SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Apizaco

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

PLATAFORMA DE PRESUPUESTOS IMPLANT

TITULACIÓN INTEGRAL POR INFORME DE RECIDENCIA PROFESIONAL

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

PRESENTA:

JOSÉ MONTOYA GUZMÁN

NOMBRE DEL ASESOR:

M.C. MARÍA LORENA ROLDÁN FLORES

Instituto Tecnológico de Apizaco

APIZACO, TLAX., DICIEMBRE 2018

AGRADECIMIENTOS

Al comenzar un camino, es normal que pensemos en las dificultades que podremos encontrar, en mi opinión eso no es malo, porque estamos analizando y razonando las posibles consecuencias, sin embargo, la vida no es para quedarse analizando o para temer, analizar no debe ser algo que nos tome más tiempo del necesario, debemos actuar.

Tener confianza en mis objetivos, esforzarme por ellos, y mantener la disciplina es la lección más grande que mi periodo en la Universidad me dejo, agradezco profundamente a quienes me acompañaron en ese proceso.

Mis amigos, siempre dando alegrías y lecciones en el día a día, demostrando que la amistad es algo que vale por buenos momentos y sobre todo solidaridad.

Profesores, esas personas que nos daban curiosidad y motivación de aprender, la primera parte que impulsa a los profesionales.

Beeva Tec, gracias, siempre recordándonos que la tecnología no es sólo nuestro trabajo, sino nuestra pasión y que somos más que compañeros, una familia con un fin en común: servir, innovar y aprender.

Mi familia, especial mención a mi mamá, mi amiga de todos los días, importante en el ingreso a mi educación superior y crucial para continuar hasta el final.

"Dudar de ti mismo, es lo único que te puede frenar"

José Montoya

Introducción

El mundo de la tecnología implica retos constantes para empresas, profesionales e incluso clientes. El core del mundo moderno, son los sistemas de información, cada uno de los cuales, es desarrollado para cubrir necesidades específicas.

Si se piensa, más que innovación, la tecnología es parte de nuestra vida en cada momento, con sus beneficios y prejuicios como un ámbito social más. Plataformas sofisticadas analizadores de riesgos (machine Learning), modelos de aprendizaje, autómatas (inteligencia artificial), reconocimiento de patrones sociales (redes sociales), gestión de datos en empresas y negocios, aplicaciones de localización, plataformas educativas (MOOCs, por ejemplo), plataforma e-commerce y puntos de venta, servicios multimedia (streaming), etc. Por lo anteriormente, mencionado, es identificable que existen sistemas para todo tipo de empresas.

Este proyecto se centra en un sistema gestor de información. Haciendo mención de la empresa en concreto donde se llevó a cabo el proyecto, el Grupo Financiero BBVA Bancomer cuenta con distintos departamentos, para dedicarse a áreas, proyectos o tareas espaciales. Siendo la demarcación de Implant donde se desarrolló el proyecto.

Implant es el departamento que gestiona el presupuesto de los proyectos tecnológicos del banco. Entre los datos que maneja, se tienen: costos, recursos, licencias, periodos, fases, medios de pago y tarifas. Dichas entidades eran manejadas por sistemas independientes, que además gestionan los datos de diferente forma. Por lo cual, hay una parte de la gestión de proyectos/presupuestos donde los datos se manejan manualmente. Una vez identificadas las problemáticas y ambigüedades mencionadas, surge la necesidad del proyecto.

Se requiere volver administrable el presupuesto de proyectos/presupuestos con el fin de mantener la integridad de la información y poder optimizar los recursos del área.

ÍNDICE

CAPÍTULO 1. GENERALIDADES DEL PROYECTO

1.1 RESUMEN

El presente documento pretende explicar los fundamentos del proyecto *Plataforma de Presupuestos Implant.*

El proyecto nace a raíz de Beeva Tec, una empresa con origen en España, dedicada a proveer soluciones de software con calidad. Concretamente en México, se ha dedicado a ofrecer a empresas, software que cause impacto y atienda necesidades. Entre sus clientes más destacados se encuentra BBVA Bancomer, pero, ¿Qué es Implant? ¿Por qué un software de presupuestos? Implant es el área dedicada a la gestión de proyectos, recursos, servicios, costos y pagos para los productos de tecnología externos que utiliza la empresa BBVA Bancomer.

Anteriormente, Implant manejaba su gestión de proyectos mediante distintos softwares y que de manera conjunta proveían una solución. Sin embargo, no solo no eran homogéneos, sino además, eran independientes, lo cual no hacia transparente las labores, ni aseguraba la integridad de los datos, como se podría pensar.

La finalidad del Proyecto es proveer una herramienta web donde se puedan realizar las actividades necesarias con una gestión integral, segura, eficiente y manejable, esto, mediante una arquitectura orientada a servicios, que lo mantenga como un sistema escalable, débilmente acoplado, bien estructurado y con el uso de tecnología cloud para que el cliente reciba lo que necesite cuando lo necesite.

1.2 Descripción de la empresa y área de trabajo

Beeva Tec Operadora S.A. de C.V., es una empresa que actúa como proveedor de servicios tecnológicos, fue constituida en España en 2012. Es un gran colaborador de partners como Google, Amazon Web Services, Azure, Mongo y Cloudera. Así mismo, cuenta con clientes como, Grupo Financiero BBVA Bancomer, gasNatural Genosa, ferrovial, Ingenico Group, viesgo, 8belts, MRM//McCANN, Dia%, etc.

En cuanto a sedes, Beeva maneja 4 sucursales en la Ciudad de México:

1. **Oficinas generales de Beeva:** con ubicación en Paseo de la Reforma 403, Delegación Cuauhtémoc, 06500.
2. **Torre BBVA Bancomer:** en Paseo de la Reforma 510, Delegación Juárez, 06600.
3. **Parques Polanco:** calz. Gral. Mariano Escobedo 303, Delegación Miguel Hidalgo 11520.
4. **Centro de Procesamiento de Datos:** avenida 3, Mz 6, Lt 2, Lago Esmeralda, Atizapán.

Cabe mencionar que en lo que se refiere a las sedes del banco, Beeva tiene colaboradores en diferentes departamentos y asignados a distintos proyectos. Dado que BBVA cuenta con muchas áreas operativas y consultores de otras empresas un mismo equipo de trabajo es conformado por elementos de diferentes consultoras.

El proyecto del que hace mención el documento, “Plataforma de Presupuestos Implant”, está pensado para su implementación en la sucursal de Parques Polanco, en el departamento del mismo nombre, con área de desarrollo en las sedes de Parques y en las instalaciones de Beeva México.

1.3 Problemas a resolver

El departamento de IMPLANT requiere de una herramienta que sea un sistema web, donde los empleados del área puedan llevar la gestión de los proyectos, recursos, medios de pago, y tarifas que administra el banco para productos de software. Entre algunas de las problemáticas que debe resolver la plataforma están:

- Seguridad de acceso mediante autenticación de google y corporativa.
- Este sistema no puede reemplazar a los sistemas que dan entrada de datos, más bien debe acoplarse e integrarlos.
- Las salidas del sistema deben ser en distintos formatos para su uso posterior por otras aplicaciones.
- Deben existir diferentes roles operativos y validación de permisos de acceso a módulos o elementos de acuerdo a ello.
- Ser un servicio que se ejecute en serverless (cloud computing) con el fin de brindar escalabilidad y respaldos, con cortos tiempos en despliegue.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar una plataforma web en la que se pueda controlar de manera organizada, integra y segura los procesos de contratación, asignación y seguimiento de recursos (empleados), presupuestos y proyectos de software de la empresa BBVA Bancomer. La plataforma funcionará en un entorno cloud de Amazon Web Services.

1.4.2 Objetivos específicos

- **Ingesta de información a la base de datos:** módulo de carga de datos desde documentos en formato .xls/.xlsx a bases de datos SQL para las entidades PEP, recursos, tarifas y proyectos.
- **Registro de proyectos:** módulo de registro de proyectos, para sus 5 tipos: cerrado, tarifario, servicios, servicios extendidos y licencias.
- **Actualización de proyectos:** módulo de seguimiento/actualización de proyectos antes mencionados.
- **Consultar entidades:** módulo de consulta de entidades Proyectos, Recursos, Proveedores, PEPs e ITZ de la base de datos, con filtros para la búsqueda de resultados.
- **Registrar usuarios:** módulo de registro y edición de usuarios y asignación de roles dentro de la aplicación (Administrador, Líder e Implant).
- **Validar sesiones:** autenticación de usuarios mediante credenciales de google y de la aplicación.
- **Validar permisos:** restricción de opciones de acuerdo a los permisos de usuario.

1.5 Justificación

IMPLANT es un departamento de BBVA dedicado a la administración de los recursos económicos que tiene la empresa para productos de software. Durante cierto tiempo, se han contado con herramientas de software de difícil uso e independientes entre sí, es decir; varias aplicaciones atienden distintas funcionalidades y no una sola, es la que gestiona este mismo proceso.

Lo anterior provoca las siguientes problemáticas:

- Al manipular la información con distintas herramientas (no acopladas entre sí) se genera inconsistencia de datos y metadatos.
- Al no ser plataformas dedicadas, los tiempos de respuesta no son eficientes.
- Es complicado tener un registro puntual de la trazabilidad de los procesos, algunas terminan quedando en control completo del usuario y al ser gran volumen de información puede perderse la integridad de la información.
- No hay un sistema de consulta integral del estatus de los datos de las entidades que se manejan en el proceso.
- Si se producen errores no es posible identificar en que parte del proceso se suscitó.

Debido a estas razones, se requiere realizar finalmente un sistema integral donde se tenga el control y seguimiento de este proceso y sus reglas de negocio, esto, en un tiempo de respuesta eficiente. Con ello, podrán eliminarse todas las ambigüedades que actualmente generan labores innecesarias e inconsistencia de datos, de esta manera se salvaguarda la información y se optimizan recursos.

1.6 Alcances y Limitaciones

1.6.1 Alcances

Brindar un sistema web que permita la carga de datos desde documentos xls/xlsx, gestión de usuarios, registro y seguimiento de proyectos, recursos asignados, presupuestos y tarifas. Debe haber las consideraciones pertinentes para que el proyecto funcione en una plataforma cloud con el debido aprovechamiento de los recursos, óptimo tiempo de ejecución y seguridad de acceso.

1.6.2 Limitantes

Existencia de datos y metadatos pertenecientes a otras herramientas de software del banco, debido a que se utilizan en formatos específicos, condicionan las entradas y salidas de la plataforma de presupuestos de IMPLANT.

CAPÍTULO 2 MARCO TEÓRICO

2.1 Acerca de Beeva

2.1.1 ¿Quiénes somos?

Somos una compañía tecnológica y de innovación. No queremos ser una compañía de TI más, así que cuando la tecnología se convierte en commodity, nosotros buscamos nuevos retos.

2.1.2 Misión

Somos un equipo de tecnología y de innovación de la compañía Blue Indico, que tiene como misión ayudar a las empresas en su transformación digital y aportar valor en el momento emergente de las tecnologías. Nuestro core se centra en cloud computing, big data y machine intelligence.

Sobre todo y concretamente, trabajar con BBVA para transformar el mundo financiero a través de la tecnología.

2.1.3 Visión

Damos mucha importancia a la innovación y lo consideramos una forma de entender la tecnología y nuestro trabajo. A través de BEEVA Labs, aportamos investigación sobre nuevas tecnologías al área IT, realizamos proyectos de innovación internos y para clientes, y diseñamos productos basados en tecnologías que testamos y cuya viabilidad evaluamos, esto, para lograr:

- Ser la mejor empresa para los que aman la tecnología.
- Reinventar con BBVA la industria financiera a través de la tecnología.
- Ser un referente para la comunidad por nuestro expertise tecnológico.

2.1.4 Valores

Personas

Somos un equipo formado por buenas personas que colaboran y comparten con generosidad, humildad, honestidad y respeto. Transmitimos pasión y energía positiva en todo lo que hacemos.

Impacto

Entendemos la tecnología como un medio para crear valor para el negocio y nuestros clientes. Somos empáticos y nos ponemos “en los zapatos” del otro para entender el problema a resolver. Somos ágiles porque nos centramos en entregar valor y adaptarnos al cambio con un aprendizaje continuo.

Innovación

Aprendemos de forma continua y exploramos nuevas formas de impactar a través de la tecnología, sin que nos frene el miedo al error. Somos OPEN, contribuimos y participamos en la comunidad. Somos rigurosos y nos ponemos el listón siempre muy alto, anticipamos y nos involucramos tomando decisiones con responsabilidad.

Confianza

Las buenas prácticas, la calidad y los procedimientos ágiles y flexibles, son parte de la transformación que llevamos a las compañías. Aplicamos lo mejor de nosotros, lo que hemos testado y funciona. Tenemos la confianza de que damos al cliente lo que querríamos para nosotros y que, parte de nuestro valor, es ofrecer la solución que mejor se adapta en cada momento.

Compromiso

Entregar y cumplir con los objetivos es algo que demuestra los valores del personal, para que tanto en lo individual, como en lo colectivo, se transmita la confianza en el grupo, para con los clientes.

Crecimiento

Aprender, conocer las nuevas tecnologías y compartir el conocimiento es uno de los temas que más se valoran, por ello la empresa a través de cursos, certificaciones y proyectos ninja busca fomentar el crecimiento de cada miembro del equipo de trabajo día a día.

2.1.5 Especialidades

Cloud Computing

Apostamos por la extensión del Centro de Procesamiento de Datos (CPD) hacia una arquitectura de Cloud híbrida, manteniendo los procesos más críticos y transaccionales en el CPD propio e integrando los servicios adicionales, o con necesidades especiales, en el cloud.

Nuestros socios son los principales proveedores de Cloud Computing de escala mundial, como AWS (Amazon Web Services) y Microsoft Azure.

Big Data

Somos expertos en plataformas cloud computing como AWS y Azure, lo que nos permite realizar proyectos big data de manera ágil y a un coste reducido. Además, trabajamos con las tecnologías big data, NoSQL y MPP más extendidas, como Hadoop, Spark o Kafka.

Utilizamos las tecnologías de hoy, pero nuestra filosofía de empresa nos obliga a ir siempre un paso por delante para ofrecer a nuestros clientes las soluciones más óptimas en cada momento.

Machine Learning

Nos apoyamos en las principales herramientas y librerías open source, como Spark/MLlib, y los ecosistemas de R y Python.

Aplicamos sistemas de aprendizaje automático a distintos casos de uso: extracción de información relevante de documentos o redes sociales, sistemas de recomendación y personalización de contenidos o asistentes virtuales. Además, seguimos profundizando en otros casos de uso más clásicos relacionados con la inteligencia de negocio y la minería de datos.

2.1.6 Certificaciones y Reconocimientos

- Más de 100 certificaciones en Amazon Web Services.

- 16 Solutions Architect con la máxima certificación de Amazon Web Services.
- Más de 50 certificaciones para desarrollar en Google Cloud Development.
- Primera compañía española en obtener la competencia Big Data de AWS en 2013.
- Primera empresa europea en ser partner del Data Warehouse de AWS, Amazon Redshift.
- Una de las 90 empresas en el mundo en contar con la certificación Managed Services Partner (MSP) de AWS desde 2016.
- Pioneros en la implementación del Cloud en proyectos desde 2011. Gestionando proyectos Big Data desde 2012.
- Hemos tenido un crecimiento del 42% en el último año y multiplicado por cuatro la plantilla en los últimos tres años.

2.2 Desarrollo web

2.2.1 Concepto

Desarrollo web significa construir y mantener sitios web, es el trabajo que tiene lugar en un segundo plano y que permite que una web tenga una apariencia impecable, un funcionamiento rápido y un buen desempeño para permitir la mejor experiencia de usuario.

Para diferenciar correctamente Diseño de Desarrollo web, veamos las siguientes definiciones.

Diseño web

Determina la apariencia. Que cubre el diseño, la navegación y los colores de un sitio web (También puede incluir el diseño gráfico y logo), está más preocupado por la estética y la experiencia del usuario que de las funciones. Un diseñador de páginas web crea sitios fáciles de usar y adecuados para su propósito.

Miscret (2012) lo define como:

> El diseño web es una actividad que consiste en la planificación, diseño e implementación de sitios web. Un diseñador web tiene que ver con cómo crear y desarrollar una página web así también como los clientes interactúan

> con ella. Los buenos diseñadores web saben cómo poner juntos los principios de diseño para crear un sitio que se vea muy bien. También entienden acerca de la usabilidad y cómo crear un sitio que los clientes quieren navegar alrededor de porque es tan fácil de hacer. (párr. 1)

Desarrollo web

Se trata de la programación de servicios de fondo y no el rostro de un sitio web. Funciones previstas en el desarrollo web incluyen el registro, los sistemas de gestión de contenidos, comercio electrónico y las aplicaciones de base de datos, además de permitir a los visitantes interactuar en un sitio web.

Mercedes (2017) lo refiere como:

> Desarrollo web significa construir y mantener sitios web; es el trabajo que tiene lugar en un segundo plano y que permite que una web tenga una apariencia impecable, un funcionamiento rápido y un buen desempeño para permitir la mejor experiencia de usuario. Los desarrolladores web son como duendes con poderes: nunca los ves, pero son los que hacen que todo esté bien y funcione de manera rápida y eficiente. (párr. 1)

2.2.2 Arquitectura: modelo de capas de un sistema web

Las plataformas web son desarrolladas por programadores y arquitectos de software que define la separación de la lógica mediante capas. Derivado del surgimiento de nuevas arquitecturas, han surgido abstracciones del modelo MVC, por ejemplo, la figura 2.2.2 muestra un diagrama de componentes de MVC en una arquitectura tipo REST, que, en la actualidad, es la forma en comúnmente funciona la web.

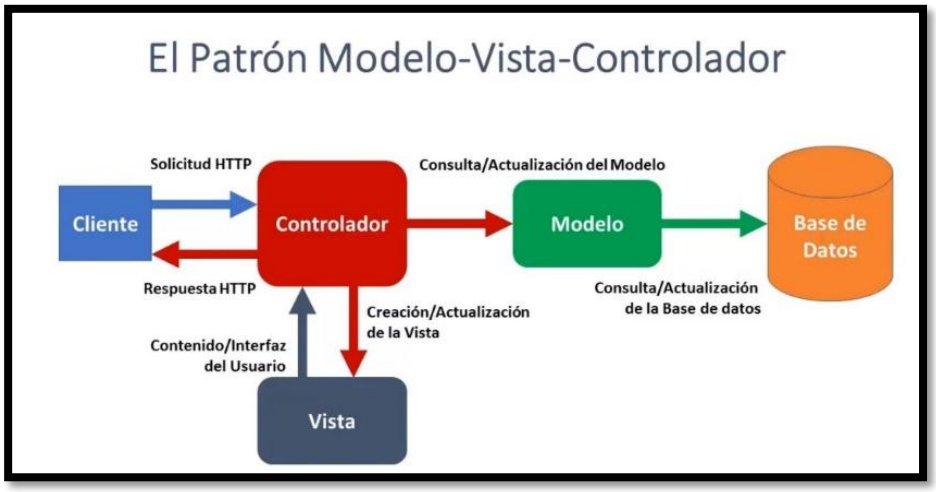


Figura 2.2.2.1: Modelo vista controlador tipo REST.

Arias, A. (2017). Curso ASP.NET MVC [Diagrama]. Recuperado de https://www.udemy.com/curso-completo-de-desarrollo-asp-net-mvc-5/learn/v4/t/lecture/7040438?start=0

Modelo vista controlador

Modelo-vista-controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

DavidEnq (2016) lo resume como:

> El concepto original de MVC nos define tres capas y cada una de ellas tiene asignado una tarea. Esto permite que cada capa sea “fácil de entender”, mantener y se puedan añadir nuevas características, y los desarrolladores y diseñadores puedan trabajar de forma simultánea sobre su capa correspondiente, evitando así el código espagueti. (párr. 3)

1. **VISTA**

También podemos considerar a esta capa como el front end de la aplicación. Es lo que el cliente puede visualizar de un sitio web, a nivel técnico se compone de la interfaz de la página, vista en el navegador.

- **Cliente:** es el navegador web, quien es el encargado de "consumir" y traducir la página a una vista gráfica, encargándose de leer el código HTML, CSS y JavaScript.
- **Solicitudes HTTP:** son peticiones generadas por el cliente para obtener algún resultado del back end.

2. **Controlador**

Es la capa intermedia entre vista y modelo, generalmente el controlador es la entrada al back end, pero puede encontrarse a nivel de código tanto en front end como en back end o en ambos. Se encarga de compartir datos entre el modelo y la vista.

3. **Modelo**

Encargado de enviar datos hacia el controlador, el modelo es propiamente el backend conformado del:

- **Modelo de datos:** código de lenguaje del lado del servidor donde se modelan las entidades de datos y trabaja la lógica de negocio.
- **Base de datos:** esquemas relaciones o no relacionales de datos, donde se almacena la información de la empresa.

2.2.3 Fases del desarrollo web

Al igual que la construcción de otros tipos de software, los pasos para la construcción de sitios web pueden variar de acuerdo a las metodologías que manejen las empresas, o los equipos de programadores, sin embargo; a continuación se mencionan algunos que son básicos y que como mínimo deberían considerarse en el desarrollo web.

1. **Investigación**

Lo primero que debe hacerse es un análisis y una revisión del contexto en el que se trabaja. Por ejemplo: mirar sitios similares (competencia), buscar un estilo que nos identifique y que por supuesto nos guste y con el que nos sintamos cómodos.

2. **Planificación**

Cuando se planifica un sitio web, se deben considerar varios factores: la audiencia y sus necesidades, el objetivo para el que se crea, los temas y contenidos que le gustaría cubrir, el nombre del sitio, la disponibilidad del nombre de dominio y registrarlo, lo que otros sitios han hecho, etc.

3. **Arquitectura del sitio y contenido**

Piense qué va a contar y cómo va a contarlo. Crear un mapa de la arquitectura del sitio para demostrar visualmente cómo se organiza el contenido y la estructura del sitio. Wireframes o mockups, es lo común a desarrollar en esta fase.

4. **Diseñar, construir y hacer pruebas**

Es la etapa del diseño web: tipos de letras, colores, plantillas, imágenes, títulos, etc. En esta etapa se empieza a probar el sitio web.

5. **Operar, mantener y evaluar**

En esta etapa el sitio Web se encuentra constantemente en mantenimiento para posibles mejoras. Se procura obtener informes sobre el rendimiento, para asegurar que el sitio Web sea un éxito.

6. **Marketing**

Es importante mencionar que no todos los sitios requieren de esta fase, ya que hay productos web que son utilizados únicamente en redes corporativas internas, como sistemas de administración de la información o puntos de venta.

Estar presente en Internet es muy importante hoy en día para las empresas. Tener una página web corporativa de calidad y actualizada es básico para posicionarse en

el mercado, darse a conocer, conseguir clientes, mejorar su reputación y, en definitiva, vender sus productos o servicios y ganar dinero

Tipos de desarrolladores

El mundo del desarrollo web es muy amplio y para incorporarse al área laboral, es importante escoger en qué especializarse para poder elegir trabajo. Se mencionarán, los tipos de desarrolladores web y de sus perfiles que podemos encontrar en la actualidad, tales como:

- Desarrolladores Front end.
- Desarrolladores Back end.
- Desarrolladores Full stack.

En el próximo capítulo se hará mención acerca de estas tres áreas y los conocimientos que debe tener un desarrollador para cada una de ellas.

2.2.4 Front end

Front end que en su traducción significa frente o fachada final. Es una especialidad para el desarrollo web, que trabaja la interfaz web y hace que el usuario pueda interactuar con nuestra web.

Chapaval (2018) lo resume “Es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios”. (párr. 2)

Está orientado a lenguaje de marcas y al lenguaje de programación web de ejecución en equipos clientes, sin necesidad de uso de servidores externos. Casi todo lo que se muestra en la pantalla cuando accedes a una web es desarrollo front end, la estructuración de los apartados, tamaños, márgenes entre estructuras, tipos de letra, colores, adaptación para distintas pantallas, los efectos de ratón, teclado, movimientos, desplazamientos, efectos visuales. Esto sería la base origen en la que se centra la especialidad front end, dar formato a contenidos, desarrollo del aspecto de la web y manipular resultados de datos obtenidos.

Un desarrollador front end suele trabajar de la mano con un diseñador web, para que los aspectos visuales/funcionales sean eficaces y agradables para los usuarios. A continuación se mencionan algunos aspectos que debe conocer un desarrollador front end.

Conocimientos técnicos

Al ser los encargados de diagramar la estructura semántica del contenido, codificar el diseño en hojas de estilo y agregar las interacciones con los usuarios, deben contar con un conjunto variado de habilidades y conocimientos, algunos de estos son:

- Contar con habilidades en HTML5 y CSS3.
- Conocimiento en JavaScript.
- Ser muy creativo para lograr visualizar las animaciones, transiciones y cambios en la aplicación del estilo visual de código.
- Dominar los estándares para la construcción de HTML, dictados por la W3C.
- Tener conocimiento de diseño y manejar los elementos visuales de un sitio web.
- Entender el trabajo del diseñador web y del desarrollador back end manejando los conceptos de usabilidad, accesibilidad y experiencia de usuario.

Responsabilidades

El trabajo del desarrollador front end comienza una vez que se ha definido y aprobado el diseño final del proyecto, entonces se encargará de:

- Traducir el diseño de un sitio a código HTML y CSS.
- Estructurar el contenido de forma semántica.
- Asegurar la accesibilidad de los sitios.
- Controlar las tipografías, plantillas, formas del diseño y la interactividad.

2.2.5 Back end

Back end es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la

aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas.

Chapaval (2018) define Back como:

> Back end es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas. (párr. 4)

Algunos de los lenguajes de programación de Back end son Python, PHP, Ruby, C# y Java, y así como en Front end, cada uno de los anteriores tiene diferentes frameworks que te permiten trabajar mejor según el proyecto que se esté desarrollando.

Un programador en esta capa, es profesional fundamental en los proyectos digitales, responsable de la programación de lógica de negocio, funcionalidades, bases de datos y servidores web, evitando problemas en las capas más profundas del proyecto. En su día a día trabaja con lenguajes del lado del servidor como PHP, Ruby, Python y con bases de datos relacionales del tipo SQL o no relacionales como MongoDB. Además, maneja JavaScript en el lado del browser, como un puente entre la interfaz y el motor del desarrollo.

Conocimientos técnicos

Contrario a lo que se podría pensar, las habilidades del desarrollador back-end están más relacionadas con la lógica y el diseño de soluciones que con conocimientos técnicos. Para lograr un buen trabajo, el profesional debe tener:

- **Capacidad de abstracción lógica:** debe visualizar el inicio y final de acciones y plantear los recorridos posibles para luego determinar la estrategia más eficiente.
- **Conocimientos y capacidad de estudio de lenguajes del lado del servidor como** PHP, NodeJS, Python, Java o Golang.

- **Manejo de al menos un framework** como Spring, Express o Django.
- **Conocimientos básicos de configuración de servidores web:** para poder adaptar elementos como la capacidad de memoria, la capacidad para subir archivos, el nivel de demanda e instalar librerías como APC (para el cache) y GD (para la manipulación de imágenes). SI bien no es imperativo, pero estos conocimientos son bien valorados.

2.2.6 Full stack

Un desarrollador web Full Stack es alguien que puede desenvolverse bien en ambas partes de una aplicación, tanto Front end (parte visual) como Back end (parte lógica). El Front end es conocido como la parte que interactúa y ven los usuarios, mientras que el Back end es la parte que maneja la lógica del negocio como conexiones de Bases de datos, Autenticaciones de usuario, configuraciones de servidor, etc. Ser Full Stack no significa que tienes que ser un maestro en cualquier tecnología de las partes implicadas, más bien significa que se tiene que tener un contexto global de como para trabajar con ambos lados de una aplicación y entender lo que hace, lo que se va hacer y cómo funciona cada parte del proceso de construir una aplicación.

Conocimientos técnicos

Además de conocimientos de Front end y Back end, un desarrollador Full stack debe tener otras habilidades, ya que estará inmerso en todo el proceso de desarrollo web.

- **Capacidad de escuchar y entender requerimientos del cliente:** debe identificar sus problemas y ofrecer soluciones, sin ponerle barreras.
- **Eficiencia y proactividad bajo presión:** al ser el último eslabón de la cadena, debe enfrentar todos los problemas derivados de las etapas anteriores, manteniendo una actitud analítica que le permite encontrar soluciones de forma ágil y creativa.
- **Capacidad para trabajar en equipos multidisciplinarios:** esto le permitirá entender y aportar en todos los procesos previos a su trabajo.
- **Habilidad de ser ordenado para trabajar:** tener el hábito de comentar código, escribir manuales y trabajar con estándares, entendidos como

> acuerdos de los métodos definidos por los equipos para comunicarse con códigos y trabajar siguiendo la misma lógica.

Romo (2015) concluye que:

> Bueno, un full-stack developer es un programador con un perfil técnico muy completo. Es alguien que conoce todas las capas del desarrollo de software y también debe conocer la manera de conectar cada capa con las otras. De esta manera, debe ser capaz de levantar una aplicación desde cero. (párr. 2)

2.3 Arquitectura Orientada a Servicios

2.3.1 Concepto

Como se mencionó en el tema anterior las arquitecturas de tipo monolítico o las que se basaban en dos o tres capas de abstracción han ido quedando en el pasado, esto debido a la necesidad de integración de aplicaciones que permita a las organizaciones unir los objetivos de negocio, y por tanto, optimizar los procesos.

En la actualidad, debido a los competitivos mercados globales, las compañías se ven presionadas a responder de la manera más efectiva. Saber actuar ante los cambios que afectan de manera natural a los negocios, optimizar los procesos, reducir los costos de TI, y lograr la flexibilidad son algunos de los factores claves para la competitividad y el crecimiento de las organizaciones.

Para lograr estos objetivos es necesario potenciar los recursos de TI, que deben estar enfocados en proporcionar sistemas más flexibles, pero integrados, poder interconectar los procesos, personas e información, tanto con la misma organización como con subsidiarias y socios comerciales.

Para lograr ello se necesita de una herramienta basada en estándares para integrar sistemas y aplicaciones heterogéneos, sobre una serie de plataformas y protocolos de comunicación con una metodología bien establecida, para lograr un nivel óptimo de integración, de manera que la infraestructura facilite los cambios posteriores que puedan surgir como respuesta a la evolución en las necesidades de la empresa, esto es puede lograse mediante una arquitectura orientada a servicio.

SOA (Arquitectura orientada a servicios)

Marsilli (2007) define:

> La arquitectura orientada a servicios (SOA) no se trata de software o de un lenguaje de programación, SOA es un marco de trabajo conceptual que permite a las organizaciones unir los objetivos de negocio con la infraestructura de TI integrando los datos y la lógica de negocio de sus sistemas separados. (párr. 4)

Esto permite la reducción de costos de implementación, innovación de servicios a clientes, adaptación ágil ante cambios y reacción temprana ante la competitividad, permitiendo, que los componentes del proceso se integren y coordinen de manera efectiva y rápida.

2.3.2 Características

No hay estándares en relación a la composición exacta de una arquitectura orientada a servicios, pero algunos de los principios más importantes y más habitualmente aplicados son los siguientes:

- **Contrato de servicios estandarizados:** implica la adhesión a un acuerdo de comunicación en virtud del cual se procede a definir y describir los servicios, tanto en conjunto como con mayor nivel de detalle.
- **Abstracción de servicios:** esta conceptualización no se refiere a su usabilidad sino a la lógica que hay detrás de cada servicio.
- **Reutilización de servicios:** en busca de la economía de desarrollo y mantenimiento, la lógica se divide en servicios con la intención de promover su reutilización.
- **Autonomía de servicios:** este principio es de aplicación a las fases de diseño y ejecución y hace referencia al control que los servicios ostentan sobre la lógica que encapsulan.
- **Descubrimiento de servicios:** con la eficacia como meta última, los servicios se complementan con los metadatos mediante los cuales se pueden descubrir e interpretar las distintas oportunidades disponibles.
- **Transparencia de ubicación de servicios:** se refiere a la capacidad que tiene un consumidor de servicios para invocar a un servicio,

independientemente de su ubicación en la red. Este principio se articula en torno al reconocimiento de la propiedad de descubrimiento y el derecho de un consumidor para acceder al servicio. También puede interpretarse en términos de virtualización de servicios, que aplicarían en caos en que el consumidor simplemente llama a un servicio lógico, mientras que un SOA habilita la ejecución del componente de la infraestructura, normalmente un bus de servicios, que mapea este servicio lógico y procede a ejecutar la llamada al servicio físico.

2.3.3 Componentes Arquitectónicos

Existen varios componentes de SOA, podemos clasificarlos principalmente en capas de software, elementos de la arquitectura y servicios web, para efectos de este capítulo, nos centraremos en los dos primeros.

Capas de software

Se refiere a la abstracción del stack que componen SOA desde la solicitud de información, hasta desde donde se almacena y sirve hacia la vista de una aplicación:

- **Aplicaciones básicas:** sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- **De exposición de funcionalidades:** donde las funcionalidades de la capa aplicativa son expuestas en forma de servicios (generalmente como servicios web).
- **De integración de servicios:** facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.
- **De composición de procesos:** que define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio.
- **De entrega:** donde los servicios son desplegados a los usuarios finales.

Elementos de la arquitectura

Para que un proyecto SOA tenga éxito los desarrolladores de software deben orientarse ellos mismos a esta mentalidad de crear servicios comunes que son orquestados por clientes o middleware para implementar los procesos de negocio.

Cuando se habla de una arquitectura orientada a servicios están hablando de un juego de servicios residentes en Internet o en una intranet, usando servicios web. Existen diversos estándares relacionados a los servicios web, como XML, HTTP, SOAP, REST, WSDL, etc.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La siguiente imagen nos muestras los elementos que componen SOA:

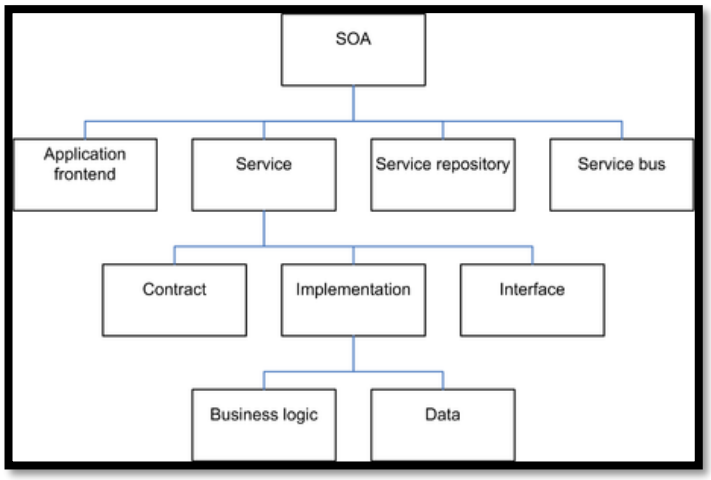


Figura 2.3.3.1: Elementos de la arquitectura SOA.
Bieberstein, N. (2006). SOA [Diagrama]. Recuperado de Service-Oriented Architecture Compass, Pearson 2006.

2.3.4 Ventajas y desventajas

Ventajas

SOA es una manera de diseñar e implementar los procesos de negocios, obteniendo una gran flexibilidad en su mantención y evolución. Una de las principales características de SOA es que resuelve los problemas de conectividad

y permite una real reusabilidad masiva y una gran independencia de las plataformas, rentabilizando las ya existentes.

Podemos citar como ventajas de SOA:

- Reduce el nivel de acoplamiento.
- Clara definición de roles de desarrollo.
- Definición de seguridad más clara.
- Fácil testeo.
- Mejora la mantención.
- Favorece la reutilización.
- Favorece el desarrollo en paralelo.
- Permite fácil escalabilidad.
- Permite un mapeo directo entre los procesos y los sistemas.
- Permite un monitoreo preciso.
- Permite la interoperabilidad.

Desventajas

Un buen SOA se basa en la implementación de estándares. Sin estándares, la comunicación entre aplicaciones requiere de mucho tiempo y además, puede comprometer la integridad de los sistemas.

SOA no es para:

- Aplicaciones con alto nivel de transferencia de datos.
- Aplicaciones que no requieren de implementación del tipo request/response.
- Aplicaciones que tienen un corto periodo de vida.
- Incrementalmente se hace difícil y costoso el ser capaz de cumplir con los protocolos y hablar con un servicio.
- Implica conocer los procesos del negocio, clasificarlos, extraer las funciones que son comunes a ellos, estandarizarlas y formar con ellas capas de servicios que serán requeridas por cualquier proceso de negocio.
- En la medida en que un servicio de negocio, vaya siendo incorporado en la definición de los procesos de negocio, dicho servicio aumentará su nivel de

criticidad. Con lo cual cada vez que se requiera efectuar una actualización en dicho servicio.

2.4 Servicios web

2.4.1 Concepto

Como se mencionó en el capítulo anterior, vamos a entrar en detalle de que es un servicio web. Un servicio es una aplicación que desempeña una actividad de negocio, la cual proporciona una interfaz que puede llamarse desde otro programa, se registra y se localiza por medio de un servicio de registro. De lo anterior, se puede definir formalmente.

Los servicios web son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar sobre http. Estas intercambian datos entre sí con el objetivo de ofrecer información. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios los solicitan llamando a estos procedimientos a través de la web. A su vez, proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario.

Lazaro (2018) lo sintetiza como:

> Un Web Service, o Servicio Web, es un método de comunicación entre dos aparatos electrónicos en una red. Es una colección de protocolos abiertos y estándares usados para intercambiar datos entre aplicaciones o sistemas. Las aplicaciones escritas en varios lenguajes de programación que funcionan en plataformas diferentes pueden utilizar web services para intercambiar información a través de una red. (párr. 1)

Características

Las características principales de los servicios Web, son las siguientes:

- Utilización de estándares de internet.
- Basados en tecnologías de paso de mensajes, la interacción entre el cliente y el proveedor del servicio.
- Combinan lo mejor de la tecnología de componentes y de la tecnología Web.

Los servicios Web presentan una funcionalidad de caja negra que puede ser reutilizada sin preocuparse de cómo es implementada y ello proporciona interfaces bien definidas.

2.4.2 Tipos de servicios web

A nivel conceptual, un servicio es un componente software proporcionado a través de un end point accesible a través de la red. Los servicios productores y consumidores utilizan mensajes para intercambiar información de invocaciones de petición y respuesta en forma de documentos auto-contenidos que hacen muy pocas asunciones sobre las capacidades tecnológicas de cada uno de los receptores. Sin embargo, los medios para exponer y consumir esos servicios puede darse de diferentes formas, las más comunes son las especificaciones SOAP y REST. Dado que el proyecto se realizó sobre REST, SOAP será tratado solo conceptualmente.

2.4.3 SOAP

Los servicios Web SOAP son aquellos que utilizan mensajes XML para intercomunicarse que siguen el estándar SOAP (Simple Object Access Protocol), un lenguaje XML que define la arquitectura y formato de los mensajes. Dichos sistemas normalmente contienen una descripción legible por la máquina de la descripción de las operaciones ofrecidas por el servicio, escrita en WSDL (Web Services Description Language), que es un lenguaje basado en XML para definir las interfaces sintácticamente. Chackray (2016) decía que SOAP es “Se trata de un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP basado en XML”.

Los web services de este tipo, funcionan con los siguientes componentes:

- **Simple Object Access Protocol:** SOAP es un protocolo escrito en XML para el intercambio de información entre aplicaciones. Es un formato para enviar mensajes, diseñado especialmente para servir de comunicación en Internet, pudiendo extender los HTTP headers.
- **Web Services Description Language:** WSDL es un lenguaje basado en XML para describir los servicios web y cómo acceder a ellos. Es el formato

estándar para describir un web service, y fue diseñado por Microsoft e IBM. WSDL es una parte integral del estándar UDDI, y es el lenguaje que éste utiliza.

- **Universal Description, Discovery and Integration:** UDDI es un estándar XML para describir, publicar y encontrar servicios web. Es un directorio donde las compañías pueden registrar y buscar servicios web. Es un directorio de interfaces de servicios web descritos en WSDL que se comunican mediante SOAP.

2.4.4 REST

Los servicios Web RESTful (Representational State Transfer Web Services) son adecuados para escenarios de integración básicos ad-hoc. Dichos servicios Web se suelen integrar mejor con HTTP que los servicios basado en SOAP, ya que no requieren mensajes XML o definiciones del servicio en forma de fichero WSDL.

REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST emergió en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.

De Seta (2008) afirma:

> REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. (párr. 1)

Principios REST

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

1. Utiliza los métodos HTTP de manera explícita.

2. No mantiene estado.
3. Expone URIs con forma de directorios.
4. Transfiere XML, JavaScript Object Notation (JSON), o ambos.

A continuación vamos a ver en detalle estos cuatro principios, y explicaremos porqué son importantes a la hora de diseñar un servicio web REST.

1. Utiliza los métodos HTTP de manera explícita

Una de las características claves de los servicios web REST es el uso explícito de los métodos HTTP, siguiendo el protocolo definido por RFC 2616. Por ejemplo, HTTP GET se define como un método productor de datos, cuyo uso está pensado para que las aplicaciones cliente obtengan recursos, busquen datos de un servidor web, o ejecuten una consulta esperando que el servidor web la realice y devuelva un conjunto de recursos.

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP, de acuerdo a esta asociación:

- Se usa POST para crear un recurso en el servidor.
- Se usa GET para obtener un recurso.
- Se usa PUT para cambiar el estado de un recurso o actualizarlo.
- Se usa DELETE para eliminar un recurso.

2. No mantiene estados

El segundo principio a abordar es la comunicación sin estado. En primer lugar, es importante señalar que, aunque REST incluya la idea de "no mantener", eso no significa que la aplicación que exponga sus funcionalidades no pueda tener estado, de hecho, esto haría que el enfoque sea inútil en la mayoría de los escenarios.

REST exige que el estado sea transformado en estado del recurso y sea mantenido en el cliente. En otras palabras, un servidor no debería guardar el estado de la comunicación de cualquiera de los clientes que se comunican con él más allá de una petición única. La razón más obvia de esto es la escalabilidad, el número de clientes que pueden interactuar con el servidor se vería significativamente afectada si fuera necesario mantener el estado del cliente.

Pero hay otros aspectos que podrían ser mucho más importantes: las restricciones de "no mantener" aislar al cliente de cambios en el servidor, por lo tanto, dos solicitudes consecutivas no dependen de la comunicación con el mismo servidor. Un cliente podría recibir un documento que contiene los enlaces al servidor y, a continuación, hacer algún procesamiento, y el servidor podría ser apagado, y el disco rígido podría ser sacado o sustituido, y el software podría ser actualizado y reiniciado.

3. **Expone la URI´s en forma de directorios**

Cuando se enfrentan con esta idea, muchas personas se preguntan si esto significa que deben exponer sus entradas de la base de datos (o sus números de identificación) directamente y, a menudo, se enojan por la simple idea, una vez que años de práctica de orientación a los objetos nos dicen que debemos ocultar los aspectos de las bases de datos. Por ejemplo, un recurso de Pedido podría estar compuesto de los temas de pedido, una dirección y muchos otros aspectos que usted por los que no sería conveniente exponer como un recurso identificable individualmente. Tomando la idea de identificar de todo lo que vale la pena ser identificado, lleva a la creación de recursos que usted no suele ver, en un típico diseño de la aplicación: un proceso o un paso de un proceso, una venta, una negociación, una solicitud de cotización. Esto, a su vez, puede conducir a la creación de entidades más persistentes que en un diseño no RESTful.

Algunos ejemplos de URI que podría tener:

- http://example.com/customers/1234
- http://example.com/orders/2007/10/776654
- http://example.com/products/4554

- http://example.com/processes/salary-increase-234

4. **Transferencia en JSON**

El formato de intercambio de datos oficial en servicios REST, es JSON (JavaScript Object Notation). JSON es un formato de texto ligero para el intercambio de datos.

Siendo REST una arquitectura diseñada para su uso sobre el protocolo http y que se utiliza en aplicaciones web, debemos considerar que por lo general, todas estas, utilizan JavaScript, ya sea nativo, mediante frameworks o abstracciones del mismo (como TypeScript). Existiendo una notación de objetos en JavaScript, era necesario considerar un formato propio, para evitar los problemas de conversión de datos entre etiquetas XML y objetos de JavaScript. De acuerdo a ello, surgió un formato de datos que fuese transparente al convertir datos, el cuál fue el mencionado JSON.

2.5 Cloud Computing

2.5.1 Introducción

Si bien es cierto, que no es un concepto muy novedoso, Cloud, es desde hace años, algo que está cambiando la informática y la forma en la que nos relacionamos con ella. Cloud Computing no es más que una versión más avanzada de los centros de Servicios de Proceso de Datos que teníamos hace 40 años. Finalmente los centros de datos eran servicios que proveían información a internet y esa información era la nube para cualquier dispositivo externo, pero en esencia eran servidores de hardware administrados por alguien más, que básicamente es la idea de cloud.

Amazon (2018) define:

> La cloud computing proporciona a los desarrolladores y departamentos de TI la capacidad de concentrarse en lo que más importa y evitar arduas tareas como el aprovisionamiento, el mantenimiento y la planificación de capacidad. (párr. 1)

El modelo cloud computing no sustituye a las arquitecturas anteriores, pero consigue cambiar radicalmente la forma en la que se utilizan, la diferencia más importante del concepto de cloud es que cualquiera puede administrarlos, definir memoria, almacenamiento, procesamiento y disponibilidad de la configuración de servidores virtuales, sin contar si quiera con los medios físicos, siendo estos, provistos por un tercero.

2.5.2 Concepto

Un servidor cloud se sustenta sobre una infraestructura especial que le permite abstraerse totalmente del hardware, donde no sólo la máquina o el servidor están virtualizados, sino que otros componentes como la red y el almacenamiento también lo están. Está abstracción respecto al hardware significa que un servidor cloud no está atado a un ordenador físico concreto, no está ubicado en un único ordenador.

Amazon (2018) sintetiza las características de la nube así:

> Hacer posibles las operaciones de vital importancia de su empresa, una plataforma de servicios en la nube proporciona acceso rápido a recursos de TI flexibles y de bajo costo. Gracias a la informática en la nube, no necesitará realizar grandes inversiones iniciales para la adquisición de equipos ni tendrá que dedicar mucho tiempo a la tediosa tarea de administrar dichos equipos. En lugar de todo eso, podrá aprovisionar el tipo y el tamaño exactos de los recursos informáticos que necesite para hacer realidad su nueva y genial idea, o para el funcionamiento de su departamento de TI. (párr. 1)

2.5.3 Características

De la computación en nube podemos resaltar siete características:

1. **Sincronización automática**

Cuando se apuesta por un servicio Cloud Computing, algo esencial es el que los archivos se sincronicen de manera automática ya que las actualizaciones en tiempo real resultan básicas para que los archivos reflejen los cambios que realizamos en ellos casi en el acto. La realidad es que no todos los servicios que ofrece la nube disponen de esta funcionalidad por lo que podríamos encontrarnos con un problema

importante si nos olvidamos de sincronizar los archivos manualmente de cara a un proyecto importante.

Los mejores servicios de almacenamiento en la nube, además de sincronización automática, nos permiten también a los usuarios programar copias de seguridad permanentes en una unidad externa especificando los intervalos de tiempo que deseemos para tal acción.

2. **Seguridad**

Una de las principales preocupaciones de una empresa, se basa en la seguridad proporcionada por un servicio Cloud Computing ya que los datos con los que trabaja son privados y no quieren que se vean comprometidos por un servicio que carezca de las características adecuadas.

Es conveniente, por tanto, elegir un servicio que cuente con encriptado de datos permitiéndonos elegir nuestra propia clave de cifrado; esto podemos complementarlo con diferentes niveles de acceso para cada uno de los empleados que formen parte de nuestra empresa pudiendo, según el trabajo de cada uno, acceder a un tipo de archivos determinados dejando el resto fuera de su alcance.

3. **Herramientas de libre elección y colaborativas**

Otra de las ventajas que definen al Cloud Computing es la libertad que nos ofrece en todo momento a los internautas para elegir qué aplicaciones queremos usar pudiendo optar entre aquellas que son gratuitas y las que son de pago. En el caso de estas últimas, su coste dependerá siempre de variables como el tipo de servicio contratado, el tiempo que vamos a usarlo, el volumen de tráfico de datos utilizado, etc., también influye si cuentan o no con diversos roles de usuario para los accesos: protección con contraseña, niveles de carpetas y subcarpetas, compartir a través de enlace directo o por correo electrónico.

Al hacer uso de herramientas colaborativas, podremos compartir archivos con aquellos usuarios autorizados en cualquier dispositivo, ordenador de sobremesa, tablet, smartphone, etc.

4. **Coste económico y escalabilidad**

Hablamos de escalabilidad como la capacidad de almacenamiento flexible a un precio asequible, es decir; según el servicio que necesitemos y las funcionalidades que tengamos, el coste variará de manera lógica comunicándonos nuestro proveedor la modificación en la tarifa de suscripción. No es lo mismo estar haciendo uso del 25% de una aplicación que el 80% ya que las necesidades de nuestro negocio pueden variar con el tiempo.

5. **Edición de archivos desde la nube**

Si bien es cierto que no todos los datos son evitables online (la música, por ejemplo, hay que descargarla en un disco duro) además de compartir archivos, poder verlos y acceder a ellos, un buen servicio Cloud Computing permite editarlos desde la propia plataforma haciendo que estos cambios se sincronicen automáticamente en todos los dispositivos a la vez que posean conexión a internet.

6. **Supervisión y mantenimiento del servicio**

En el caso de las aplicaciones de computación en la nube, es más sencillo, ya que no necesitan ser instalados en el ordenador de cada usuario y se puede acceder desde diferentes lugares. Controlar y optimizar el uso de los recursos de manera automática por lo que, el uso de los mismos, puede seguirse y notificarse aportando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

7. **Independencia entre el dispositivo y la ubicación**

Permite a los usuarios acceder a los sistemas utilizando un navegador web, independientemente de su ubicación o del dispositivo que utilice (por ejemplo, PC, teléfono móvil). Siempre aquél que mejor se adapte al crecimiento de nuestro negocio.

2.5.4 Tipos de servicios

La entrega de servicios de computación bajo demanda (desde aplicaciones a centros de datos) a través de internet en una base de pago por uso. Basándose en tres puntos esenciales:

1. **Recursos flexibles:** aumente o disminuya los recursos de forma rápida y sencilla para satisfacer la demanda.
2. **Servicios a la medida:** para que sólo pague lo que utilice.
3. **Autoservicio:** todos los recursos de TI que necesita con acceso de autoservicio.

Cada forma de proveer servicios traen consigo varias capas, dependiendo de estas, clasificamos el tipo en:

- Infraestructura como servicio (IaaS).
- Plataforma como servicio (PaaS).
- Software como servicio (SaaS).

Adicional, existe el modelo tradicional (Legacy), donde las organizaciones administraban el hardware físicamente, On-Premises.

La imagen que se muestra a continuación nos muestra de la diferencia entre la administración de servicios es los 4 modelos:

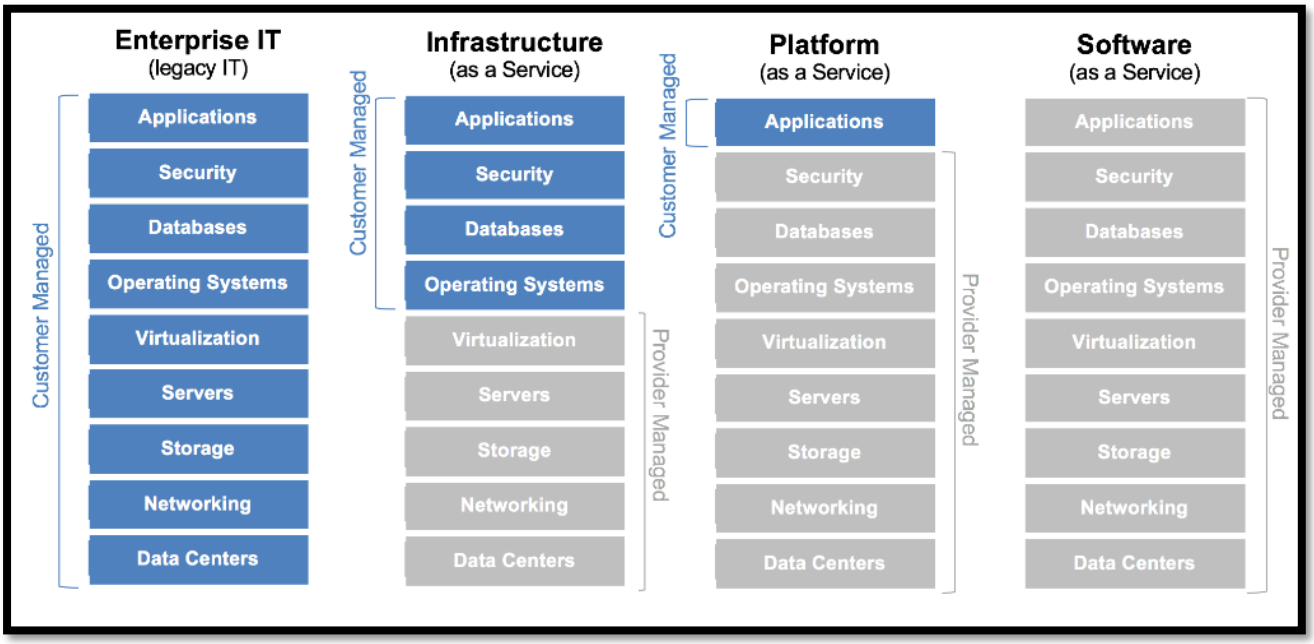


Figura 2.5.4.1: Representación Gráfica de Premises, IaaS, PaaS y SaaS.

Yolar (2018). Iaas Paas Saas [Gráfica]. Recuperado de http://yolar.cinetonic.co/iaas-paas-saas/.

On-Premises

Puede traducirse el término como “instalado” o “en las instalaciones”, es la arquitectura de hardware que se encuentra área geográfica de la empresa u organización (no remota), para atender los servicios informáticos de la misma.

Básicamente, la diferencia fundamental entre el software en la nube y el on-premise es donde reside. El software local se instala localmente en las computadoras y servidores de su empresa, donde el software en la nube se aloja en el servidor del proveedor y se accede a través de un navegador web.

Aunque en teoría, los servicios basados en hardware físico, que administra directamente la organización, son más controlables pero menos escalables en implementación y crecimiento.

- DevOps y no de un Administrador de Sistemas.

Debido a estos contras de hardware local, han surgido paradigmas que ofrecen estos productos como servicios, abstrayendo la implementación física de los mismos.

Infraestructura como servicio (IaaS)

La infraestructura como un servicio brinda a las empresas los recursos de la computación, incluso servidores, redes, almacenamiento y espacio para centro de datos con una base de pago según el uso.

Amazon (2018) define:

> La Infraestructura como servicio, que a veces se abrevia a IaaS, contiene los bloques de creación fundamentales para la TI en la nube. Por lo general, proporciona acceso a las características de redes, a los equipos (virtuales o en software dedicado) y al espacio de almacenamiento de datos. La Infraestructura como servicio le proporciona el mayor nivel de flexibilidad y control de la administración en torno a sus recursos de TI y guarda el mayor

> parecido con los recursos de TI existentes con los que muchos departamentos de TI y desarrolladores están familiarizados. (párr. 3)

En otras palabras, IaaS proporciona acceso a recursos informáticos situados en un entorno virtualizado, la"nube" (cloud), a través de una conexión pública, que suele ser internet. En el caso de IaaS, los recursos informáticos ofrecidos consisten, en particular, en hardware virtualizado, o, en otras palabras, infraestructura de procesamiento. IaaS abarca aspectos como el espacio en servidores virtuales, conexiones de red, ancho de banda, direcciones IP y balanceadores de carga. Físicamente, el repertorio de recursos de hardware disponibles procede de multitud de servidores y redes, generalmente distribuidos entre numerosos centros de datos, de cuyo mantenimiento se encarga el proveedor del servicio cloud. El cliente, por su parte, obtiene acceso a los componentes virtualizados para construir con ellos su propia plataforma informática.

Plataforma como servicio (PaaS)

Es una categoría de servicios cloud que proporciona una plataforma y un entorno que permiten a los desarrolladores crear aplicaciones y servicios que funcionen a través de internet. Los servicios PaaS se alojan en la nube, y los usuarios pueden acceder a ellos simplemente a través de su navegador web.

Amazon (2018) define:

> Las Plataformas como servicio eliminan la necesidad de las compañías de administrar la infraestructura subyacente (normalmente hardware y sistemas operativos) y le permiten centrarse en la implementación y la administración de sus aplicaciones. Esto contribuye a mejorar su eficacia, pues no tiene que preocuparse del aprovisionamiento de recursos, la planificación de la capacidad, el mantenimiento de software, los parches ni ninguna de las demás arduas tareas que conlleva la ejecución de su aplicación. (párr. 5)

El modelo PaaS permite a los usuarios crear aplicaciones de software utilizando herramientas suministradas por el proveedor. Los servicios PaaS pueden consistir en funcionalidades pre configuradas a las que los clientes puedan suscribirse, eligiendo las funciones que deseen incluir para resolver sus necesidades y

descartando aquellas que no necesiten. Así, los paquetes pueden variar desde un sencillo entorno que se maneje con el ratón y no requiera ningún tipo de conocimiento o instalación especial por el lado del usuario, hasta el suministro de opciones de infraestructura para desarrollo avanzado.

La infraestructura y las aplicaciones se gestionan en nombre del cliente, y se ofrece también soporte técnico. Los servicios se actualizan constantemente, mejorando las funcionalidades existentes y añadiendo otras nuevas. Los proveedores de PaaS pueden colaborar con los desarrolladores desde la concepción de sus ideas originales hasta la creación de las aplicaciones, llegando incluso hasta las fases de pruebas e implantación. Y todo eso se consigue utilizando un solo mecanismo gestionado.

Al igual que en la mayoría de las propuestas de servicios cloud, los servicios PaaS suelen facturarse como una suscripción en la que el cliente acaba pagando al final sólo por lo que realmente utiliza. Además, puede beneficiarse de las economías de escala que aporta el hecho de estar compartiendo una misma infraestructura física subyacente entre muchos usuarios, lo que se traduce en una reducción de costes.

Software como servicio (SaaS)

Las aplicaciones basadas en la nube, o software como servicio, se ejecutan en computadoras que están distantes “en la nube” que son poseídas y gestionadas por otros, y que se conectan a las computadoras de los usuarios a través de Internet y, normalmente, de un navegador web. Estas aplicaciones, tienen como características:

- Puede registrarse y empezar a utilizar rápidamente las innovadoras aplicaciones de negocio.
- Las aplicaciones y los datos se encuentran accesibles desde cualquier computadora conectada.
- Ningún dato se pierde si su computadora deja de funcionar, ya que los datos están en la nube.
- Este servicio es capaz de escalar dinámicamente en función de las necesidades de uso.

Amazon (2018) define:

> El Software como servicio le proporciona un producto completo que el proveedor del servicio ejecuta y administra. En la mayoría de los casos, quienes hablan de Software como servicio en realidad se refieren a aplicaciones de usuario final. Con una oferta de SaaS, no tiene que pensar en cómo se mantiene el servicio ni en cómo se administra la infraestructura subyacente. Solo tiene que preocuparse de cómo utilizar el software concreto. (párr. 7)

2.5.5 Tipos de nubes

Existen diferentes implementaciones de la nube como servicio, estos tipos son:

- Nube pública.
- Nube privada.
- Nube híbrida.

Nube pública

Las nubes públicas perteneces y son administradas por empresas que ofrecen acceso rápido a recursos de computación accesibles a través de una red pública. Con los servicios de nube pública, los usuarios no necesitan adquirir hardware, software o infraestructura de soporte, ya que pertenecen y se gestionan por proveedores.

Microsoft (2018) lo define como:

> Las nubes públicas son la forma más común de implementar la informática en la nube. Los recursos de la nube (como servidores y almacenamiento) son propiedad de otro proveedor de servicios en la nube, que los administra y ofrece a través de Internet. (párr. 1)

Características

- Las innovadoras aplicaciones de negocio de SaaS para aplicaciones que abarcan desde la gestión de recursos del cliente (CRM) hasta la gestión de transacciones y analítica de datos.
- IaaS flexible y escalable para servicios de almacenamiento y computación en cualquier momento.
- PaaS potente para entornos de despliegue y desarrollo de aplicaciones basadas en la nube.

Nube privada

Una nube privada es una infraestructura que se opera exclusivamente por una única organización, ya sea gestionada internamente o por un tercero, y es alojada internamente o externamente. Las nubes privadas pueden aprovechar las eficiencias de la nube, a la vez que ofrecen un mayor control de los recursos y evitan la multitenencia.

Microsoft (2018) lo define como:

Una nube privada está compuesta por recursos informáticos que utilizan exclusivamente una empresa u organización. La nube privada puede ubicarse físicamente en el centro de datos local de su organización u hospedarla un proveedor de servicios externo. (párr. 1)

Características

- Funciona con una interfaz que controla los servicios, lo que permite al personal de TI suministrar, asignar y distribuir los recursos de TI bajo demanda.
- Gestión altamente automatizada de las agrupaciones de recursos para todo, desde la capacidad de cómputo al almacenamiento, analítica y middleware.
- Seguridad y control sofisticados proyectados para los requisitos específicos de la empresa.

Nube híbrida

Una nube híbrida utiliza una base de nube privada, combinada con la integración estratégica y el uso de servicios de nube pública. En realidad, una nube privada no puede existir en asilado del resto de los recursos de TI de una empresa ni de la nube pública. La mayoría de las empresas con nubes privadas evolucionarán para gestionar cargas de trabajo en todos los centros de datos, nubes privadas y nubes públicas, de esta manera crearán nubes híbridas.

Microsoft (2018) define:

> Las nubes híbridas, que suelen llamarse "lo mejor de ambos mundos", combinan infraestructura local (o nubes privadas) con nubes públicas, de modo que las organizaciones puedan beneficiarse de las ventajas de ambas. En una nube híbrida, los datos y las aplicaciones pueden moverse entre nubes privadas y públicas para obtener más flexibilidad y opciones de implementación. (párr. 5)

Características

- Permite que todas las empresas mantengan las aplicaciones críticas y los datos confidenciales en un entorno de centro de datos tradicional o en una nube privada.
- Permite beneficiarse de los recursos de la nube pública como SaaS, para obtener las aplicaciones más recientes y la infraestructura IaaS para obtener recursos virtuales de forma flexible.
- Facilita la portabilidad de datos, aplicaciones y servicios, y más opciones de modelos de despliegue.

2.5.6 Ventajas

Mientras se han desarrollado los conceptos de cloud en los puntos del capítulo, se han podido observar, las distintas ventajas que existen de utilizar cloud, a continuación se resumen las siete ventajas más destacables:

1. **Reducción de costes**

La empresa prescinde de inversiones en infraestructura TI propia y licencias de software. Para empezar a trabajar no es necesario instalar ningún tipo especial de hardware. Por el contrario, adquirir una infraestructura tecnológica propia es mucho más costoso y no todos los negocios pueden permitírselo. ¿Por qué? porque no se trata solamente de la adquisición tecnológica, sino de todo lo que esto conlleva: gastos de mantenimiento, energéticos, personal técnico etc.

Está demostrado que con la gestión desde la nube un negocio puede ahorrar entre un 20% y 30%. Por otro lado, la empresa se puede centrar exclusivamente en su actividad sin necesidad de tener que dedicar tiempo y recursos a mantenimiento tecnológico, que a veces puede salir tremendamente caro.

2. **Movilidad: acceso desde cualquier dispositivo y lugar**

Desde cualquier punto se tiene acceso a toda la información de la empresa. La movilidad se ha convertido en una gran ventaja competitiva, tanto para trabajar o atender clientes desde cualquier lugar, como para favorecer la flexibilidad laboral de un trabajador que puede organizar toda su actividad sin depender de una oficina. Simplemente basta con tener conexión a Internet para acceder a las aplicaciones o a la información. Además, los documentos están alojados en la nube, no en equipos individuales, por lo tanto distintos usuarios pueden compartirlos o trabajar sobre ellos sin necesidad de estar físicamente juntos. Según distintos estudios, las empresas que aprovechan la movilidad y el trabajo colaborativo, aumentan sus ingresos en un 50% de media.

3. **Pago por uso y gasto bajo control**

El cloud computing se basa en modelos de pago por uso. La empresa contrata únicamente los servicios que necesita en cada momento y tienen la posibilidad de ajustar los gastos a sus necesidades reales. El cliente añade o elimina servicios en función de lo que requiera, evitando tener que invertir en infraestructura propia que con el tiempo quedaría obsoleta, es decir; paga por lo que necesita y cuando lo necesita.

4. **Tecnología siempre actualizada**

Poder disfrutar siempre de las últimas versiones del software y las más modernas aplicaciones hasta hace poco tiempo era un “lujo” solo al alcance de las grandes compañías. Con la nube, el cliente se asegura una tecnología siempre actualizada y optimizada.

5. **Capacidad de almacenamiento ilimitada**

Hoy en día manejamos grandes cantidades de datos y la nube ofrece un almacenamiento prácticamente ilimitado. Por ejemplo, si tu ordenador tiene 500 gigabytes de disco duro, ten en cuenta que eso es infinitamente pequeño comparado con los terabytes disponibles en la nube, es decir; el usuario no está obligado a realizar ampliaciones en sus propios equipos cada poco tiempo, con todo el gasto que esto conlleva.

6. **Apoyo al medio ambiente**

Aunque en cloud computing casi siempre se habla en términos de rentabilidad o productividad, este punto es también muy importante. Hacer uso de la nube, reduce la huella de carbono de una empresa al ahorrar recursos que pasan de estar almacenados en componentes físicos a ser virtuales. Esto supone un considerable ahorro en consumo de energía, lo que se traduce en importantes beneficios para el medio ambiente. La virtualización puede llegar a reducir el consumo de energía y sus niveles de contaminación en más de un 60%.

7. **Seguridad**

A diferencia de la informática tradicional, en la que cualquier fallo en el PC puede destruir todos los datos, en este entorno, si el disco del ordenador deja de funcionar eso no afectaría a tus datos. Y poniéndonos algo menos “drásticos”, si simplemente se da el caso en el que el ordenador se bloquea, todos los datos seguirán en la nube y totalmente accesibles desde otro dispositivo. En cuanto a las copias de seguridad de la información, está claro que son vitales para cualquier negocio, pero ¿cuál es la realidad? que no todo el mundo las hace, o al menos no de forma regular. En el entorno cloud computing el usuario se despreocupa de las copias de seguridad porque la nube las realiza automáticamente y con un cifrado seguro, a prueba de cualquier ataque de hackers.

CAPÍTULO 3 DESARROLLO DE LA PLATAFORMA WEB

3.1 Resumen del proceso de desarrollo

El presente esquema funge como resumen del proceso de desarrollo de la plataforma, el cuál es explicado posteriormente dentro de cada tema.

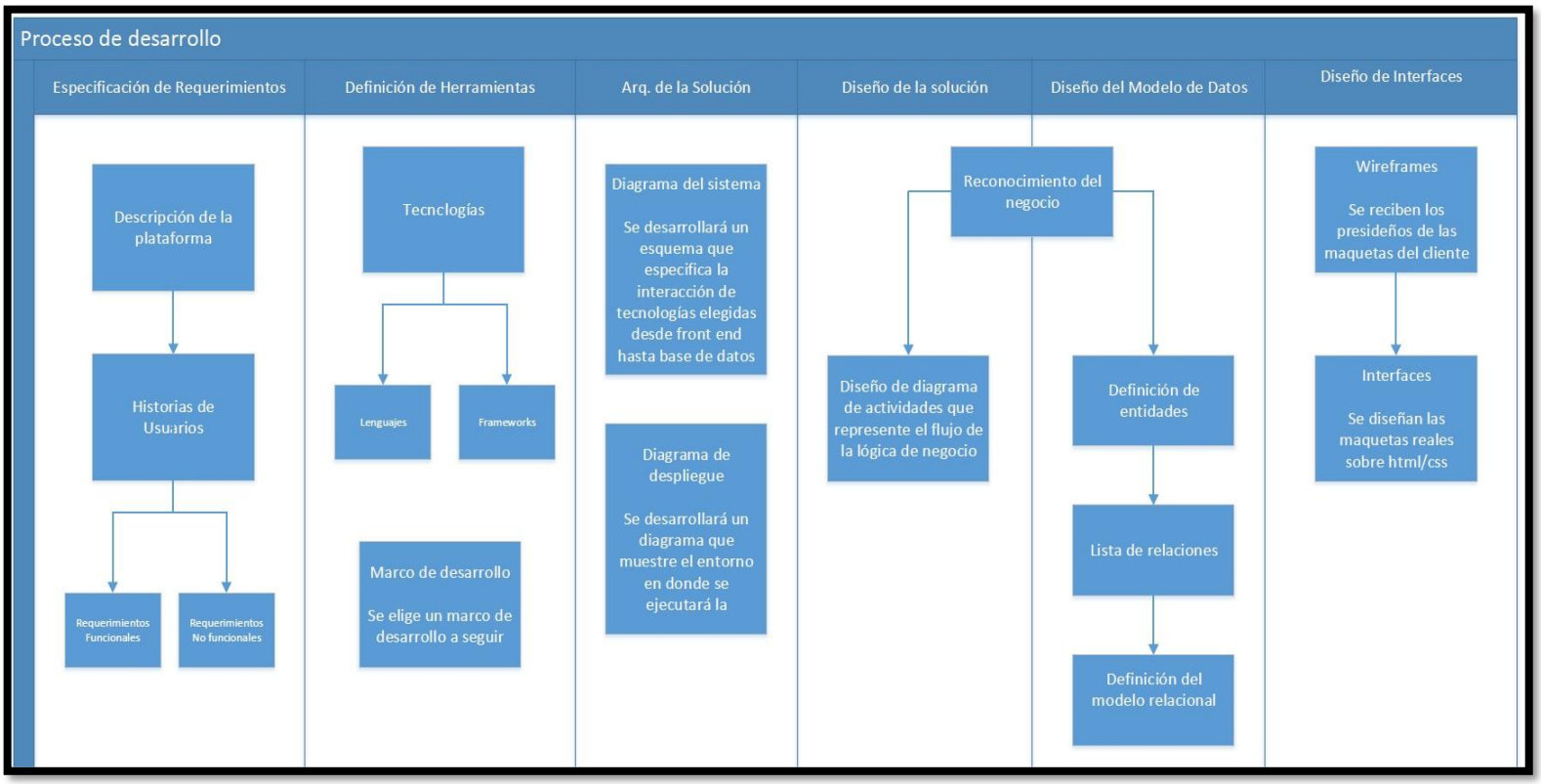


Figura 3.1.1: Proceso de desarrollo.

[Diagrama]. Diseño propio.

3.2 Especificación de requerimientos

En los próximos puntos, se desglosarán los aspectos que competen al análisis de la plataforma, entendiendo por separado la necesidad del cliente, las funcionalidades que se planean y los requerimientos tanto de operación como de demanda, así mismo se definirá que es lo que se entrega como producto final.

3.2.1 Descripción general de la plataforma

Implant presupuestos debe ser un sistema web en el que se pueda controlar de forma integral, segura y dinámica los procesos de contratación, asignación y seguimiento de recursos (empleados), presupuestos y proyectos del área de

tecnología de la empresa BBVA Bancomer. Adicionalmente, debe funcionar en un entorno cloud de Amazon Web Services.

3.2.2 Historias de usuario

Finalmente presentamos una lista de las funcionalidades y que es lo que deben cumplir, para considerarlas como realizadas.

<table>
<tr><th colspan="6">Historia de usuario</th></tr>
<tr><th>ID</th><th>Título</th><th>Descripción</th><th>Definitions of Ready</th><th>Definitions of Done</th><th>Valor</th></tr>
<tr><td>1</td><td>Login</td><td>Login de usuario mediante correos de google que estén previamente registrados.</td><td>• Tener las estructuras de los proyectos con sus respectivos repositorios.
• Creación de la BD.
• Definición de roles.</td><td>• Autenticación mediante google. Validar solo correos registrados en BD.</td><td>5</td></tr>
<tr><td>2</td><td>Pantalla Home</td><td>Pantalla de bienvenida amigable con una de noticias de la API de BBVA.</td><td>• Tener las estructuras de los proyectos con sus respectivos repositorios.</td><td>• Mostrar noticias paginadas.</td><td>1</td></tr>
<tr><td colspan="6">Catálogos</td></tr>
<tr><td>3</td><td>Proyectos</td><td rowspan="5">Catálogos para cada una de las entidades más importantes de la aplicación, con opciones de búsqueda.</td><td rowspan="5">• Contar con los documentos para pruebas iniciales e ingesta a BD.
• Finalización de tareas de 8-13.</td><td rowspan="5">• Muestran datos los datos más relevantes de todos los registros.
• Están paginados.
• Existen filtros por columnas.
• Hay una búsqueda por texto.</td><td rowspan="5">4</td></tr>
<tr><td>4</td><td>Recursos</td></tr>
<tr><td>5</td><td>Proveedores</td></tr>
<tr><td>6</td><td>Pep’s</td></tr>
<tr><td>7</td><td>ITZ´s</td></tr>
<tr><td colspan="6">Registro de proyectos</td></tr>
<tr><td>8</td><td>General</td><td rowspan="5">Realizar el alta de ello, de acuerdo al formulario correspondiente.</td><td rowspan="5">• Modelo de datos.</td><td rowspan="5">• Se registra correctamente los datos en la tabla correspondiente.
• Hace validaciones de los campos.</td><td rowspan="5">5</td></tr>
<tr><td>9</td><td>Cerrado</td></tr>
<tr><td>10</td><td>Tarifario</td></tr>
<tr><td>11</td><td>Servicios Extendidos</td></tr>
<tr><td>12</td><td>Licencias</td></tr>
</table>

13	Servicios			• Valida lógica de negocio. • Los datos son recuperables. • Cada registro tiene un id alfanumérico (ITZ).	
Actualización de proyectos					
14	Cerrado	Actualización y llenado de datos adicionales para seguimiento y cambio de fase de los proyectos.	• Registro del proyecto respectivo	• Actualiza los datos del proyecto de manera correcta. • Permiten el cambio de status del proyecto. • No permite repetición de datos. • Hace validaciones de los campos. • Valida lógica de negocio. • Generación de finalización. • Asigna un código 17, XXX a proyectos terminados.	**5**
15	Tarifario				
16	Servicios Extendidos				
17	Licencias				
18	Servicios Extendidos				
19	Ingesta de información	Se puede llenar la BD con datos de archivos Excel mediante el uso de lambdas.	• Se cuenta con los documentos pertinentes (estandarizados).	• Se realiza una validación de carga. • Se devuelve una respuesta del status. • Es posible cargar archivos que correspondan.	**5**

				• Se mantiene un registro de la carga.	
20	Seguimiento de carga de archivos	Existe una tabla de consulta para verificar que archivos se cargan.	• La tarea 20 esta completada.	• Se puede consultar el nombre del documento, quien lo subió, fecha y hora, y cuál fue el status de la carga.	**5**
21	Registro de usuarios	Es posible agregar nuevos usuarios y controlar su nivel de acceso.	• Los roles se han definido correctamente.	• Un nuevo se registra. • Solo se permite el uso de dominios restringidos. • Existen validaciones de datos.	**4**
22	Actualización de usuarios	Se pueden cambiar datos de los usuarios así como darlos de baja.	• Tener concluida la tarea número 21.	• Se puede actualizar cualquier dato del usuario. • Puede darse de baja un usuario. • Solo se permite el uso de dominios restringidos. • Se validan los campos.	**4**
23	Actualización de moneda	El valor de conversiones de estos catálogos debe ser actualizable.	• Haber concluido las tareas 1 a 23.	• Se puede actualizar el valor de la moneda y el match de perfiles. • Existen validaciones de datos.	**4**
24	Actualización de perfiles				

25	Página de contacto	Se define una vista con contactos de jefes de área y soporte técnico.	• El sistema ya se encuentra en producción, o al menos, en pruebas finales.	• Se muestra el contacto de líder Implant, jefe de departamento, desarrollador y DevOps.	**3**

Tabla 3.2.1: Historias de usuario.

3.2.3 Requerimientos funcionales

Dadas las funcionalidades generales, es preciso comenzar a desglosar cada una de ellas para identificar que deben satisfacer para considerarse realizadas. Priorizarlas, mediante un modelo SCRUM permite documentar el trabajo realizado y mantener el orden y las iteraciones que propone la metodología, en cada Sprint.

Los requerimientos funcionales se presentan en la siguiente tabla surgida de la generalización de las historias de usuario:

Número	Nombre
1	Login.
2	Consulta de catálogos de entidades.
3	Registros de proyectos.
4	Seguimiento de proyectos.
5	Ingesta y actualización de información por medio de cargas XLS.
6	Consulta de archivos cargados.
7	Registro de usuarios.
8	Actualización de usuarios.
9	Actualización de moneda.
10	Actualización de perfiles.
11	Página de contacto.
12	Pantalla de bienvenida.

Figura 3.2.3.1: Requerimientos funcionales.

3.2.4 Requerimientos no funcionales

Los requerimientos no funcionales incluyen historias de usuarios que no afectan la funcionalidad core de la herramienta, además de ellas, se agregan los requerimientos del entorno productivo en base a la demanda y uso del sistema.

Número	Nombre	Descripción
1	Serverless	Estar montado en cloud.
2	Carga mínima y máxima de usuarios.	Mínimo: 10 usuarios.
3		Máximo: 70 usuarios.
4	Tiempos de respuesta.	Latencia de 1.2 segundos o menos.
5	Seguridad de acceso a servicios.	Los servicios solo pueden ser consumidos por la plataforma.
6	Protección de acceso.	Para la plataforma por zona geográfica.
7	Lazy Loading.	Para consumo de grandes cantidades de datos y no afectar la experiencia de usuario.
8	Service Worker.	Para velocidad de acceso y recarga de páginas.
9	Migración de datos.	Generar un script para migrar datos en AWS**.**
10	Permiso de acceso a datos.	Configuración del servidor de desarrollo y producción de PostgresSQL.

Figura 3.2.4.1: Requerimientos no funcionales.

3.3 Definición de Herramientas

Primeramente comenzó por definir las herramientas con las que se trabajaría, de acuerdo a lo solicitado por el cliente. Esto se dividió en dos partes:

- Selección de herramientas técnicas.
- Metodología de desarrollo.

3.3.1 Definición de tecnologías

3.3.1.1 Lenguajes de programación

- **HTML/CSS3:** utilizados para maquetar la aplicación. Cabe mencionar que de manera personal los estilos fueron bastante regulares ya que la idea era que fuera un sistema formal y sencillo visualmente.

- **TypeScript:** súper conjunto de JavaScript diseñado por Microsoft en 2012. Dado que el framework en cargado de la interactividad de la aplicación fue Angular 5, las implementaciones de dinámica nativa fueron principalmente en TypeScript.
- **JavaScript:** lenguaje interpretado que apareció en 1995 y que es y ha sido el más popular y fuerte dentro del desarrollo web. Dependiendo de la necesidad se usó mínimamente JavaScript en el proyecto, tratando de ajustarse al estándar de TypeScript. Cabe mencionar que su sintaxis es relativamente transparente.
- **Java:** en su versión 8 el lenguaje de programación más popular en entornos empresariales fungió en el back end como manejador del modelo y la lógica de negocio.
- **Postgres SQL:** SGBD relacional con una longevidad de más de 20 años, siendo la opción libre de SQL más potente en la actualidad. La Base de Datos fue administrada con esta herramienta por las razones comentadas.

3.3.1.2 Frameworks

- **MaterializeCSS:** framework de CSS basado en Material Design creado por un equipo de estudiante de la universidad de Carnegie Mellon en 2014 y que desde sus inicios se colocó a la par de Bootstrap o Foundation entre los Frameworks CSS con mayor aceptación. Este framework permitió el uso de componentes base y administración del grid, haciendo más ágil el desarrollo.
- **Angular:** framework de google creado en 2016, con su nueva versión, dejando atrás AngularJS. Es una herramienta robusta y que provee de bastantes funcionalidades que permiten manejar la lógica de procesos en front end.
- **Java Spark:** es un es un conjunto de librerías para la construcción de servicios web REST. Su implementación y gestión de servicios es bastante intuitiva y útil para agilizar desarrollos a gran escala.
- **Sql2o:** es un framework Java que ayuda a simplificar el acceso a datos, esta librería utiliza internamente la API JDBC para lanzar las consultas contra la base de datos, la misma agiliza el desarrollo al permitir operaciones como

manejo de excepciones, mapeo de datos, etc. Además de ser muy útil y usar poco código permite centrarse en la funcionalidad.

3.3.2 Marco de desarrollo

Como parte del proceso de desarrollo se trabajó mediante el marco SCRUM, a continuación, se presenta un esquema de los conceptos de SCRUM y posteriormente, se da una breve descripción de cómo se aplicó en el proyecto.

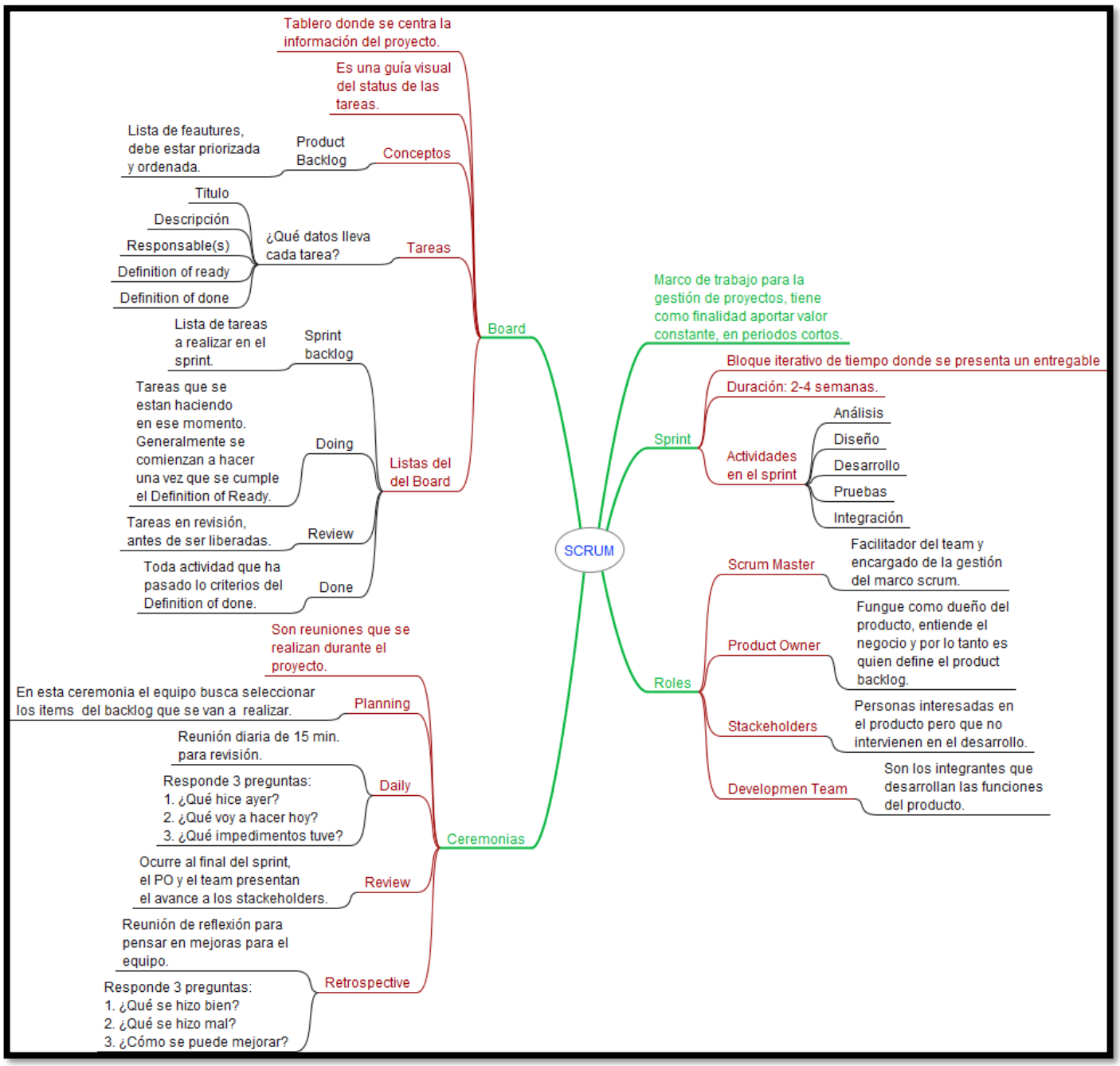


Figura 3.3.2: Marco SCRUM.

Montoya, J. (2018). [Mapa mental]. Diseño propio.

Aplicación de SCRUM

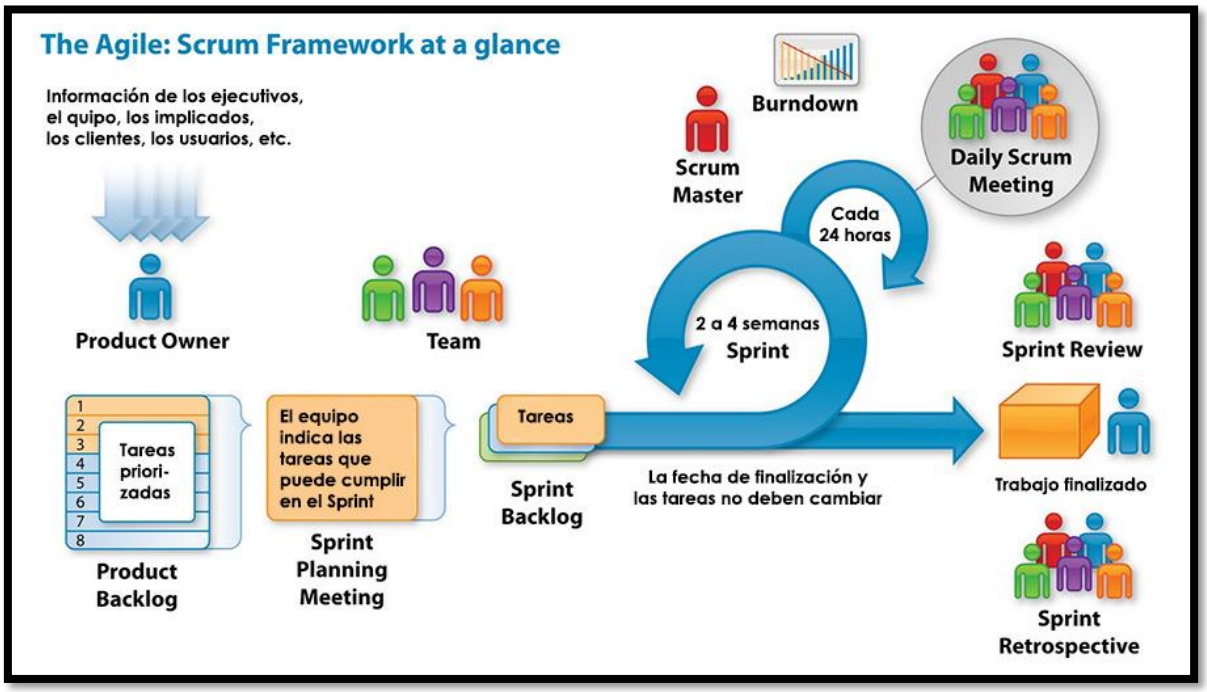


Figura 3.3.2: Flujo Scrum.

Torras, I. (2015). [Figura]. Recuperado de: https://metodologiascrum.readthedocs.io/en/latest/Scrum.html.

- Existen los cuatro roles de SCRUM en el proyecto, además de un Product Manager que evalúa resultados contractuales y los usuarios que son quienes probarán la plataforma.
- Los Sprint tienen una duración de 2 a 3 semanas según la complejidad de las tareas y se definirá en el Sprint Planning,
- El Scrum Board se gestionará mediante la aplicación Trello, considerando una lista para backlog, doing, review y do.
- La lista de tareas del backlog debe estar correctamente priorizada.
- Antes de iniciar una tarea deben estar satisfechas sus dependencias (Definition of Ready).
- Toda tarea en doing, debe tener un miembro responsable de ella.
- Preferentemente, no debería haber más de dos tareas en doing, para el mismo miembro del equipo.

- Cada tarea debe contar con una descripción y con su respectivo Definition of Done y of Ready.
- Toda tarea finalizada debe pasar por review y no pasará a done hasta que cumpla el Definition of Done.
- Si una tarea nueva surge debe considerar la prioridad con la que se agrega al backlog desde un inicio.
- Deben realizar dailys claras y concisas en beneficio de cosas que detengan el desarrollo y mantener informado al equipo.
- Las review y juntas de presentación deben acordarse con anticipación para evitar informalidades o problemas de entrega.

Como puede verse, el proyecto siguió los principios de la metodología SCRUM, aprovechando sus flexibilidades pero a su vez, respetando la guía de aplicación.

3.4 Arquitectura de la solución

La arquitectura del sistema se divide en dos rubros:

- Diagrama de componentes: corresponde a la infraestructura del sistema.
- Diagrama de despliegue: esquema arquitectónico sobre el que la plataforma está funcionando.

3.4.1 Diagrama de componentes

La aplicación maneja los componentes del modelo MVC bajo las tecnologías mencionadas en apartados anteriores, estas se exponen en la siguiente imagen:

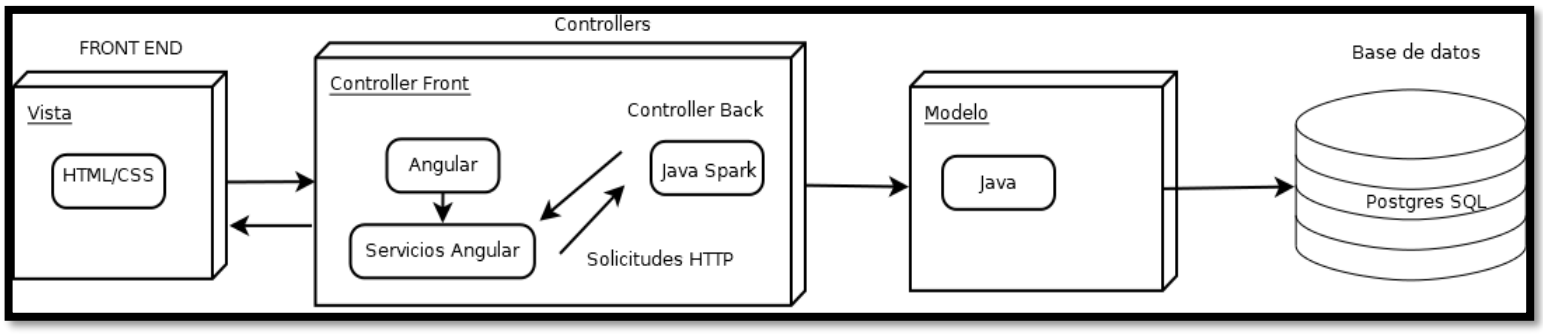


Figura 3.4.1.1: Diagrama de componentes del sistema.

Montoya, J (2018). [Diagrama]. Diseño propio.

3.4.2 Diagrama de despliegue

Recordando que la aplicación está sobre la nube, es necesario mostrar cómo está su distribución de componentes, donde se explica que servicios de AWS está utilizando y con qué fin. La imagen 13 nos da apoyo con su representación de componentes:

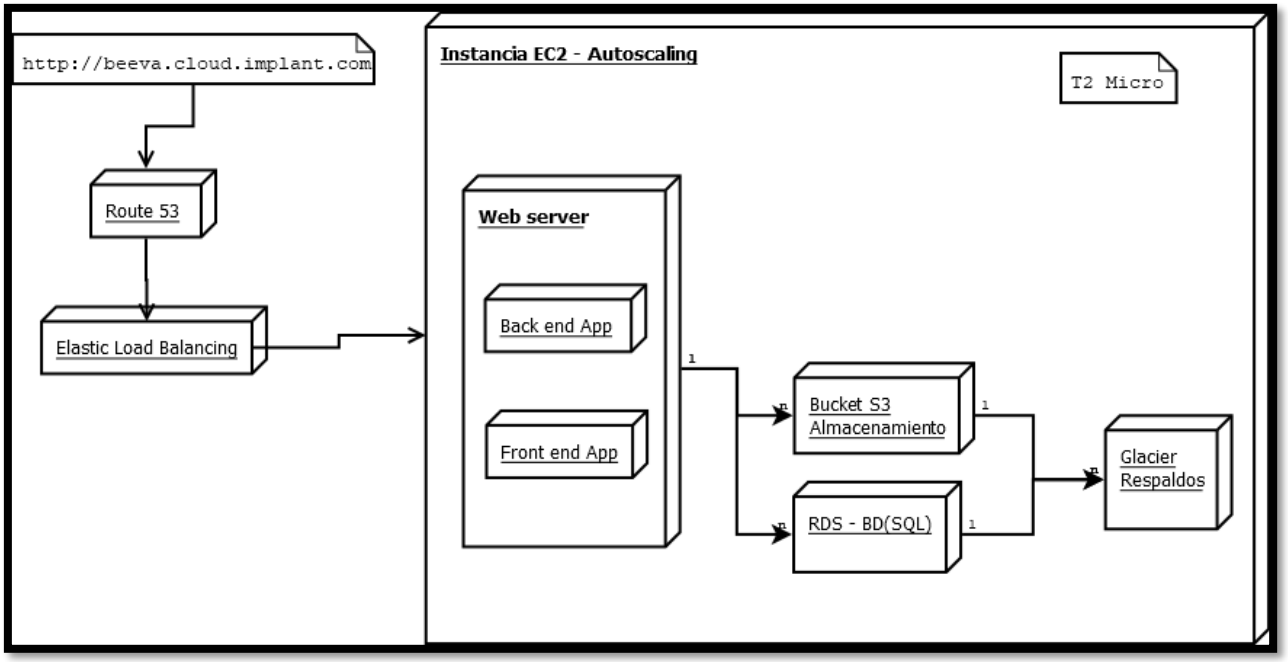


Figura 3.4.2.1: Diagrama de componentes de despliegue.

Montoya, J (2018). [Diagrama]. Diseño propio.

3.5 Diseño de la solución

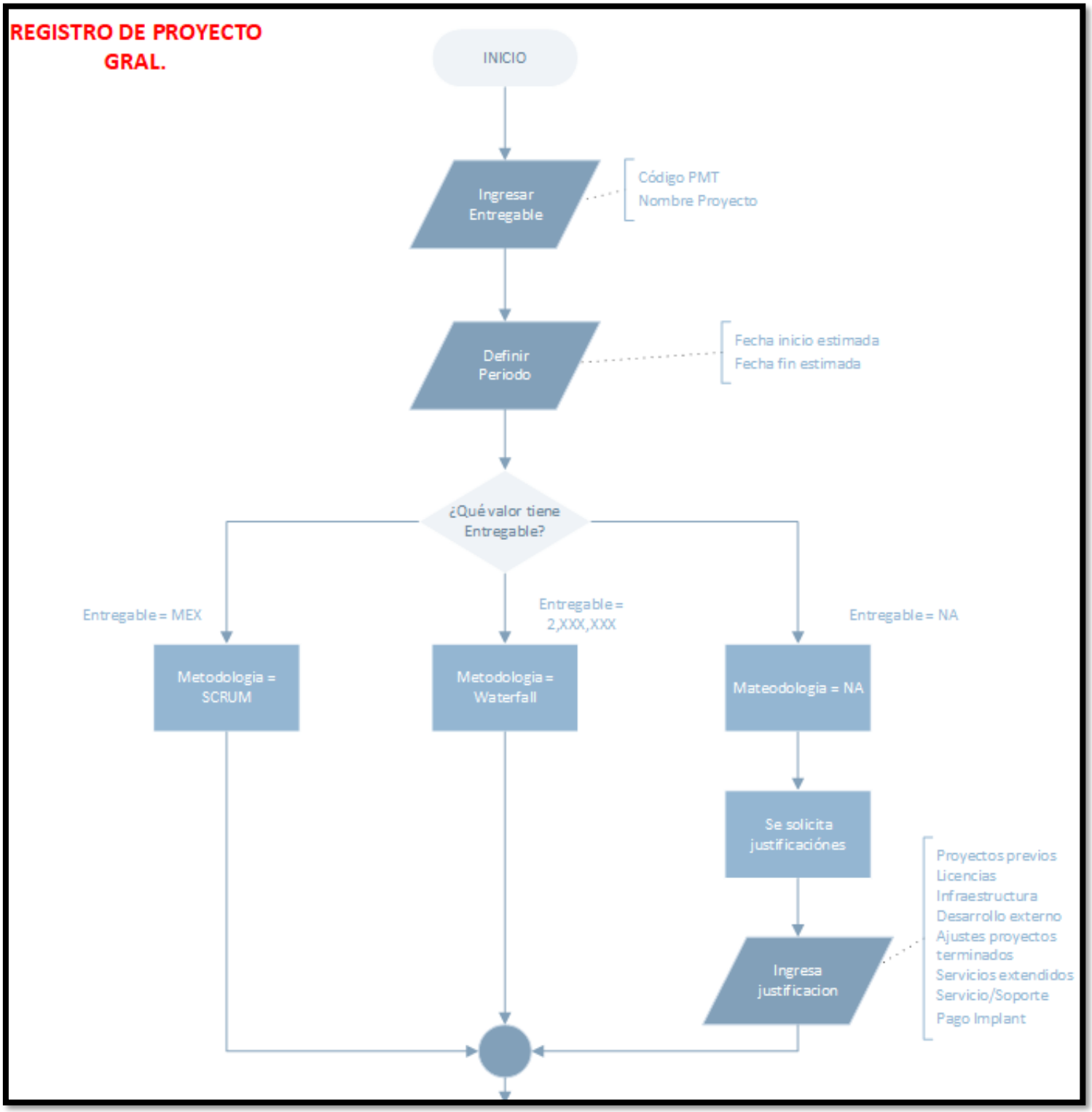
Se plantearon dos tipos de diagramas que tiene como objetivo comprender el funcionamiento esperado de la aplicación y representar las historias de usuario, tratadas con anterioridad. Estos objetivos se cumplirán mediante diagramas de actividades y de casos de uso.

3.5.1 Diagramas de actividades

Los diagramas que se presentan a continuación van a identificar el flujo que se sigue dentro del sistema desde que se ingresa a la URL hasta que se cierra sesión. Se presentan los diagramas de secuencia que implican la lógica de negocio, únicamente de los componentes core, registro general, registro de proyecto cerrado y tarifario y roles.

Registro general

El flujo del sistema en registro, así como sus reglas de negocio y validaciones puede observarse en la figura 3.5.1.1.



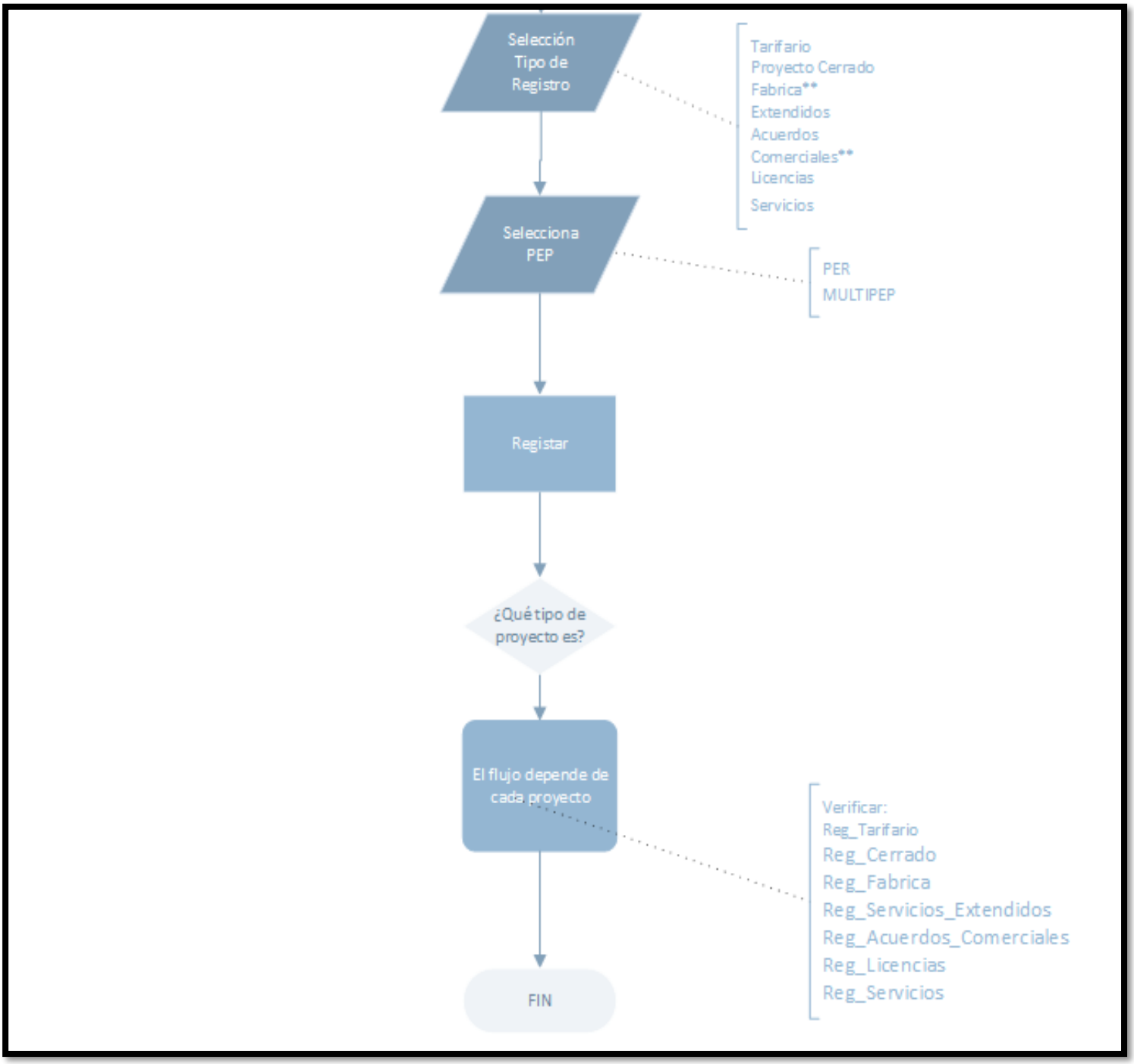
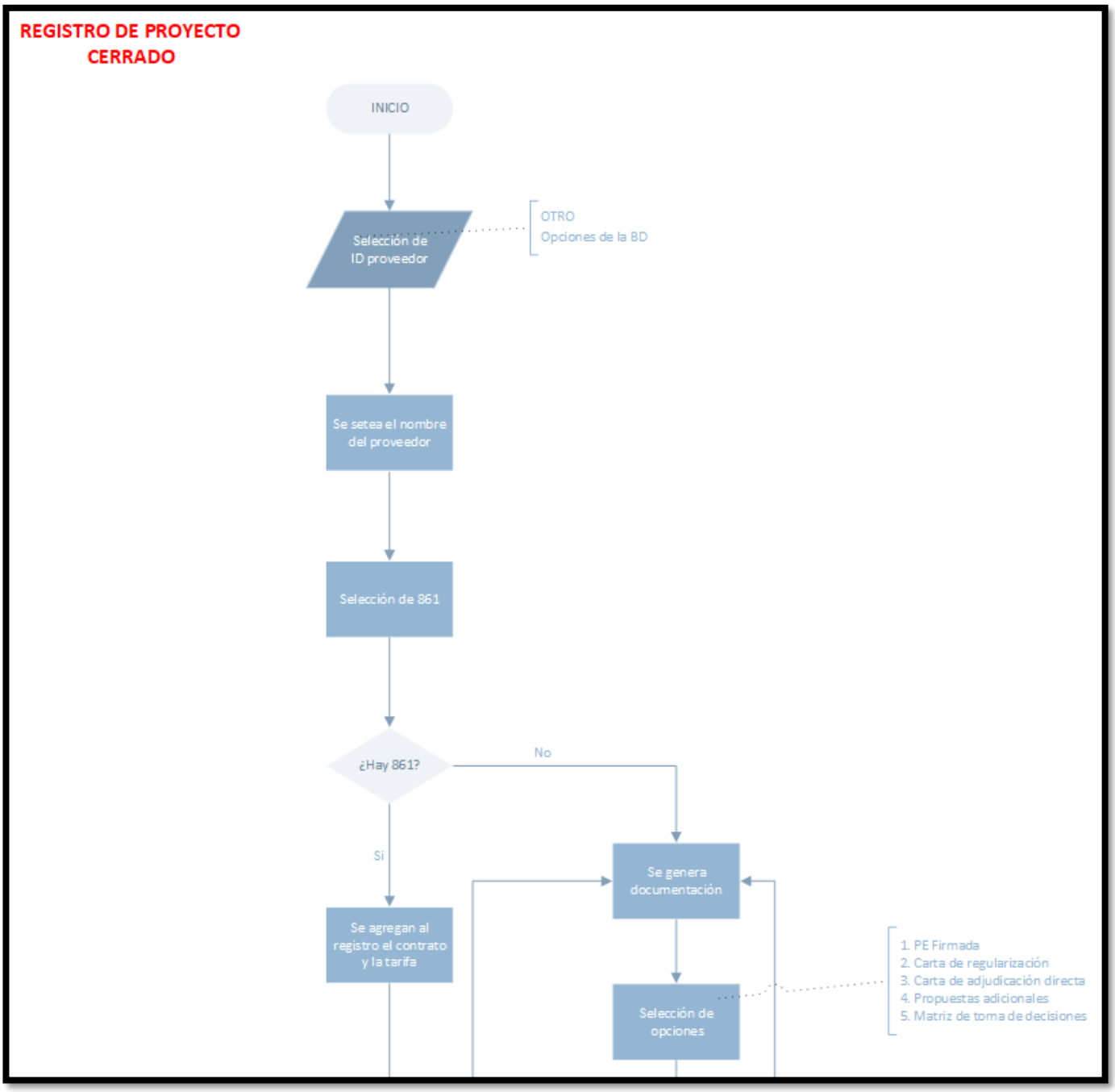


Figura 3.5.1.1: Diagrama de actividades del registro general del proyecto.

Montoya, J (2018). [Diagrama]. Diseño propio.

Proyecto cerrado

El flujo del sistema en registro de proyecto cerrado, así como sus reglas de negocio y validaciones puede observarse en la siguiente figura 3.5.1.2.



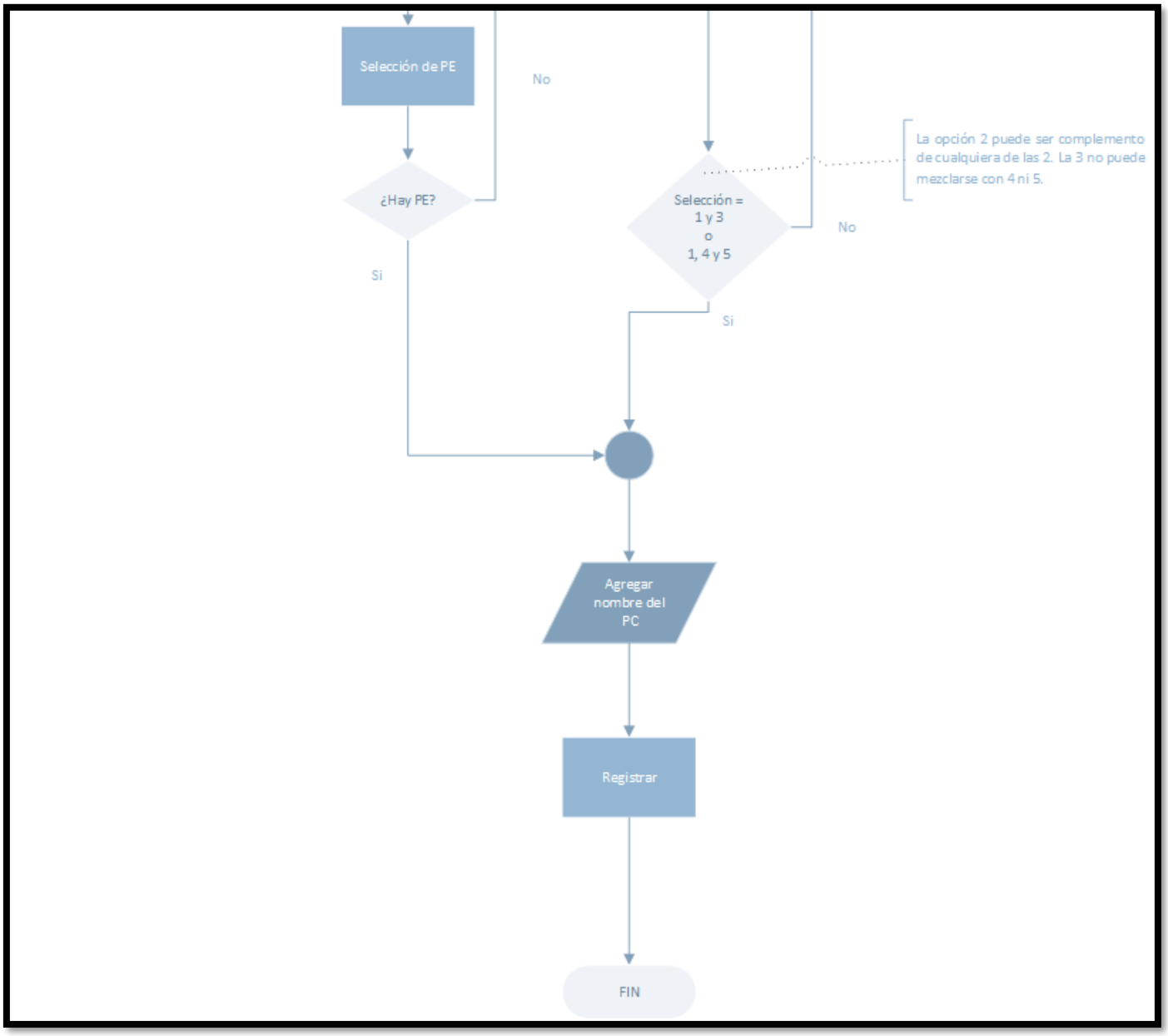
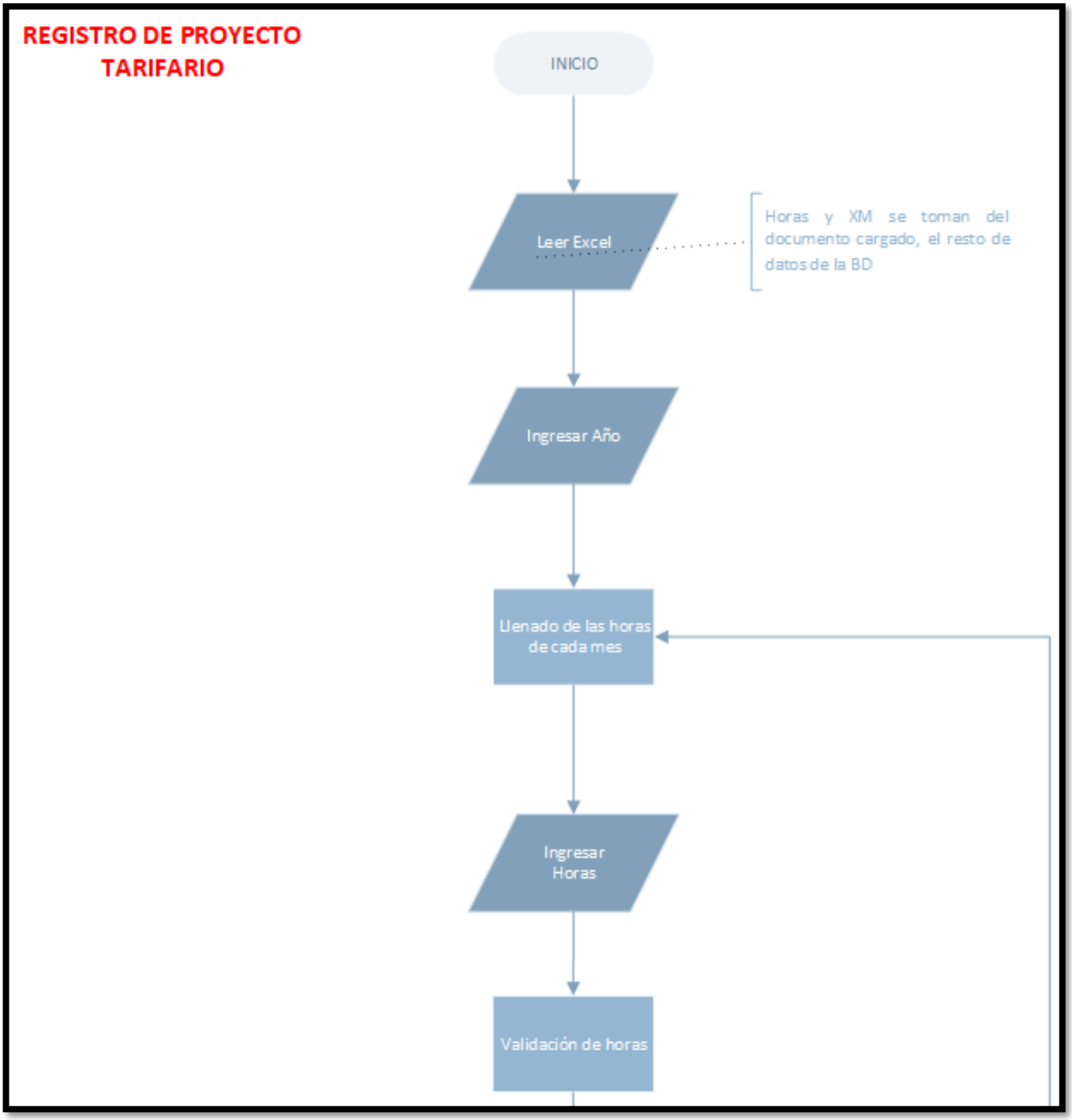


Figura 3.5.1.2: Diagrama de actividades del registro de proyecto cerrado.

Montoya, J (2018). [Diagrama]. Diseño propio.

Proyecto tarifario

El flujo del sistema en registro de proyecto cerrado, así como sus reglas de negocio y validaciones puede observarse en la siguiente figura 3.5.1.3.



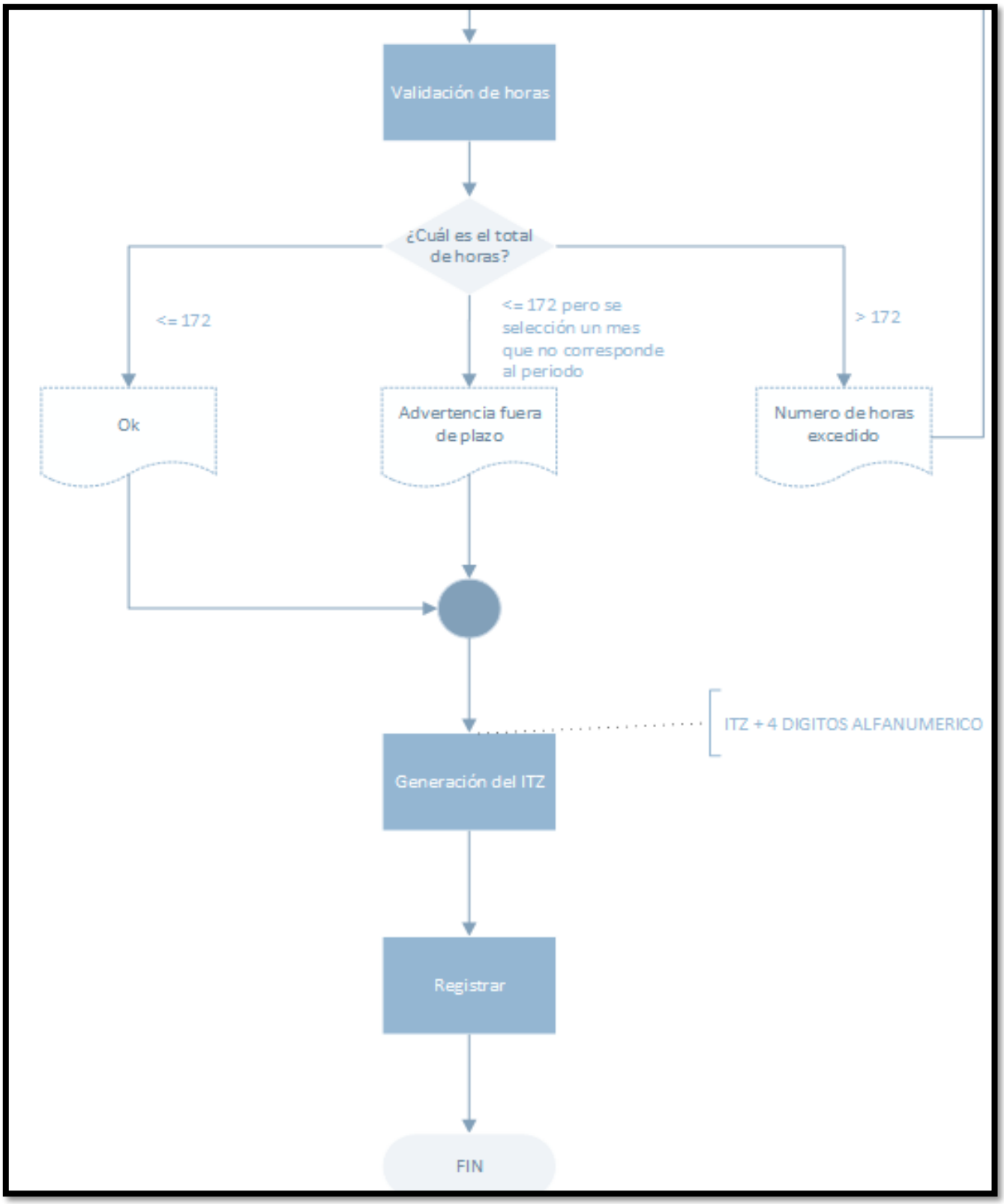
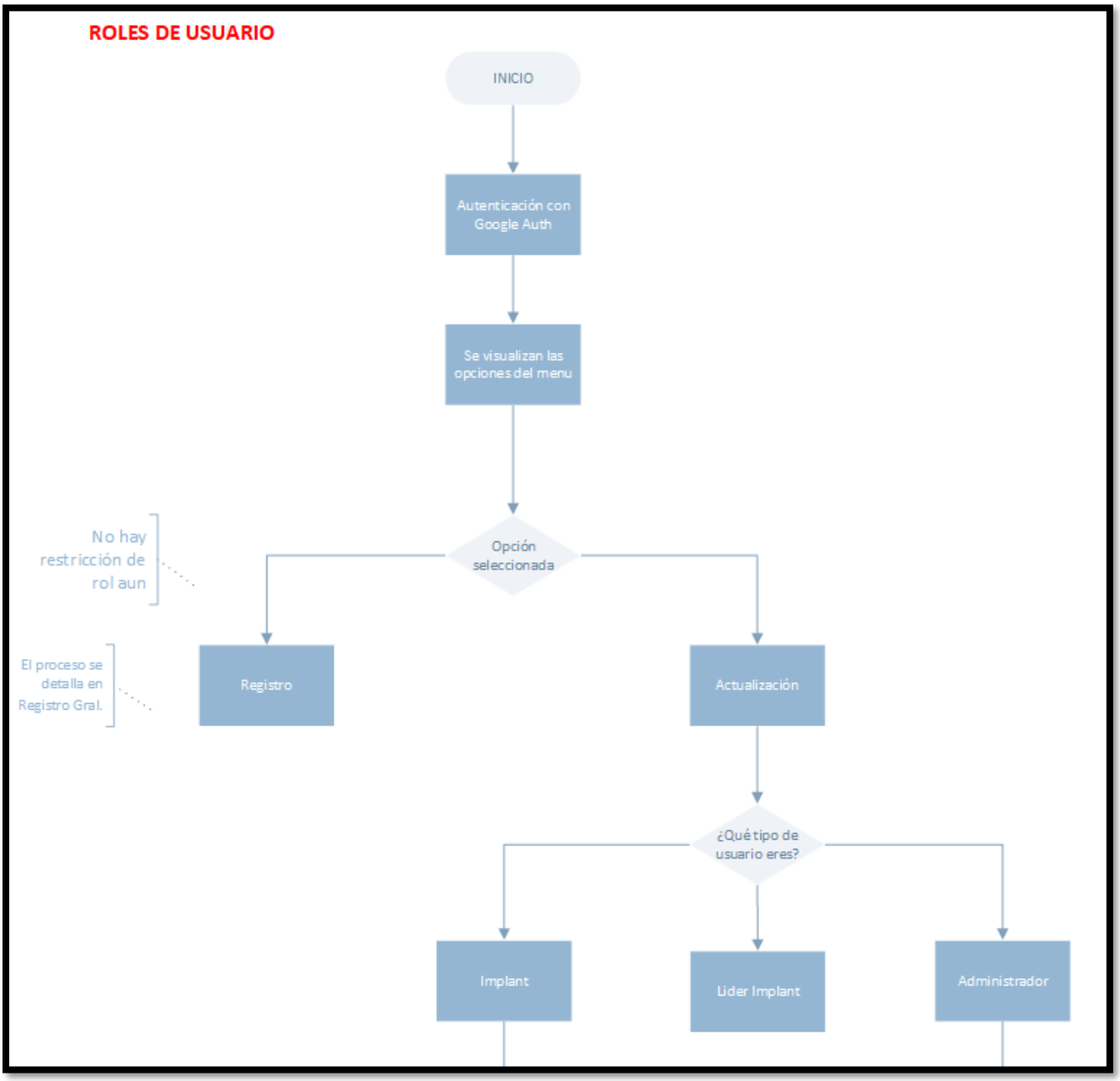


Figura 3.5.1.3: Diagrama de actividades del registro de proyecto tarifario.

Montoya, J (2018). [Diagrama]. Diseño propio.

Roles y permisos

Los roles y permisos con los que debe contar la plataforma se defines en el siguiente figura.



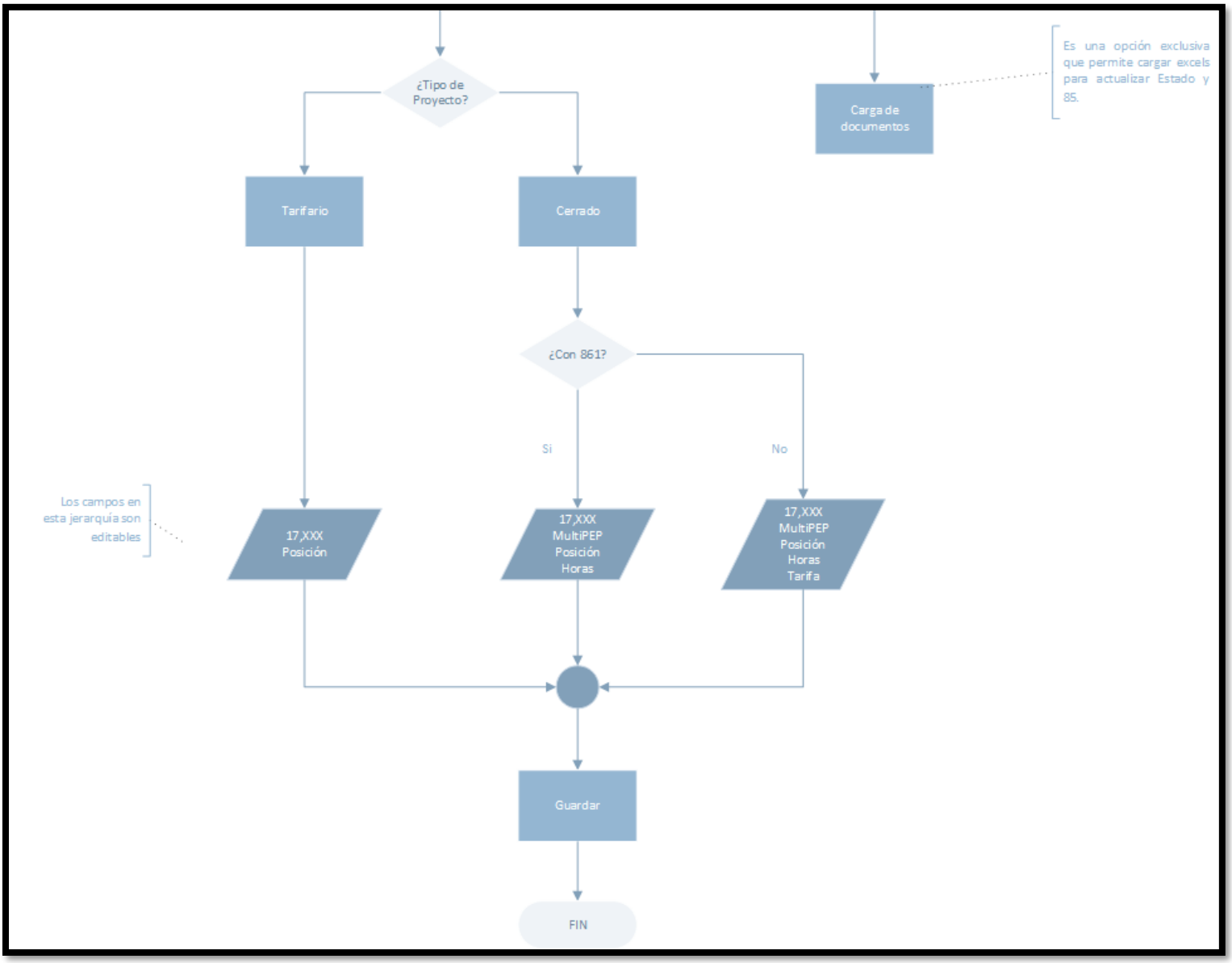


Figura 3.5.1.4: Diagrama de actividades de roles y permisos.

Montoya, J (2018). [Diagrama]. Diseño propio.

3.6 Diseño del modelo de datos

Para realizar el diseño de una base de datos es oportuno recordar, que los datos se construyen a partir de lo que utiliza un negocio y no son los datos quienes condicionan la lógica de negocios, ni de procesos de ningún sistema.

La lógica de negocio que maneja Implant para presupuestos, es sumamente compleja. Por lo cual el diseño de la base de datos tomo un tiempo considerable. Este apartado se dedicará a detallar que entidades fueron identificadas, listar las

relaciones entre ellas y posteriormente presentar el diagrama en donde se mapean las relaciones.

3.6.1 Definición de entidades

La lista de entidades identificadas es:

- Archivo.
- Moneda.
- Usuario.
- Pep.
- Horas.
- Proveedor.
- Proyecto.
- Recurso.
- Seguimiento.

3.6.2 Lista de relaciones

La lista de entidades identificadas es:

- Una moneda pertenece a un seguimiento.
- Muchos usuarios pertenecen a más de un seguimiento.
- Un usuario tiene un rol.
- Muchos PEP pueden pertenecer a muchos seguimientos.
- Un PEP puede pertenecer a un registro de horas.
- Un proveedor tiene más de un perfil.
- Un recurso tiene más de un perfil.
- Muchos seguimientos tienen muchos recursos.
- Muchos seguimientos tienen muchos proveedores.
- Muchos seguimientos tienen muchos proyectos.
- Un seguimiento tiene un tipo.
- Un seguimiento tiene una metodología.
- Un seguimiento tiene un estado.
- Un seguimiento tiene muchos seguimientos de cerrado.

- Un seguimiento tiene muchos seguimientos tarifario.
- Un seguimiento tiene muchos seguimientos servicio.
- Un seguimiento tiene muchos seguimientos servicios extendidos.
- Un seguimiento tiene muchos seguimientos licencias.
- Muchas horas tienen muchos registros de proveedores.
- Muchas horas tienen muchos registros de recursos.

3.6.3 Modelo relacional

La imagen 18 muestra el diagrama relacional de la plataforma:

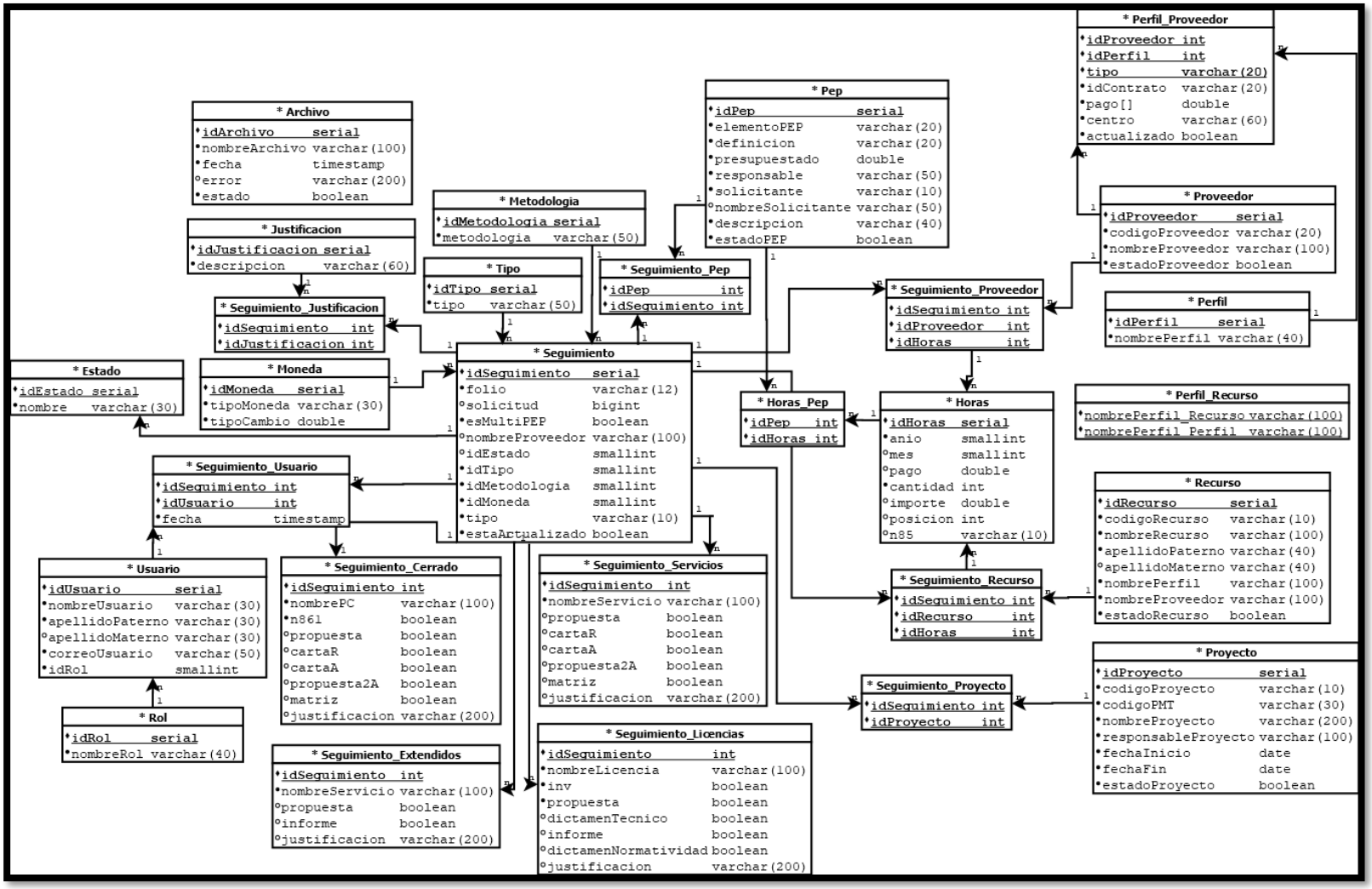


Figura 3.6.3.1: Modelo relacional de la BD de Implant.

Montoya, J (2018). [Diagrama]. Diseño propio.

3.7 Diseño de interfaces

Son dos tipos de interfaces que se mostrarán a continuación. Una son las interfaces que inicialmente indico el cliente y además están aquellas que se desarrollaron de alto nivel, a las cuales mediante tareas asignadas, fueron dándosele los estilos que el cliente solicitaba. Es por ello, que existen cierta diferencia entre uno y otro, ya que lo que se pretendía demostrar con unas era la idea del cliente y con otras un prototipo, para posteriormente y de forma incremental adecuar mejoras a la idea conforme el desarrollo.

Es importante mencionar que en algunos casos lo correcto es desarrollar el diseño inicialmente y en otros, generar una estructura funcional, para aplicar paulatinamente los diseños que se soliciten o generen las áreas de diseño. En este caso por requerimiento y agilidad del desarrollo del producto se optó por la segunda opción.

3.7.1 Wireframes de la plataforma

Los diseños que se muestra continuación son los mockups que solicito el cliente, posteriormente el equipo de desarrollo de encargo de darle forma y estilizar la misma idea.

Registro general

La siguiente imagen nos muestra la definición del wireframe del registro general.

Figura 3.7.1.1: Wireframe de la pantalla de registro general.

Montoya, J (2018). [Diagrama]. Diseño propio.

Registro de proyecto tarifario

La siguiente imagen nos muestra la definición del wireframe del registro de proyecto tarifario.

Figura 3.7.1.2: Wireframe de la pantalla de registro de proyecto tarifario.

Montoya, J (2018). [Prototipo]. Diseño propio.

Semáforos en proyecto tarifario

La siguiente imagen nos muestra el wireframe de las advertencias para la validación de horas en proyecto tarifario.

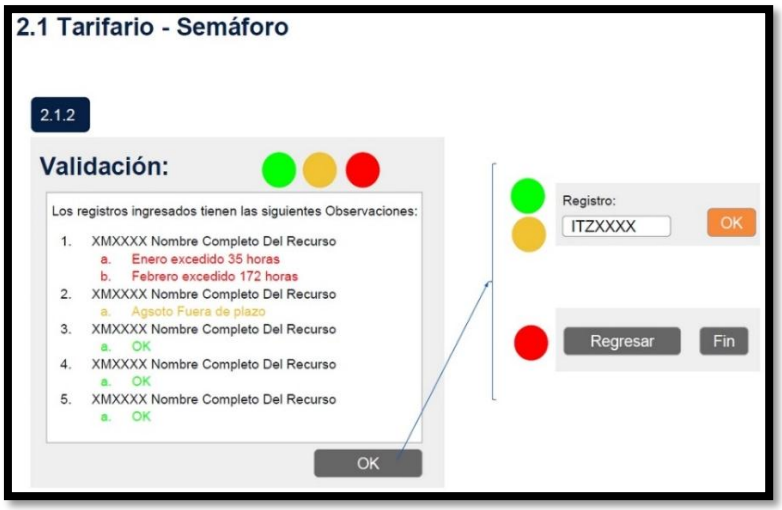


Figura 3.7.1.3: Wireframe de las validaciones de horas para proyecto tarifario.

Montoya, J (2018). [Prototipo]. Diseño propio.

Registro de proyecto cerrado

La siguiente imagen nos muestra la definición del wireframe del registro de proyecto cerrado.

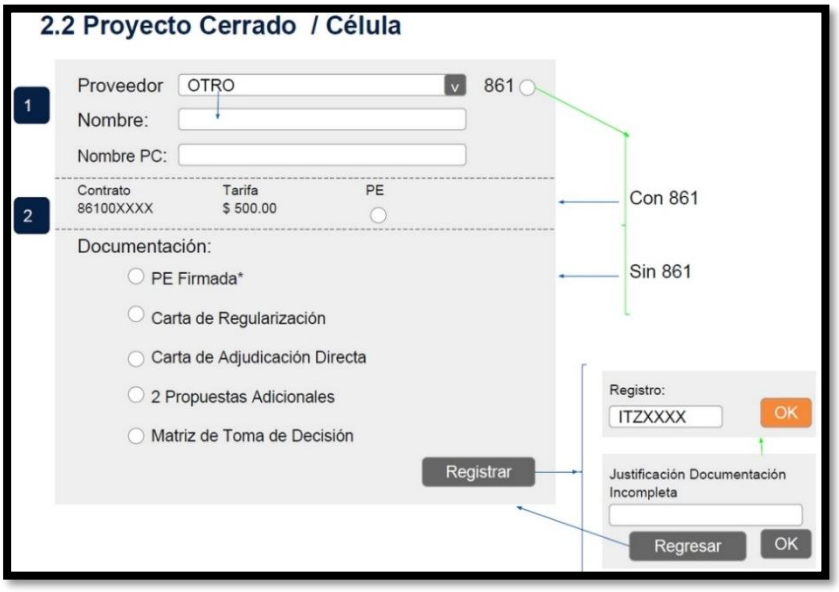


Figura 3.7.1.4: Wireframe de la pantalla de registro de proyecto tarifario.

Montoya, J (2018). [Prototipo]. Diseño propio.

Registro de licencias

La siguiente imagen nos muestra la definición del wireframe del registro de proyecto de licencias.

Figura 3.7.1.5: Wireframe de la pantalla de registro de proyecto de licencias.

Montoya, J (2018). [Prototipo]. Diseño propio.

Registro de servicios extendidos

La siguiente imagen nos muestra la definición del wireframe del registro de proyecto servicios extendidos.

Figura 3.7.1.6: Wireframe de la pantalla de registro de proyecto de servicios extendidos.

Montoya, J (2018). [Prototipo]. Diseño propio.

Registro de servicios

La siguiente imagen nos muestra la definición del wireframe del registro de proyecto de servicio.

Figura 3.6.1.7: Wireframe de la pantalla de registro de proyecto de servicios.

Montoya, J (2018). [Prototipo]. Diseño propio.

Actualización de proyectos

La siguiente imagen muestra la definición del wireframe correspondiente a la actualización.

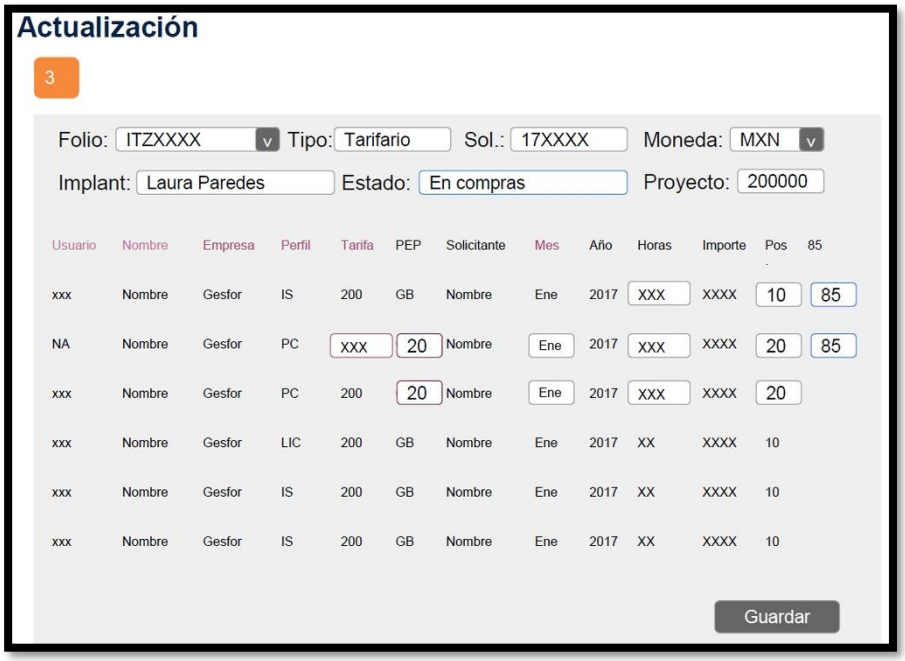


Figura 3.6.1.8: Wireframe de la pantalla de actualización de proyectos. Montoya, J (2018). [Prototipo]. Diseño propio.

3.7.2 Pantallas Implant

Esta sección se presenta unas pantallas de muestra, de la versión final de la plataforma.

Pantalla principal

La figura 3.7.2.1 muestra la pantalla de inicio al entrar a la aplicación.

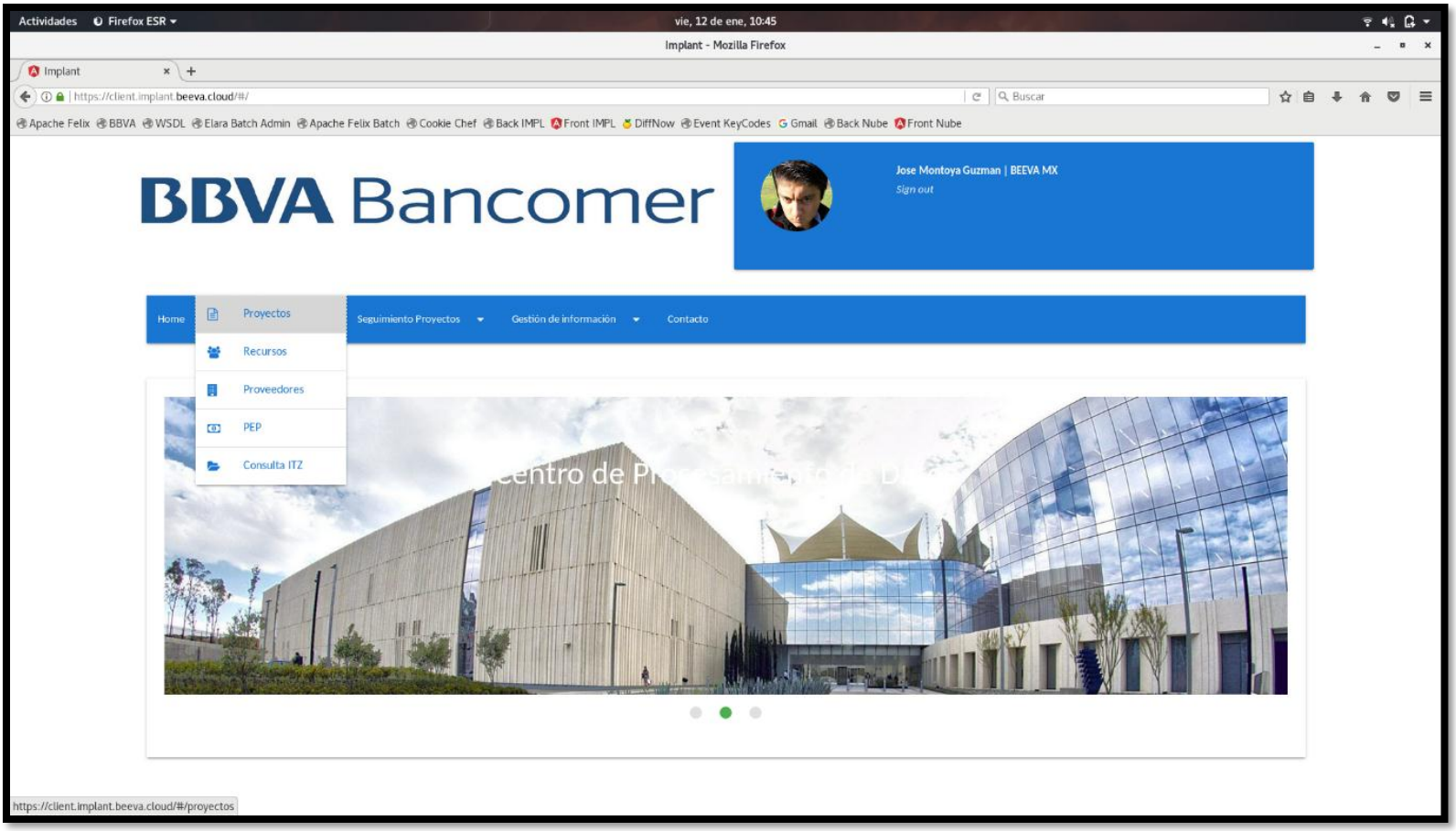


Figura 3.7.2.1: Pantalla principal.

Montoya, J (2018). [Imagen]. Diseño propio.

Consulta de entidades

La figura 3.7.2.2 muestra como es la consulta y filtro de datos.

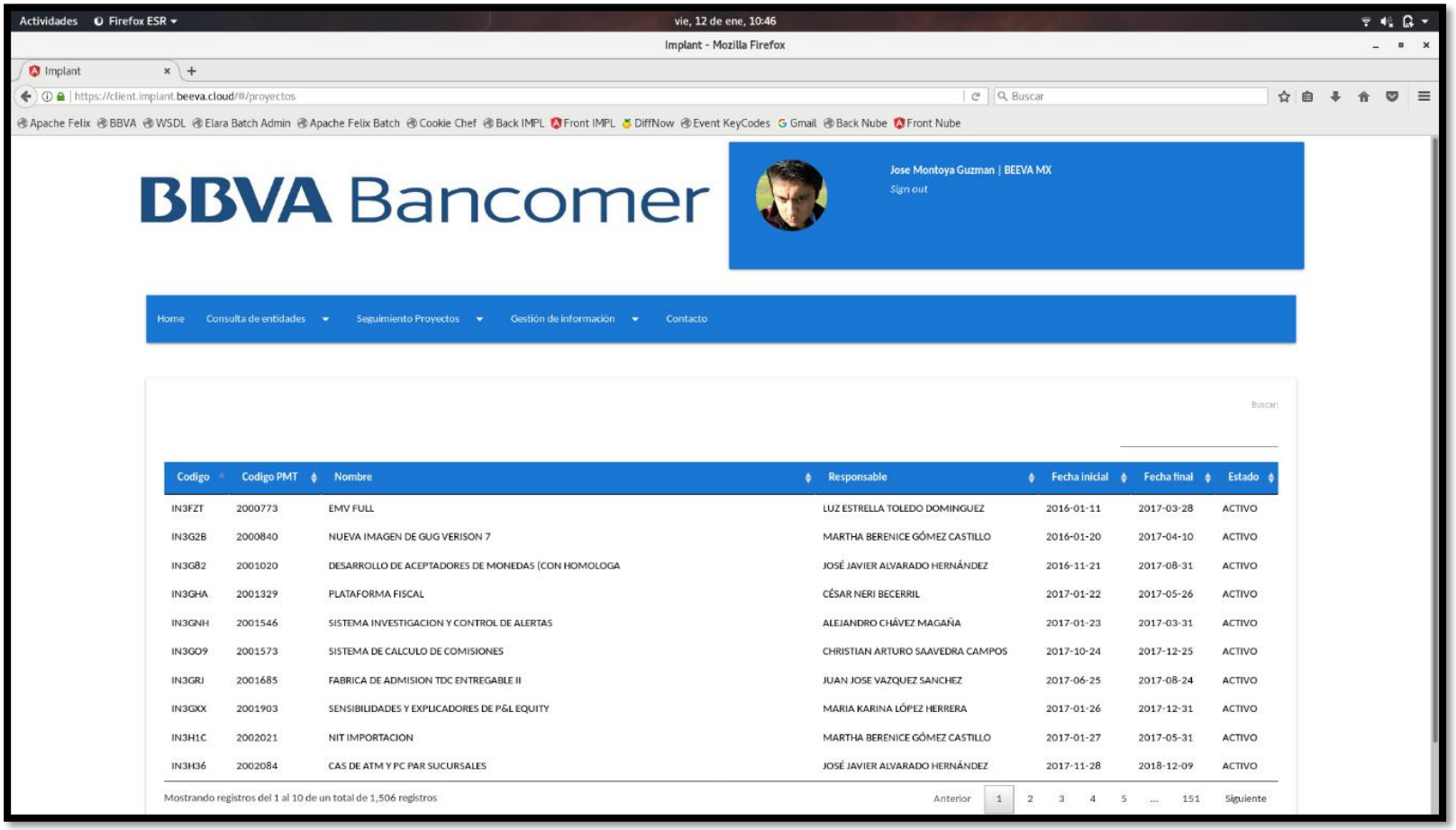


Figura 3.7.2.2: Pantalla consulta.

Montoya, J (2018). [Imagen]. Diseño propio.

Registro general

La figura 3.7.2.3 muestra como es la pantalla de registro general.

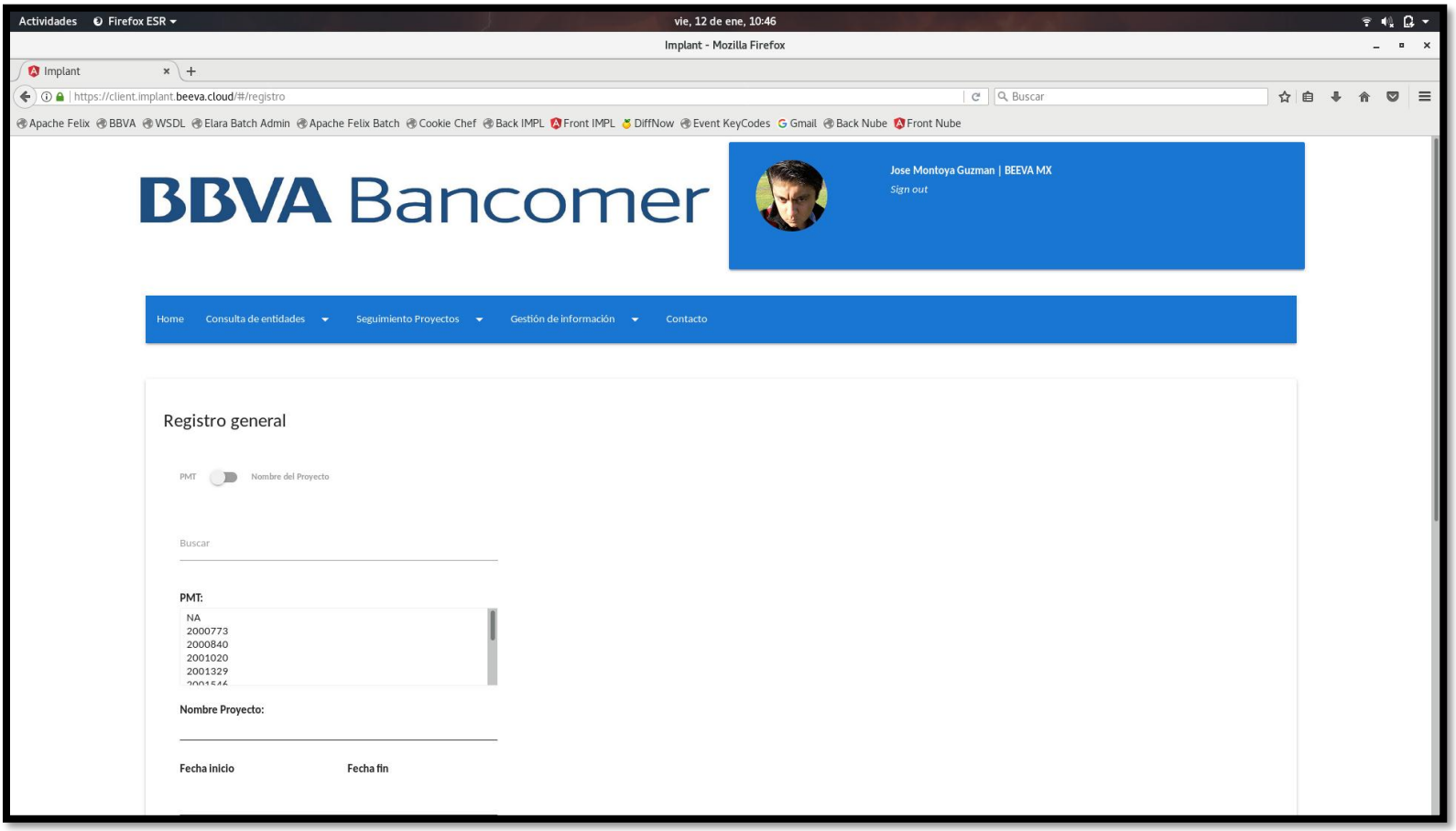


Figura 3.7.2.3: Pantalla de registro general.

Montoya, J (2018). [Imagen]. Diseño propio.

Registro de proyecto tarifario

La figura 3.7.2.4 muestra como es la pantalla de registro tarifario.

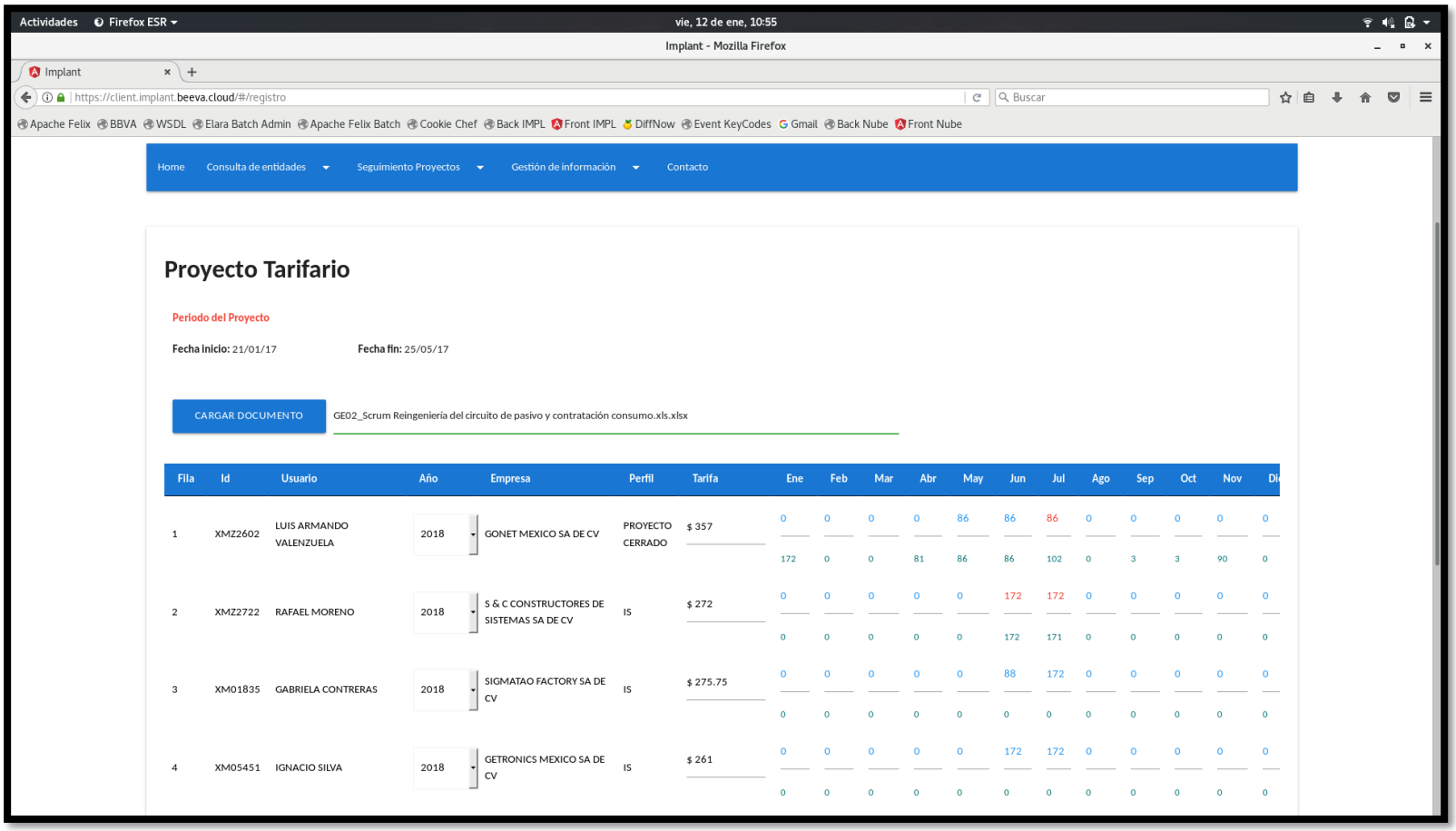


Figura 3.7.2.4: Pantalla registro de proyecto tarifario.

Montoya, J (2018). [Imagen]. Diseño propio.

3.8 Aportaciones del marco SCRUM

La aplicación de marco SCRUM trabajo beneficios considerables al desarrollo de la plataforma, estos se mencionan a continuación:

- Comunicación constante entre el equipo.
- Definición correcta de las tareas.
- Responsabilidades bien definidas.
- Minimizar el impacto de dependencias.
- Entrega de producto de valor.
- Espacio a mejoras.
- Seguimiento del proyecto por cada responsable.

Como conclusión adicional, hay que considerar que los tiempos reales, presupuesto, incidencias y/o dependencias no previstas suelen complicar la aplicación de metodologías o marcos de trabajo y en ocasiones terminan por abandonarse, es importante considerar que para que esto no pase, se deben considerar dos aspectos fundamentales:

- Que el equipo conozca claramente cómo funciona un marco/metodología y por qué debe seguirlo.
- No siempre se puede aplicar el marco/metodología de forma “pura”, se debe tener criterio para ser flexible y tomar lo mejor de ella.

CONCLUSIONES

La terminación de un proyecto siempre genera conocimiento de valor, sobre el trabajo realizado, al respecto quisiera mencionar las siguientes conclusiones.

Trato con el cliente, es un elemento importante para lidiar con recabar requerimientos y manejar cambios, en el proyecto hubo ocasiones donde los requerimientos no estaban claros o bien se cambiaban una vez entregados, el saber entender y delimitar las necesidades de un cliente indica una buena comunicación que impacta en favorablemente a la funcionalidad y tiempos de entrega.

Las reglas de negocio, definen el modelo de datos, anteriormente consideraba que al realizar un sistema primero de debía definir el modelo de datos y posterior a ella las reglas negocio, todo lo contrario noté en este proyecto profesional. Son las reglas de negocio quienes definen los datos de la empresa porque la necesidad de un sistema surge de lo que se trabaja manualmente día a día, los datos son el resultado del trabajo, la finalidad del proceso.

No todo es controlable, existen situaciones donde probablemente pensamos que tenemos la mejor alternativa pero es la decisión del cliente a la que debemos adaptarnos (una vez que se han hecho propuestas claro). Caso concreto con el modelo de datos donde inicialmente se le planteo al cliente el uso de NOSQL para base de datos, pero el cliente decidió SQL por mayor capacidad de integridad de datos, pese a que había elementos que permitían obtener los beneficios de un esquema no relacional.

Aplicación de una arquitectura de software adecuada. El tener una correcta abstracción de las capas de un sistema me permitió que cuando fuese necesario migrar en cierta parte de la aplicación, solo tuviera que hacerse ese cambio y para las demás capas fuera transparente.

Aplicación de buenas prácticas de desarrollo. Esta es una tarea que en lo personal me gustaba manejar en proyectos anteriores. En este proyecto, me quedo como experiencia que no hay mejor acción de desarrollo que seguir las buenas prácticas y convenciones, porque facilita la legibilidad de código y su mantenimiento.

RECOMENDACIONES

Concluido el proyecto me queda claro que los sistemas permiten ahorrar muchos procesos y recursos de la logística de una empresa, entre más grande sea, mayor el impacto, además de mantener la integridad de la información. Estas son las dos principales razones por las que el área Implant debe valorar el proyecto, donde anteriormente no se homologaban estos puntos específicos.

Si bien es cierto que las dependencias de información, metadatos y documentos de otras áreas dificultan cerrar el proceso con una sola plataforma, dados los beneficios aplicativos que se obtuvieron, son alicientes a comenzar a migrar paulatinamente los sistemas de los departamentos que sean obsoletos o complejos. Concretamente esa es mi primera recomendación, si un departamento ha mejorado considerablemente el proceso, y es quien se encargar de la gestión de proyectos, podría comenzar una iniciativa, de migración para los sistemas administrativos, dado que mejorar los procesos de una empresa siempre impacta económicamente a favor.

Por otro lado, si bien “Plataforma de Presupuestos Implant” es una herramienta que ha traído muchos beneficios, también se recomienda la apertura de nuevas fases del proyecto. Durante esta primera fase, se lograron cubrir los requerimientos del área y sus reglas de negocio, sin embargo; de acuerdo a lo analizado durante los feedback con el usuario, existen otras necesidades que también pueden aportarle valor, estas necesidades son suficientes para generar una nueva fase, trayendo consigo más apoyo a los proyectos existentes, a otros proyectos del área (que quedaron fuera de este entregable), a mejorar la experiencia de uso y administración de reglas.

Por la parte técnica recomiendo a la empresa, la posible migración de la base de datos SQL a una de tipo no SQL, durante el trabajo día a día se pudo notar, el porqué de este cambio. Existen las siguientes razones: velocidad en consultas, uso de redundancia de datos, minimizar la complejidad en estructuras y el sharding que puede ofrecer una base de datos no relacional, ya que traerá beneficios de experiencia de usuario e integridad de la información muy notables.

EXPERIENCIA PERSONAL ADQUIRIDA

Con cada nuevo proyecto que realizamos, vienen nuevas experiencias, colaboramos con otros, vemos nuevos enfoques, e incluso, aprendemos nuevas tecnologías, que se vuelven un reto más que aprovechar. Aprender y estar en la disposición de ello es algo que parece la oración de todos los días en la universidad, es como el típico regaño que recibíamos de nuestros padres de pequeños, “¡te vas a caer!”, y que nunca hacíamos caso, pero finalmente nos caíamos…

Pudiera pensarse que al ser mayores y profesionales de tecnología, que entender la idea de mantenernos en aprendizaje constante, es simple y evidente, pero no lo es. Tanto en caso propio, como en compañeros que conocí como universitario y después como empleado, pude darme cuenta de ello. Y esto es porque no dimensionamos la demanda que existe y la gran cantidad de tecnologías y herramientas que van surgiendo y actualizándose constantemente. En ocasiones pensamos que, con ser bueno en unas cuantas cosas, será suficiente para ingresar de buena forma en la industria, y nada es más falso. Si bien, no todo es estudiar o aprender y estoy a favor de la recreación y el descanso, hay situaciones que considerar.

¿Qué pasa si eres de los que no te gusta seguir aprendiendo? ¿Qué pasa si en tu trabajo deciden cambiar la tecnología de trabajo actual? ¿Y si surge una nueva tecnología que es mejor pagada, y por limitarte detienes tu crecimiento? En escenarios profesionales como los mencionados, no basta saber algo, hace falta la disposición, como programador, de saber adaptarse y crecer. Tan claro es que lo que aprendemos a fondo en la universidad son las bases, para poder reconocer que existen muchas herramientas distintas.

Entre mi experiencia profesional quise mencionar esto, porque muchas veces la inexperiencia no nos hace verlo.

También suele ocurrir que el miedo de salir de lo que es cómodo para nosotros nos conforma, o bien, el tercer caso puede ser que, simplemente dejemos pasar el tiempo por pensar que si es necesario aprender algo, lo haremos en cuanto sea estrictamente necesario.

En mi opinión, el tercer caso es el peor, porque el tardarse en reconocer una necesidad o en enfrentar nuestros miedos es algo que por ley de la vida en algún momento entenderemos pero, creer que podemos aprender mucho de un momento a otro es un mal pensamiento no solo para un tecnólogo sino para cualquier persona, es algo que se deriva de la desidia, exceso de confianza e irresponsabilidad porque no proveemos que no siempre la situación será un entorno controlado, ni pensamos en los inconvenientes que pudieran surgir y que en ocasiones no dependen de nosotros resolver, como enfermedades o percances, por mencionar algo.

Es resumen, es importante aprender, por gusto y pasión como tecnólogo, como ser humano, porque el avanzar nos hace estar preparados ante las situaciones que no podemos prever.

¡Sigue, el camino está para recórrelo!

COMPETENCIAS DESARROLLADAS

A continuación, mencionare cuáles fueron las competencias que pude desarrollar durante el proyecto y cómo me fueron aportando experiencia en el desarrollo de productos de software.

Análisis: esta competencia la pude desarrollar desde un inicio, ya que a pesar de tener ideas sobre lo que se requería, no se tenía nada concreto, solo propuestas que fue necesario definir para poder comenzar a definir lo necesario para trabajar.

Capacidad crítica y autocrítica: fundamental competencia durante el proyecto, muchas ocasiones había requerimientos o reglas no definidas correctamente donde el cliente desvariaba mucho sobre lo que quería y poder orientarlo en definir lo que realmente necesitaba reduce tiempo de desarrollo y satisfacción con el cliente. Así mismo, también es importante la autocrítica porque me dio pie a saber cuándo se puede mejorar un trabajo para que éste realmente sea de calidad, sin la necesidad de que otros nos hagan el comentario.

Habilidades de búsqueda y selección de información: dicha competencia me fue muy útil durante la investigación de nuevas tecnologías, como no había desarrollado anteriormente con Angular o Java Spark pude encontrar la forma de investigar. Darme cuenta de dónde y cómo buscar/seleccionar información. Así mismo también fue aplicaba para saber cuándo es necesario profundizar en un tema y cuándo no es lo recomendable.

Generación de nuevas ideas: esta competencia me fue útil en el momento de desarrollar propuestas o ideas que pudieran simplificar procesos y/o funcionalidades pero que esto no afectara la idea o necesidad del usuario.

Capacidad de aplicar conocimientos a la práctica: el saber migrar de lo teórico a lo práctico, es algo que suele ser complicado siempre que tenemos nuevos conocimientos, pero esta competencia me permitió encarar el reto de la mejor formar posible, el saber cómo es importante aplicar lo que aprendemos en pequeños periodos para reforzar el conocimiento y agilizar el aprendizaje. En entornos colaborativos esto da valor a los recursos de una empresa.

REFERENCIAS

Bibliográficas

Allamaraju, Subbu (2010). *Introduction to REST. RESTful Web Services Cookbook.* California, EU: O´Really.

Amazon Web Services Inc. (2018). *AWS Lambda: Guía para desarrolladores.* EU: Amazon Web Services.

Bieberstein, Norbert (2005). *SOA Service-Oriented Architecture Compass. EU:* IBM Press.

Echeverria, Jose (2012). *Design of software process.* La Habana, Cuba: Instituto Politécnico Jose Antonio Echeverría.

Flanagan, David (2011). JavaScript subsets an extensions. En JavaScript The Definitive Guide. California, EU: O´Really pp. 320-345.

Gothelf, Jeff (2016). *Lean UX: Designing Great Products with Agile Teams.* California, EU: O'Reilly.

Kalin, Martin (2009). *Java Web Services: Up and Running.* California, EU: O´Really.

Momjian, Bruce (2000). Customizing Queries. En PostgreSQL Introduction and Concepts. New Jersey, EU: Pearson Education pp. 23-55.

Richarson, Leonard, Ruby Sam (2007). The Resource-Oriented Architecture. En Restful Web Services. California, EU: O'Reilly pp. 79-105

Stallman, Richard (2004). Como promover el software libre. En Software para una sociedad libre. Madrid, España: Mapas pp. 63-66.

Digitales

Angular (2018). What is angular? Recuperado de <https://angular.io/docs>.

ALIA2NET (2015). ¿Cuál es la diferencia entre el diseño web y desarrollo web? Recuperado de <https://alia2net.com/alia2net-proveedor-de-soluciones-para-su-sitio-web/cual-es-la-diferencia-entre-el-diseno-web-y-desarrollo-web/>.

Blancarte, Oscar (2017). Escalabilidad Horizontal y Vertical**.** Recuperado de < https://www.oscarblancarteblog.com/2017/03/07/escalabilidad-horizontal-y-vertical/>.

Blog 4r (2014). UX y UI: ¿cuáles son las diferencias? Recuperado de <http://www.4rsoluciones.com/blog/ux-y-ui-cuales-son-las-diferencias-2/>.

Blogger (2012). Cloud Computing. Recuperado de <http://cloud-fi.blogspot.com/2012/11/antecedentes-cloudcomputing-no-es-un.html>.

Blue chip (2015). Los 8 principios del diseño SOA. Recuperado de < http://bluechip.ignaciogavilan.com/2015/07/los-8-principios-del-diseno-soa.html#.Wx2D-UgvxPZ>.

Developeando (2014). Tipos de desarrolladores web**.** Recuperado de < http://developeando.net/tipos-de-desarrolladores-web/>.

Digital Guide (2017). UI: qué define a una buena interfaz gráfica de usuario. Recuperado de <https://www.1and1.mx/digitalguide/paginas-web/diseno-web/ui-que-es-una-interfaz-de-usuario/>.

Dos Ideas (2008). Introducción a los servicios web RESTful. Recuperado de < https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful/>.

ENAE (2010). Ventajas y desventajas del Cloud Computing. Recuperado de < http://www.enae.es/blog/ventajas-y-desventajas-del-cloud-computing#gref>.

Fachin, José (2017). ¿Qué es la experiencia de usuario y cómo mejorarla en la web de tu negocio? Recuperado de <https://webescuela.com/experiencia-de-usuario/>.

IBM (2018). Arquitectura orientada a servicios. Recuperado de <https://www.ibm.com/support/knowledgecenter/es/SSFPJS_8.5.6/com.ibm.wbpm.wid.main.doc/prodoverview/topics/csoa.html>.

IBM (2018). Arquitectura orientada a servicios**.** Recuperado de <https://www.ibm.com/support/knowledgecenter/es/SSFPJS_8.5.6/com.ibm.wbpm.wid.main.doc/prodoverview/topics/csoa.html>.

Ida Blog (2015). ¿Qué es cloud computing? Recuperado de < https://www.ibm.com/cloud-computing/mx-es/learn-more/what-is-cloud-computing/>.

Ida Blog (2015). ¿Qué esperamos de un desarrollador back-end? Recuperad.o de <https://blog.ida.cl/desarrollo/que-esperamos-de-un-desarrollador-back-end/>.

MegaPractical (2017). Los Microservicios Vs. Arquitectura SOA: el gran debate. Recuperado de <https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/los-microservicios-vs-arquitectura-soa-el-gran-debate>.

MegaPractical (2017). Top 5 Metodologías de Desarrollo de Software. Recuperado de <https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/metodologias-de-desarrollo-de-software>.

Neosystems (2014). Desarrollo web: 6 Fases para el desarrollo de un proyecto. Recuperado de <http://www.neosystems.es/es/noticias/desarrollo-web-6-fases-para-el-desarrollo-de-un-proyecto>.

OpenClassrooms ES (2017). ¿Qué es el desarrollo web? Recuperado de < http://blog.openclassrooms.com/es/2017/09/11/que-es-el-desarrollo-web/>.

OK Hosting (2018). Metodologías del desarrollo de software. Recuperado de < https://okhosting.com/blog/metodologias-del-desarrollo-de-software/>.

Pedraza, Albert (2014). ¿Qué es desarrollo front end? Recuperado de < https://desarrollofrontend.com/que-es-desarrollo-frontend/>.

Power Data (2014). Qué es la arquitectura orientada a servicios SOA. Recuperado de <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/394442/qu-es-la-arquitectura-orientada-a-servicios-soa>.

Power Data (2014). Los principios de la arquitectura orientada a servicios. Recuperado de <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/394446/los-principios-de-la-arquitectura-orientada-a-servicios>.

Rodríguez, J (2011). Método de las 6D. Recuperado de <http://algoritmoyestructuradedatos.blogspot.com/2011/08/metodo-de-las-6d.html>.

Sales Force (2017). Cloud Computing - Aplicaciones en un solo tacto. Recuperado de <https://www.salesforce.com/mx/cloud-computing/>.

Sistemas VD (2008). Fases del proceso de desarrollo de software. Disponible en: <https://sistemasvd.wordpress.com/2008/07/05/fases-del-proceso-de-desarrollo-del-software/>

SOA (2011). SOA y Web Services. Recuperado de < http://soawebs.blogspot.com/2011/01/soa-y-web-services.html>.

SearchDataCenter (2018). Comparación de servicios de big data entre AWS, Azure y Google. Recuperado de < https://searchdatacenter.techtarget.com/es/cronica/Comparacion-de-servicios-de-big-data-entre-AWS-Azure-y-Google>.

Universidad de Alicante (2014). Introducción a los Servicios Web. Recuperado de <http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html>

Wang, A., Chang, A., Mark, A., Louie, K. (2018). Materialize Dociencia. Recuperado de <http://www.um.es/docencia/barzana/materializecss/>.

Xperience Group (2017). Cloud Vs On Premise Software: Which is Best For Your Business? Recuperado de <https://www.xperience-group.com/blog/cloud-vs-on-premise-software/>.

ZDNet (2018). Top cloud providers 2018. Recuperado de <https://www.zdnet.com/article/cloud-providers-ranking-2018-how-aws-microsoft-google-cloud-platform-ibm-cloud-oracle-alibaba-stack/>.