesidades del contratante, realizar un plan de acción y ponerlo en marcha para lograr los resultados esperados.

Hasta el momento en que terminas tus estudios y entras al campo laboral, es cuando te das cuenta de que todas las herramientas que te dan dentro de la carrera existen en el plan de estudios por una razón, no importa si son materias técnicas o humanísticas, todas son necesarias, desde saber cómo funciona un sistema, hasta saber redactar un informe o un documento profesional.

Referencias.

- Lavín Iván (julio 14, 2016) *La consultoría a través del tiempo*. Recuperada en 19 de mayo de 2018 del sitio: http://www.milenio.com/firmas/ivan_lavin/consultoria-traves-tiempo_18_774102587.html
- David, Entendiendo MVC y sus problemas. Recuperado el 25 de octubre, 2018. De la página: https://medium.com/@davideng/entendiendo-m-de-mvc-y-sus-problemas-ebc0cbf518ec
- Montero Ortega, J.M, La arquitectura MVVM y sus componentes. Recuperado el 12 de mayo, 2021. De la página: https://openwebinars.net/blog/la-arquitectura-mvvm-y-sus-componentes/

Bibliografía.

- Icono Digital. (marzo 6, 2015). *Metodologías de desarrollo de software*. Recuperada el 22 de Agosto de 2016, de Icono Digital Sitio web: http://www.iconodigital.com/2015/03/06/metodologias-de-desarrollo-de-software/
- Earl Leonel, et al. Junio 8, 2018. *MVC Architecture*. Recuperado el 26 de octubre de 2018, de la página: https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture
- Romero Pérez J.E., *La externalización de actividades laborales (Outsorcing),* Recuperado el 26 de octubre, 2018, de la página: https://revistas.ucr.ac.cr/index.php/juridicas/article/view/13379
- Amazon, *¿Qué es DNS?*. Recuperado el 29 de septiembre de 2020, de la página: https://aws.amazon.com/es/route53/what-is-dns/
- Kaspersky. *¿Qué son los ataques DDoS?.* Recuperado el 29 de septiembre de 2020 de la página: https://latam.kaspersky.com/resource-center/threats/ddos-attacks
- Ccori Huaman, W. (septiembre 7, 2018), *10 Patrones comunes de arquitectura de Software.* Recuperado el 10 de mayo, 2021, de la página: https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b
- IBM. *IBM Cloud Products.* Recuperado el 10 de mayo, 2021, de la página: https://www.ibm.com/cloud/products
- Adictos al trabajo (octubre 7, 2012), MVC y MVVM. Recuperado el 10 de mayo, 2021, de la página: https://www.adictosaltrabajo.com/2012/10/07/zk-mvc-mvvm/
- Montero Ortega, J.M, La arquitectura MVVM y sus componentes. Recuperado el 12 de mayo, 2021. De la página: https://openwebinars.net/blog/la-arquitectura-mvvm-y-sus-componentes/