Desarrollo de un software para la aplicación y monitoreo de la evaluación de desempeño en administrativos de la Corporación Universitaria Del Huila - CORHUILA

Presentado por:

Juan Sebastián Parrasi Yanguma

Corporación Universitaria del Huila CORHUILA

Facultad de Ingeniería

Ingeniera de Sistemas

Neiva – Huila

2019

DESARROLLO DE UN SOFTWARE PARA LA APLICACIÓN Y MONITOREO DE LA EVALUACIÓN DE DESEMPEÑO EN ADMINISTRATIVOS DE LA CORPORACION UNIVERSITARIA DEL HUILA - CORHUILA.

Presentado por:

Juan Sebastián Parrasi Yanguma

Director del Proyecto

Ing. Eduardo Martínez Vidal

Corporación Universitaria del Huila CORHUILA

Facultad de Ingeniería

Ingeniera de Sistemas

Neiva – Huila

2019

Dedicatoria

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme el haber llegado hasta este momento tan importante de mi formación profesional llenándome de sabiduría, inteligencia y fuerzas para superar cada etapa de mis estudios. A mis padres, por ser el pilar más importante y por demostrarme siempre su cariño y apoyo incondicional. A mi esposa e hijo que son el motor de mi vida, que todo lo que hago es por ellos y para ellos.

1 Tabla de contenido

Lista de tablas

Lista de Ilustraciones

1. Área problemática y problema de investigación

1.1 Área problemática

En términos de aportar a la eficiencia y eficacia en los procesos administrativos que contribuyen al cumplimiento de la calidad institucional de CORHUILA, es de carácter perentorio realizar una evaluación y medición de todas las áreas que compone el organigrama de la organización junto con el recurso humano que lo integra, partiendo de la alta gerencia hasta la subordinación; que para tal fin se requiere realizar una evaluación de desempeño de cada funcionario del área administrativa.

En relación al número de empleados que conforma el área administrativa de CORHUILA para la aplicación y monitoreo de la evaluación de desempeño a cada una de estas personas, por parte de la división de Talento Humano se convierte en una labor compleja en razón que se puede dar un cuello de botella por otras obligaciones que se tiene con la empresa como liquidación de nómina y otros servicios que se prestan a la organización, esto puede influir en el límite de la capacidad productiva del personal de la oficina de Talento Humano, teniendo presente si se quiere hacer una óptima medición y análisis de los datos recolectados por el instrumento de la evaluación de desempeño y elaboración de informes como resultado de esta labor que se presenta a la alta gerencia, teniendo presente que esta actividad se desarrolla de manera manual y por semestre.

En este orden de ideas se pone a consideración desarrollar un software que permita realizar el proceso de aplicación y monitoreo de desempeño a través de la red interna de CORHUILA

1.2 Problema de investigación

¿De qué manera la implementación de un software en entorno web, mejorará la gestión de procesos manuales para la evaluación de desempeño al personal administrativo de la Corporación Universitaria del Huila CORHUILA?

2 Justificación

Las tecnologías de información y comunicación ofrecen una gran oportunidad para el desarrollo y el avance de universidades e institutos de educación superior, que traen múltiples aportes a los diversos sectores de la sociedad. Estas tecnologías permiten la interacción a distancia de actores, el procesamiento y almacenamiento de grandes cantidades de datos. En síntesis, estas tecnologías son herramientas valiosas en la gerencia y el control empresarial e institucional (Garcia Sanchez, Reyes Añorve, & Godinez Alarcon, 2017).

En la actualidad, es necesario que la información sea procesada y almacenada de una forma efectiva para agilizar los procesos y así lograr un control integral de estas actividades.

El desarrollo de un Software que sirva de apoyo al proceso de control y seguimiento a la evaluación de desempeño administrativo de la Corporación Universitaria del Huila CORHUILA, proporcionará efectividad y eficiencia en el manejo y procesamiento de los grandes volúmenes de información que requieren este tipo de procesos, ejecutando las actividades con el menor esfuerzo humano y en el menor tiempo posible.

Al implementar el software los instrumentos de evaluación se reflejarán de manera digital, los evaluadores diligenciarán los instrumentos en línea, el sistema arrojara alertas a los evaluadores recordando la fecha límite y se podrán monitorear las evaluaciones realizadas. El jefe de la oficina de talento humano junto con la alta gerencia, podrá tomar decisiones

respecto a la mejora continua del servicio, procesos y de la calidad de la institución, basado en los informes generados por el sistema.

3 Objetivos

3.1 Objetivo general

Desarrollar un software que permita aplicar, controlar y hacer seguimiento a la evaluación de desempeño de administrativos de la Corporación Universitaria del Huila CORHUILA

3.2 Objetivo específicos

- Analizar e identificar los requerimientos del software, mediante entrevistas y formatos de evaluación entregados por el área de Talento Humano
- Diseñar los casos de uso, base de datos y la interfaz gráfica del software haciendo uso de buenas prácticas para la calidad del software
- Realizar las pruebas unitarias y funcionales del software
- Implementar en entorno de producción el software desarrollado

4 Referente teórico

La IEEE define la Ingeniería de Software como “la aplicación de un enfoque sistemático, cuantificable y disciplinado al desarrollo, operación y mantenimiento del software” (IEEE Computer Society, 2014); y desde su aparición formal a finales de los años 60, ha evolucionado notoriamente en teorías, métodos, procesos y herramientas para dar soporte a la industria del software.

Uno de los logros en el desarrollo de la Ingeniería de Software, es la definición de una guía para el cuerpo del conocimiento: SWEBOK, la cual permite caracterizar la profesión y presenta de manera global las prácticas, herramientas y métodos que se usan actualmente en el escenario de la ingeniería de software.

El SWEBOK abarca diez (10) áreas de conocimiento especificadas en la ingeniería de Software y ocho (8) disciplinas de soporte (Guia al cuerpo de conocimiento de la ingenieria de software SWEBOK, 2004).

Considerando que el SWEBOK presenta un acceso a los temas del conocimiento de la Ingeniería de Software, es relevante para el presente trabajo estudiar y aplicar esta guía, por el aporte significativo en la clarificación de las áreas del conocimiento y disciplinas; información que es importante para cumplir con el propósito del trabajo de investigación propuesto, aplicando de manera práctica en el proceso de desarrollo de software que se pretende implementar en el proyecto.

Las áreas del conocimiento del SWEBOK, que se consideran con mayor prioridad en relación con el proyecto a realizar son:

Requerimientos de Software: Es un área de conocimiento crucial para cualquier proyecto de desarrollo de software; en esta etapa se obtiene las especificaciones, análisis y validación de los requisitos de las expectativas del cliente respecto al producto a construir. Para el proyecto que se pretende realizar, es crítico porque influye de manera significativa en todo el ciclo de vida del software; al no tener de forma clara las especificaciones del cliente, las posibilidades de fracaso son altas en el desarrollo de un producto que no satisfaga los requerimientos solicitados.

Se desea proponer un proceso que permita entregar un producto, en plazos de tiempo programados, sin dejar a un lado el diseño, pruebas, mantenimiento, configuración y la aplicación de modelos y métodos; que parten del análisis de requisitos; en esta fase se aplicarán las técnicas de creatividad que contribuyan a la construcción de Software innovador (IEEE Computer Society, 2014).

Ingeniería de procesos de Software: Esta área de conocimiento corresponde al capítulo ocho (8) de la guía, que consiste en una serie de actividades para llevar a cabo una transformación de productos a partir de la coordinación y la gestión de proyectos para medir y mejorar la

calidad del producto que se está construyendo, a través de una mejora de procesos continua que incluye modelos y métodos (IEEE Computer Society, 2014).

Para el presente proyecto de investigación se tendrán como referentes los temas de definición de procesos de software, evaluación de procesos de software y mejora, y herramientas de ingeniería de proceso de software. Estos temas se deben tener en cuenta en el escenario de desarrollo de software de la ciudad en la CORHUILA, en busca de la calidad y mejora de procesos en cada una de las actividades en el proceso de desarrollo propuesto.

Las principales actividades en las cuales se harán especial énfasis en el proceso a implementar son: Planificación, Implementación, Documentación, Despliegue y Mantenimiento; actividades que definen el proceso de desarrollo de software siguiendo la política que las entradas de criterios y responsabilidades de tareas asignadas, con el propósito de controlar las actividades y realizar mejoras continuas cuando estas presenten alguna inconsistencia en su proceso de desarrollo.

La aplicación de modelos en el proceso de desarrollo, es un punto clave en razón que para el proyecto a desarrollar se propondrá un proceso que incluirá un conjunto de actividades relacionadas; incluyendo un valor agregado que es el de crear un producto software innovador.

Angular: es una framework que permite desarrollar aplicaciones web en la sección cliente utilizando HTML y JavaScript para que el cliente asuma la mayor parte de la lógica y descargue al servidor con la finalidad de que las aplicaciones ejecutadas a través de Internet sean más rápidas. El hecho de estar mantenido por Google, así como una serie de innumerables razones técnicas, ha favorecido su rápida adopción por parte de la comunidad de desarrolladores.

Permite la creación de aplicaciones web de una sola página (SPA: single-page application) realizando la carga de datos de forma asíncrona (Boada Oriols & Gomez Guiterrez, 2019).

SPA (Single Page Application): Es un tipo de aplicación web donde todas las pantallas las muestra en la misma página, sin recargar el navegador.

Técnicamente, una SPA es un sitio donde existe un único punto de entrada, generalmente el archivo index.html. En la aplicación no hay ningún otro archivo HTML al que se pueda

acceder de manera separada y que nos muestre un contenido o parte de la aplicación, toda la acción se produce dentro del mismo index.html (Alvarez, Desarrollo Web, 2016).

Laravel: es un framework de código abierto para aplicaciones y servicios web PHP escrito por Taylor Otwell (Rees & Laguna, 2013)

PHP: es un lenguaje interpretado del lado del servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos e PHP son embebidos directamente en el código HTML y ejecutados por el servidor web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en forma de código HTML. Es un lenguaje open source y multiplataforma; los programas funcionan igual sobre diferentes plataformas, trabajando sobre la mayoría de servidores web y estando preparado para interactuar con más de 20 tipos de base de datos (Cobo, Gomez, Perez, & Rocha, 2005).

ORM: El mapeo objeto-relacional es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (esencialmente la herencia y el polimorfismo). Entre las ventajas que ofrecen los ORM se encuentran: rapidez en el desarrollo, abstracción de la base de datos, reutilización, seguridad, mantenimiento del código, lenguaje propio para realizar las consultas (Yanes Enriquez & Gracia del Busto, 2011).

MVC (Modelo-Vista-Controlador): El patrón MVC es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite “no mezclar lenguajes de programación en el mismo código”.

MVC divide las aplicaciones en tres niveles de abstracción:

- Modelo: representa la lógica de negocios. Es el encargado de acceder de forma directa a los datos actuando como "intermediario" con la base de datos.
- Vista: es la encargada de mostrar la información al usuario de forma gráfica y "humanamente legible".
- Controlador: es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta, lo presente al usuario, de forma "humanamente legible". (Bahit)

MVP (Modelo- Vista- Presentador): es un patrón derivado del MVC, que nos permite separar de forma muy clara nuestras vistas de la lógica de nuestras aplicaciones.

- Modelo: Esta capa gestiona los datos. Son las clases que denominaríamos de lógica de negocio.
- Vista: Se encarga de mostrar los datos. Aquí se encontrarían nuestros Fragmentos y Vistas.
- Presentador: Se sitúa entre el modelo y la vista, permitiendo conectar la interfaz gráfica con los datos. (Develapps, 2016)

MVVM (Modelo- Vista - VistaModelo): MVVM es una arquitectura desarrollada por Microsoft alrededor de 2004, cuando también se creó Windows Presentation Foundation.

Está dirigido a modernas plataformas de desarrollo de interfaz de usuario que soportan programación orientada a eventos, como HTML5, Windows Presentation Foundation WPF, Silverlight y el Framework ZK.

- Modelo: son los modelos que contienen información en este caso el modelo únicamente se comunica a través de la vista modelo, pero nunca hacia la vista.

- Vista: las vistas son la presentación de los datos de un modelo mediante lo que se procesa en la vista modelo (Comportamientos, eventos y enlace de datos), pero está nunca se comunica directamente con el Model
- Vista Modelo: es el conector entre el modelo y la vista el cuál mediante enlace de datos hace está comunicación, también se dice que aquí se tiene toda la lógica de presentación. (Montero Ortega, 2019)

Web Services: El termino describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuáles son los servicios disponibles. Uno de los usos principales es permitir la comunicación entre las empresas y entre las empresas y sus clientes. Los Web Services permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos Sistemas de Información.

Los Web Services no proveen al usuario una interfaz gráfica (GUI). En vez de ello, los Web Services comparten la lógica del negocio, los datos y los procesos, por medio de una interfaz de programas a través de la red. Es decir, conectan programas, por tanto, son programas que no interactúan directamente con los usuarios. Los desarrolladores pueden por consiguiente agregar a los Web Services la interfaz para usuarios, por ejemplo, mediante una página Web o un programa ejecutable, con el fin de entregarle a los usuarios una funcionalidad específica que provee un determinado Web Service. (Saffirio, 2006)

API: son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un creador de programas, ya que no tiene que

«escribir» códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa. El usuario nunca ve las API en pleno proceso de trabajo, pero sí detalles de sus acciones. La API es una interfaz o rostro que sólo da la cara al software. Es decir, el usuario no ve eso. (ABC Consultorio, 2015)

REST: REST es una interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

El formato más usado en la actualidad es el formato JSON, ya que es más ligero y legible en comparación al formato XML. Elegir uno será cuestión de la lógica y necesidades de cada proyecto.

REST se apoya en HTTP, los verbos que utiliza son exactamente los mismos, con ellos se puede hacer GET, POST, PUT y DELETE. De aquí surge una alternativa a SOAP.

Cuando hablamos de SOAP hablamos de una arquitectura divididas por niveles que se utilizaba para hacer un servicio, es más complejo de montar como de gestionar y solo trabajaba con XML. (Rosa Moncayo, 2018)

SOAP: La funcionalidad que aporta SOAP es la de proporcionar un mecanismo simple y ligero de intercambio de información entre dos puntos usando el lenguaje XML. SOAP no es más que un mecanismo sencillo de expresar la información mediante un modelo de empaquetado de datos modular y una serie de mecanismos de codificación de datos. Esto permite que SOAP sea utilizado en un amplio rango de servidores de aplicaciones que trabajen mediante el modelo de comunicación RPC (Remote Procedure Call). (Cubo Velasquez)

5 Descripción de la propuesta

Se propone un sistema con una arquitectura que permita la escalabilidad e integración continua de la aplicación, que se alinie a los requerimientos funcionales y no funcionales que se presenten en la fase de análisis, diseño y desarrollo de la solución.

En este sentido el sistema se basa en la arquitectura REST siendo el estándar de comunicaciones cliente servidor más usado en el mercado de aplicaciones web y móviles gracias a su facilidad de escalabilidad y sostenibilidad

A partir de esto se desarrollará un software en ambiente web, con conectividad sobre el motor de base de datos Oracle 11g y el Front del cliente en el lenguaje PHP 7.0, aplicando la arquitectura de software Modelo Vista Controlador (MVC).

El resultado de la solución a desarrollar, es la implementación de una herramienta que satisfaga las necesidades correspondientes al proceso de la evaluación de desempeño de la Corporación Universitaria del Huila CORHUILA, controlado y monitoreado por la división de talento humano de la institución.

6 Arquitectura de la solución

En los últimos años los avances en ingeniería web han permitido que las aplicaciones web que se crean sean más potentes visual y funcionalmente (Gomez, 2015).

Debido a este continuo avance, existen una gran cantidad de tecnologías en el lado de cliente que hacen que cada vez sea más complicado desarrollar una aplicación de manera ordenada. Con la ayuda de los frameworks de programación del lado del cliente se busca unir todas esas tecnologías, reducir su complejidad y separar conceptos mediante el patrón MVC y sus variantes para que cuando aumente el tamaño del código no se haga inmanejable su desarrollo.

En el lado servidor ocurre algo parecido, cada vez las aplicaciones son más grandes y hacen uso de diversas tecnologías lo cual implica que se necesiten de más recursos y por tanto la capacidad de procesamiento se ve afectada. Mediante los Web Services seremos capaces de liberar en cierta medida al servidor, haciendo las peticiones más ágiles. De esta manera y al disponer de Web Services, no estamos atados a una determinada tecnología en el lado cliente: tanto un navegador como un dispositivo móvil pueden lanzar peticiones a un API siendo transparente para el servidor (Gomez, 2015).

6.1 Supuestos y dependencias

La definición del sistema se ha realizado teniendo en cuenta los siguientes criterios:

- CORHUILA contará con toda la infraestructura de hardware, software de base, comunicaciones y redes suficientes para la operación del sistema.
- El sistema debe ser diseñado teniendo en cuenta los macro y procesos de gestión los negocios involucrados y no los procesos individuales de cada una de las áreas.

- La arquitectura del sistema debe estar orientada, unificada y generalizada de tal manera que todos los macro procesos queden cubiertos, pero no detallados.
- La arquitectura tecnológica que apoye el sistema debe estar orientada con base en la tecnología disponible y más ampliamente difundida en el mercado.
- Con respecto a la información, el sistema manejará lo que es exclusivo a las acciones descritas en los macro procesos o requisitos de alto nivel.
- Los subsistemas estarán integrados entre sí y con los sistemas tanto de tipo estratégico como de apoyo administrativo de forma automática.
- El sistema deberá contar con una interfaz que permita integrar la solución con sistemas externos de otras entidades con los cuales se requiera la integración de forma automática.

6.2 Restricciones

- Disponibilidad de recursos económicos para contar con los recursos necesarios (hardware, software de base, comunicaciones y redes) para la operación del sistema.
- Los sistemas estratégicos y de apoyo administrativo y contable de la CORHUILA, así como los sistemas existentes que continuarán operando en la entidad, deberán ser adaptados para su integración con el sistema desarrollado.
- Acceso a la red de la CORHUILA, por cualquier medio, físico o móvil.
- El sistema debe ser diseñado considerando los requerimientos de seguridad de la información, de acuerdo con los niveles de criticidad de la información personal manejada por los subsistemas o sistemas de software internos o externos de CORHUILA.
- El sistema debe ser diseñado para ejecutarse a través de un entorno web.

6.3 Metas de la Arquitectura

- Implementar los patrones y estilos de diseño más adecuados para la aplicación.
- Reducir los riesgos tecnológicos del proyecto asociados a sistemas de gran escala.
- Diseñar los componentes de software que soporten la operación del sistema y cumplan con los requerimientos no funcionales.
- Facilitar el diseño, implementación y despliegue de la aplicación.
- Implementación de tecnologías, herramientas y arquitecturas, de manera coherente para que colaboren entre ellos, analizándolos y adecuándolos con base a las necesidades de la aplicación.
- Crear una API RESTful usando Web Services con PHP y Laravel
- Hacer uso de dicha API desde una aplicación web cliente.
- Utilizar el framework Angular para mostrar la información en un navegador web.
- Utilizar los estándares de diseño responsivo para las interfaces gráficas.
- Conseguir abstraer la capa de persistencia en el servidor usando un ORM.
- Usar del patrón MVC en diferentes contextos y variantes (MVP y MVVM)
- Configurar diversos entornos para el proyecto.
- Implementar medidas de seguridad sobre el acceso a la API por medio de usuarios con roles y permisos.
- Integrar todo lo antes mencionado en una única aplicación.
- Crear un modelo arquitectónico siguiendo una metodología

6.4 Requerimientos no funcionales y características de alto nivel

6.4.1 Consideraciones de Seguridad y Privacidad

- Los permisos de acceso al sistema podrán ser cambiados solamente por los usuarios administradores de acceso a datos.
- El nuevo sistema debe desarrollarse aplicando patrones y recomendaciones de programación que incrementen la seguridad de datos.
- Todos los sistemas deben respaldarse al menos cada semana. Los respaldos deben ser almacenados en una localidad segura ubicada en un edificio distinto al que reside el sistema.
- Todas las comunicaciones externas entre servidores de datos, aplicación y cliente del sistema deben estar encriptadas.
- Todos los datos sensibles residentes en bases de datos locales deben estar encriptados para dificultar su acceso a terceros.
- Si se deben identificar ataques de seguridad o brechas del sistema, determinando la procedencia del ataque y bloqueando la dirección IP y/o usuario que esté relacionado con él, el mismo no continuará operando hasta ser desbloqueado por un administrador de seguridad.

6.4.2 Requisitos de Disponibilidad

- El sistema debe tener una disponibilidad del al menos el 99% de las veces en que un usuario intente accederlo.
- El tiempo para iniciar o reiniciar el sistema no podrá ser mayor a 15 minutos.
- La tasa de tiempos de falla del sistema no podrá ser mayor al 1% del tiempo de operación total.
- El promedio de duración de fallas no podrá ser mayor a 15 minutos.
- La probabilidad de falla del Sistema no podrá ser mayor a 1%.

6.4.3 Expectativas de Volumen y Rendimiento

- El sistema debe ser capaz de procesar al menos 2 transacciones por segundo. Esto se medirá por medio de la herramienta de Software Testing de servicios web.
- Toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menos de 5 segundos.
- El sistema debe ser capaz de operar adecuadamente con hasta 1000 usuarios con sesiones concurrentes.
- Los datos modificados en la base de datos deben ser actualizados para todos los usuarios que acceden en menos de 2 segundos

6.5 Arquitectura de Alto Nivel

6.5.1 Servicios REST

Cada día necesitamos más usar servicios web REST (Representational State Transfer) es un estilo de arquitectura para desarrollar servicios web.

Los servicios web que siguen este estilo deben cumplir con las siguientes premisas: cliente/servidor, sin estado, cache, servicios uniformes, arquitectura en capas.

6.5.2 Cliente/Servidor

Como servicios web se pueden clasificar en la arquitectura cliente servidor y definen una interface de comunicación entre ambos separando completamente las responsabilidades entre ambas partes.

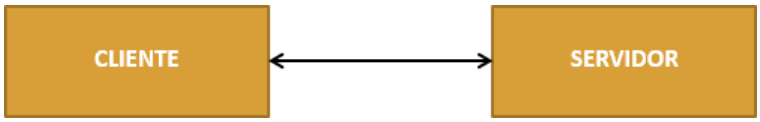

Ilustración 1 Cliente/Servidor.

Fuente Propia

6.5.3 Sin estado

Son servicios web que no mantienen estado asociado al cliente. Cada petición que se realiza a ellos es completamente independiente de la siguiente. Todas las llamadas al mismo servicio serán idénticas

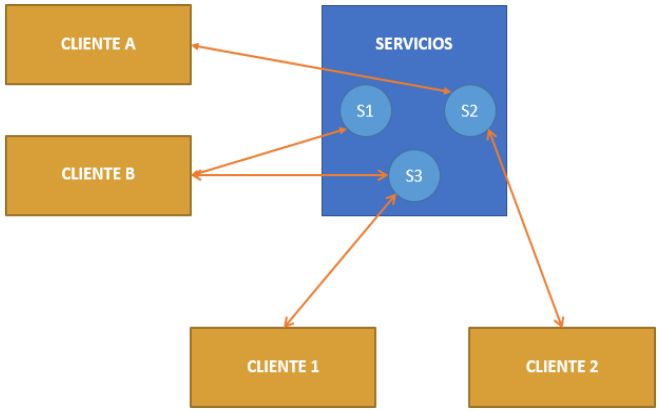

Ilustración 2 Sin Estado

Fuente Propia

6.5.4 Cache

El contenido de los servicios web REST se puede cachear de tal forma que una vez realizada la primera petición al servicio el resto puedan apoyarse en la cache si fuera necesario

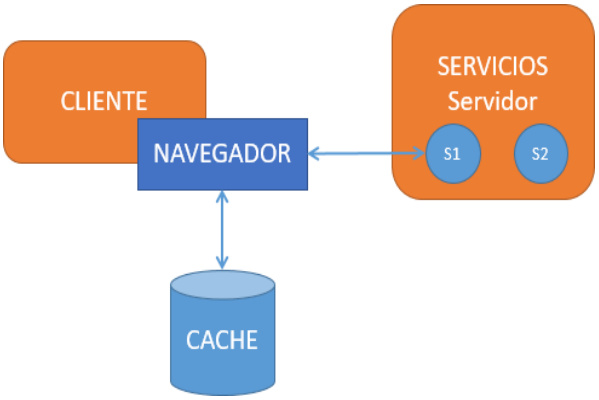

Ilustración 3 Cache

Fuente Propia

6.5.5 Servicios Uniformes

Todos los servicios REST compartirán una forma de invocación y métodos uniforme utilizando los métodos GET, POST, PUT, DELETE.

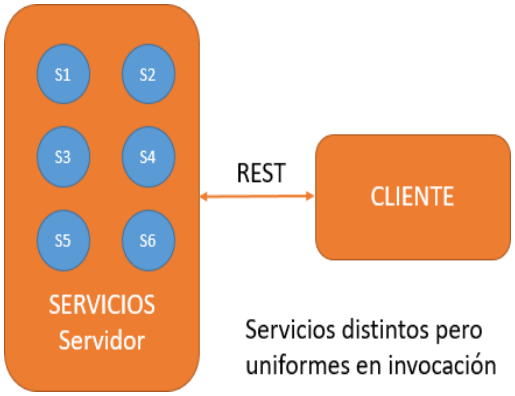

Ilustración 4 Servicios uniformes

Fuente Propia

6.5.6 Capas

Todos los servicios REST están orientados hacia la escalabilidad y un cliente REST no será capaz de distinguir entre sí está realizando una petición directamente al servidor, o se lo está devolviendo un sistema de cache intermedio o por ejemplo existe un balanceador que se encarga de redirigirlo a otro servidor. (Caules, 2013)

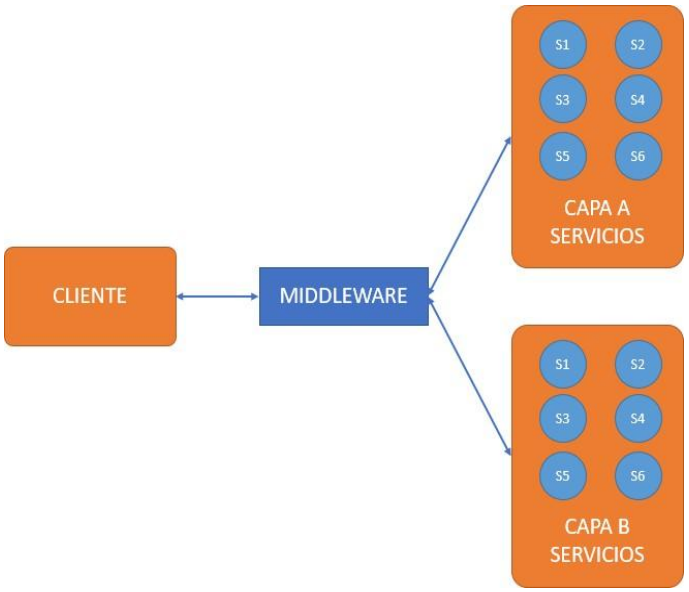

Ilustración 5 Capas

Fuente Propia

6.6 Arquitectura de la Aplicación

La arquitectura seleccionada para el desarrollo del sistema es una arquitectura n-capas con cliente WEB para el consumo de APIs del servidor.

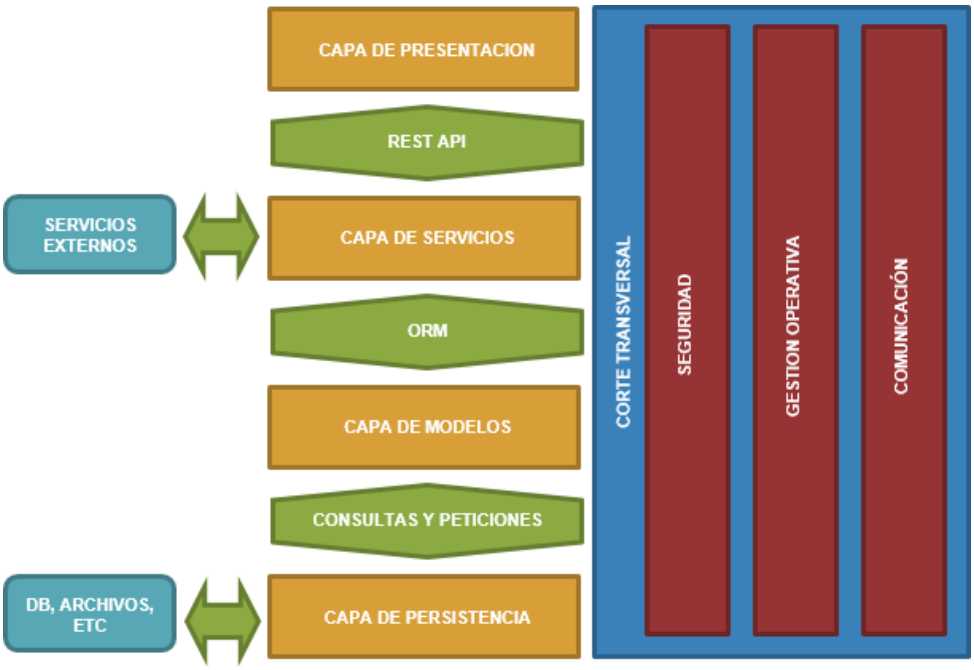

Ilustración 6 Aplicación

Fuente Propia

6.6.1 Capa de Presentación

La capa de presentación corresponde a una aplicación bajo el patrón MVC, utilizando Angular como framework para el lenguaje JavaScript.

Vistas: Páginas web HTML vinculadas con tag libraries de Angular que despliegan la interfaz gráfica al usuario.

Controladores: Controladores de Angular, que reciben y procesan las solicitudes del usuario.

Modelos: Objetos reutilizados de la capa de modelo de dominio de Angular.

Componentes: Define la lógica de la aplicación en un componente dentro de una clase. La clase interactúa con la vista a través de una API de propiedades y métodos.

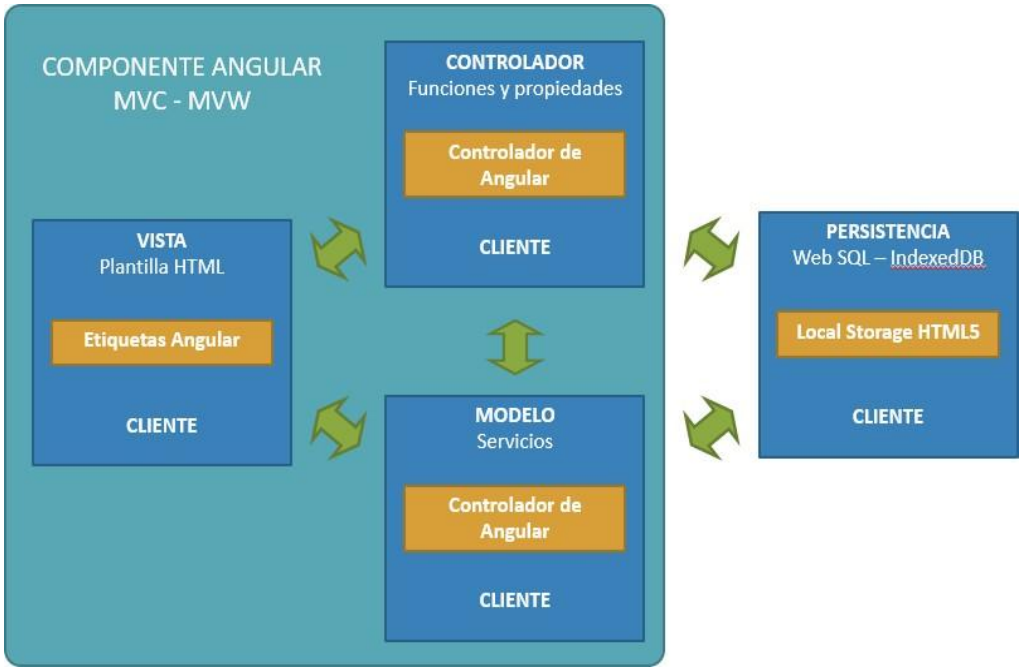

Ilustración 7 Capa de presentación

Fuente Propia

6.6.2 Capa REST API

Representational State Transfer (REST), es un tipo de arquitectura de desarrollo Web que usa el protocolo HTTP para su funcionamiento, en tal sentido, para poder usar REST no necesita de la implementación de ningún otro protocolo adicional sino el mismo que usa la WEB, a diferencia de otros tipos de arquitectura.

En el desarrollo de servicios Web o una aplicación Web y sobre todo en una API REST se dan tres niveles de calidad que se describen en el modelo “Richardson Materita Medel” siendo estos:

- Uso correcto de URls
- Uso correcto de HTTP
- Implementar Hypermedia

Ilustración 8 REST API

Fuente Propia

HTTP que es un protocolo sin estado, cuando es utilizado por REST nos posibilita una mayor escalabilidad, y permite alternar con distintas tecnologías para servir determinados recursos de una misma API, por lo tanto, temas como el almacenamiento de variables de sesión, ya no constituyen una preocupación técnica. (Santos Hernadez & Serrano Parreño, 2017)

6.6.3 Capa de servicios

Esta capa será desarrollada en el lenguaje PHP con el framework Laravel 5.x, que es capaz de soportar todas las capas correspondientes a un API RESTful del lado del servidor.

Laravel cuenta con clases fachada como Reoute, Resorce, Request, que facilitan el uso y manejo de las peticiones HTTP por medio de URLs.

6.6.3.1 Urls y Uris

Las URL (Uniform Resource Locator), al ser un tipo de URI, (Uniform Resource Identifier), esta permite localizar y reconocer de forma única el recurso, además de poder acceder a él se puede compartir su ubicación. Cada página, información en una sección, archivo, cuando se habla de REST, se los nombra como recursos, por lo tanto, es la información a la que se desea acceder, modificar o borrar, independiente mente del formato que tenga.

Existen varias reglas que se deben utilizar para ponerle nombre a la URI de un recurso:

- Los nombres de URI no deben implicar una acción, por lo tanto, debe evitarse usar verbos en ellos.
- Deben ser únicas, no debemos tener más de una URI para identificar un mismo recurso.
- Deben ser independiente de formato.
- Deben mantener una jerarquía lógica.
- Los filtrados de información de un recurso no se hacen en la URI.

Al momento de desarrollar APIs REST los aspectos claves que hay que priorizar son:

6.6.3.2 Métodos HTTP

Para manipular los recursos, HTTP nos proporciona los métodos con los cuales debemos operar siendo estos los siguientes:

- GET: Para consultar y leer recursos
- POST: Para crear recursos

- PUT: Para editar recursos
- DELETE: Para eliminar recursos.
- PATCH: Para editar partes concretas de un recurso.

6.6.3.3 Códigos de Estado

En el desarrollo de una API se debe evitar crear herramientas propias y en su lugar utilizar las que ya han sido creadas, pensadas y testeadas; ya que al crear herramientas propias se cae en uno de los errores más frecuentes en el desarrollo de las APIs, que ocurren principalmente en el manejo de los códigos de error y estado (Santos Hernadez & Serrano Parreño, 2017).

Algunos de estos códigos de estado pueden ser:

- 100: Informan al navegador de algunas acciones que se van a realizar.
- 200: Indican que la petición del navegador se ha recibido, procesado y respondido correctamente.
- 300: Indican que el navegador debe realizar alguna acción adicional para que la petición se complete (como por ejemplo redirigirse a otra página).
- 400: Indican que se ha producido un error cuyo responsable es el navegador:
- 500: Indican que se ha producido un error cuyo responsable es el servidor

En la realización de una operación, es necesario saber si dicha operación ha sido exitosa o ha fallado, de haber ocurrido esto nos interesa saber el por qué ha fallado; ya que HTTP tiene un listado de códigos de estado muy amplio que cubre todos los posibles sucesos, que se deben añadir en las respuestas cuando las operaciones han sido exitosas o se produjo algún error.

Aceptación de tipos de contenido

HTTP permite especificar en qué formato se desea recibir el recurso en orden de preferencia y para ello utiliza el header Accept.

Al cliente se devolverá el header Content-Type, para que se conozca en que formato se devuelve la respuesta. (Santos Hernadez & Serrano Parreño, 2017)

6.6.3.4 Implementar Hypermedia

Considerando que la finalidad de Hypermedia es conectar mediante vínculos las aplicaciones clientes con las APIs y que además se añade información extra al recurso sobre su posible conexión a otros recursos relacionados con él, en los casos que sean necesarios.

6.6.4 Capa ORM

Object-Relational mapping, o lo que es lo mismo, mapeo de objeto-relacional, es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador. El ORM al tener una capa intermedia, abstrae al programador de la base de datos y le centra en el desarrollo de la aplicación.

Otro punto importante es la facilidad de trabajo, un ORM, nos facilita las labores básicas de cualquier acceso a datos, el CRUD (Create, Read, Update y Delete). Realizando todas estas labores a través de un lenguaje de alto nivel orientado a objetos.

Laravel cuenta con el ORM Eloquent el cual es extensamente usado y documentado, siendo uno de los más populares en PHP. (Gonzalez, 2013)

6.6.5 Capa de Modelos

Los modelos son uno de los componentes principales de las aplicaciones desarrolladas bajo el patrón MVC, que tienen la responsabilidad de acceder a los datos, modificarlos, etc. En el patrón además los modelos mantienen lo que se llama la lógica de negocio, que son las reglas que deben cumplirse para trabajar con los datos.

Por tanto, el tipo de acciones que le vamos a solicitar a un modelo es, por ejemplo, obtener datos, insertarlos, modificarlos, etc. En las operaciones que alteren los datos se tendrá que realizar la validación de los mismos, para asegurarnos que tienen la forma necesaria antes de ser guardados.

Un modelo funciona con datos que pueden ser tomados de varias fuentes. Generalmente será la base de datos, pero podría ser un API, Servicio web, sistema de archivos, etc.

En este apartado Laravel integra Doctrine, un paquete de PHP especializado en la interacción con bases de datos. (Ruiz Ruso & Alvarez, 2015)

6.6.6 Capa de consultas y peticiones

Una capa de consulta es una capa o tabla independiente que se define mediante una consulta SQL. Las capas de consulta permiten almacenar, modificar o eliminar información en una base de datos para integrarla fácilmente.

Cuando se trabaja con un ORM, se crean capas de consulta al definir una consulta SQL por medio de un modelo o query builder. A continuación, la consulta se ejecuta en las tablas y vistas de la base de datos y el conjunto de resultados se agrega al mapa como una capa o una tabla independiente (según la consulta).

La consulta SQL que define una capa de consulta es una sentencia SQL que se ejecuta dentro de la base de datos cada vez que la capa se muestra o se utiliza en el mapa. Esto permite que esté visible la última información sin realizar una copia o instantánea de los datos. Esto resulta especialmente útil cuando se trabaja con información dinámica que cambia con frecuencia. No obstante, hay muchos casos en los que partes de las sentencias SQL no se conocen de antemano. Los parámetros de las capas de consultas pueden ayudar a hacer que los componentes de las sentencias SQL sean dinámicos.

Las capas de consulta permiten que el sistema integre datos de otras fuentes como APIs, servicios, sistemas de archivos, etc y también de sistemas de administración de bases de datos. Por lo tanto, las capas de consulta se pueden integrar rápidamente a la información independientemente de dónde y cómo se almacena esa información

Por lo general esta capa es manejada por el ORM, pero no necesariamente, eso depende del tipo de petición y recurso que se requiera (ArcGis Pro, s.f.).

6.6.7 Capa de persistencia

6.6.7.1 Asociación Objeto-Relacional

La asociación objeto-relacional que también es conocido por su nombre en inglés, Object-Relational Mapping, o sus siglas O/RM, ORM, y O/R mapping, es una técnica de programación que convierte datos entre el sistema de tipos haciendo uso del lenguaje de programación encaminado a objetos y el utilizado en una base de datos relacional (MADEJA, s.f.). En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

Un objeto está compuesto de propiedades y métodos. Como las propiedades representan a la parte estática de ese objeto, son las partes que se dotan de persistencia. Cada propiedad puede ser simple o compleja.

Por simple, se entiende que tiene algún tipo de datos nativos como, por ejemplo: entero, coma flotante o cadena de caracteres.

Por complejo se entiende algún tipo definido por el usuario, ya sean objetos o estructuras.

Por relación se entiende asociación, herencia o agregación. Para dotar de persistencia las relaciones, se usan transacciones, ya que los cambios pueden incluir varias tablas. (MADEJA, s.f.)

Para vincular las relaciones, se usan los identificadores de objetos (OID). Estos OID se agregan como una columna más en la tabla donde se quiere establecer la relación. Dicha columna es una clave foránea a la tabla con la que se está relacionada. Así, queda asignada la relación. Recordar que las relaciones en el modelo relacional son siempre bidireccionales. (MADEJA, s.f.)

6.6.7.2 Manejo de la cache

En la mayoría de las aplicaciones, se aplica la regla del 80-20 en cuanto al acceso a datos, el 80% de accesos de lectura accede al 20% de los datos de la aplicación. Esto significa que hay un conjunto de datos dinámicos que son relevantes a todos los usuarios del sistema, y por lo tanto accedido con más frecuencia. Las aplicaciones de sincronización de caché normalmente necesitan escalarse para manejar grandes cargas transaccionales. Así, múltiples instancias se pueden procesar simultáneamente.

Es un problema serio para el acceso a datos desde la aplicación, especialmente cuando los datos involucrados necesitan actualizarse dinámicamente a través de esas instancias. Para asegurar la integridad de datos, la base de datos comúnmente juega el rol de árbitro para todos los datos de la aplicación. Es un rol muy importante dado que los datos representan la parte de valor más significativa de una organización. Desafortunadamente, este rol no está fácilmente distribuido sin introducir problemas importantes, especialmente en un entorno transaccional.

Es común para la base de datos usar replicación para lograr datos sincronizados, pero comúnmente ofrece una copia offline del estado de los datos más que una instancia secundaria activa. Es posible usar bases de datos que puedan soportar múltiples instancias activas, pero se pueden volver caras en cuanto a mantenimiento y escalabilidad, debido a que introducen el bloqueo de objetos y la latencia de distribución. La mayoría de los sistemas usan una única base de datos activa, con múltiples servidores conectados directamente a ella, soportando un número variable de clientes.

En esta arquitectura, la carga en la base de datos se incrementará linealmente con el número de instancias de la aplicación en uso, a menos que se emplee alguna caché. Pero implementar un mecanismo de caché en esta arquitectura puede traer muchos problemas, incluso corrupción en los datos, porque la caché en el servidor 1 no conocerá los cambios en el servidor 2 (MADEJA, s.f.).

6.6.7.3 Concurrencia de usuarios

La capa de persistencia debe permitir que múltiples usuarios trabajen en la misma base de datos y proteger los datos de ser escritos erróneamente. También es importante minimizar las restricciones en su capacidad concurrente para ver y acceder.

La integridad de datos es un riesgo cuando dos sesiones trabajan sobre la misma tabla: la pérdida de alguna actualización está asegurada. También se puede dar el caso, cuando una sesión está leyendo los datos y la otra los está editando: una lectura inconsistente es muy probable.

Hay dos técnicas principales para el problema: bloqueo pesimista y bloqueo optimista. Con el primero, se bloquea todo acceso desde que el usuario empieza a cambiar los datos hasta que se hace COMMIT en la transacción. Mientras que, en el optimista, el bloqueo se aplica cuando los datos son aplicados y se van verificando mientras los datos son escritos.

En la mayoría de los casos el bloqueo optimista es suficiente, controlando los lost-updates. Para casos concretos, por ejemplo, en los que se necesite generar algo como un número de factura sin que queden huecos entre uno y otro se podría usar el bloqueo pesimista. (MADEJA, s.f.)

6.6.8 Capa transversal de seguridad

En esta área se incluyen conceptos y recomendaciones para la seguridad en los procesos de desarrollo de aplicaciones informáticas, teniendo en cuenta la Ley Orgánica de Protección de Datos.

Además de seguir estas consideraciones en los desarrollos de las aplicaciones, es muy importante prestar atención en los casos en que se utilicen componentes de terceros. En este sentido, es recomendable disponer de un procedimiento documentado para la adquisición de componentes cuya evaluación se haya realizado conforme a normas colombianas o internacionales (p.ej: cumple la ISO/IEC 15408). Hay que tener en cuenta que la inclusión de un componente con vulnerabilidades en un sistema producirá probablemente un detrimento de la seguridad del mismo.

Una parte importante de los problemas de seguridad de los sistemas de información tiene su origen en defectos o carencias en las aplicaciones que los integran, lo cual eleva enormemente el riesgo de sufrir un incidente.

En una aplicación web, dividimos la seguridad en:

Disponibilidad: Asegurar que las entidades o procesos autorizados tienen acceso a los activos cuando lo requieren.

Autenticidad: Verificamos quien es el usuario. Generalmente, esto se hace pidiéndole un nombre de usuario y un password, es decir, se verifica que una entidad es quien dice ser o bien que garantiza la fuente de la que proceden los datos.

Integridad: Asegurar que los datos que se envían entre el cliente y el servidor no hayan sido alterados de forma no autorizada. Para esto, se utilizan generalmente algoritmos de encriptación.

Confidencialidad: Asegurar que la información ni se pone a disposición, ni se revela a individuos, entidades o procesos no autorizados. Esto puede hacerse con algoritmos de encriptación y con certificados digitales.

Trazabilidad: Verificar que las actuaciones de una entidad pueden ser imputadas exclusivamente a dicha entidad.

El análisis de las vulnerabilidades potenciales es un objetivo básico para el incremento de la seguridad en las aplicaciones web, que en ocasiones es subestimado como factor de riesgo crítico. El mantener parámetros que no son verificados, roles sin controlar, desbordamientos que se producen en la memoria son algunas de las situaciones que pueden provocar brechas de seguridad en las aplicaciones. (MADEJA, s.f.)

Objetivos de la capa de seguridad

- Asegurar la autenticación de los usuarios Gestionar los permisos de acceso a los recursos
- Evitar la corrupción de los datos enviados entre cliente y servidor
- Asegurar la no visibilidad de la información por terceros en las comunicaciones

6.6.9 Capa transversal de gestión operativa

Si utilizamos la abstracción llegaremos a la conclusión de que una operación, por ejemplo, de compra es independiente del canal, es decir, una compra se registra en nuestro sistema con los mismos datos y desencadenando los mismos procesos derivados (facturación, entrega, etc) independientemente de que si el cliente la ha realizado desde su computador o desde su teléfono; o si realiza una reserva directamente desde una página web o desde el sistema interno por medio de un recepcionista.

Aunque es muy de sentido común y parece obvio, en muchas organizaciones no existe un punto de ejecución único, sino que se duplica la lógica por cada canal, lo que aumenta considerablemente el esfuerzo de mantenimiento y en la misma proporción, el número de errores.

La Arquitectura Orientada a Servicios es idónea para la gestión operativa. Si recordamos, el principio básico de SOA es la reutilización y se basa es exponer servicios del negocio reutilizables.

En la siguiente imagen se intenta ofrecer a muy alto nivel una arquitectura base para la implementación con éxito de la gestión operativa:

Ilustración 9 Capa transversal

Fuente Propia

Como se puede ver, los servicios de nuestro Backend se exponen hacia los diferentes canales a través de una serie de APIS. El punto final de ejecución es único en cada módulo.

REST es la herramienta perfecta para exponer nuestros servicios y que puedan ser consumidos sin problemas.

Si se detallara más la arquitectura propuesta se podría establecer una capa "adaptadora" de los mensajes recibidos de los canales a los servicios y viceversa ya que sí es cierto que o bien, cierta información se omite para algunos dispositivos o módulos, o bien queremos tener una serie de campos de control de canales.

Otra subcapa que será implementada de sería una capa de auditoria que clasificase las peticiones dependiendo de su origen

6.6.10 Capa transversal de comunicación

Se debe diseñar la interfaz del servicio Web intentando minimizar el tráfico de la red. Para ello es necesario agrupar la funcionalidad del servicio de manera que el número de comunicaciones necesarias sea el menor posible.

Se deben construir servicios generales, es decir, que agrupen un gran número de funcionalidades y devuelvan una gran cantidad de información, pero se debe ser precavido ya

que con un gran tamaño de funcionalidades afecta el tiempo de ejecución para interpretarlos y se puede necesitar mayor cantidad de tiempo. Con ello se reduce la sobrecarga de la red y se mejora el tiempo de respuesta del servicio.

Hay que evaluar a qué tipo de clientes está orientado el servicio. La seguridad tiene un impacto en el desempeño. No todo el tráfico REST necesita estar protegido, ya que el coste en rendimiento de una seguridad de punta a punta (por ejemplo, con WS-Security) es en muchos casos mucho mayor a implementar un mecanismo de seguridad a nivel de transporte como SSL. Definir políticas de seguridad en función del tipo de servicio y del entorno del mismo.

La utilización de caché es una forma de mejorar el rendimiento de los sistemas que hacen uso intensivo del procesador. Sin embargo, esto solo aplica para los servicios de solo lectura.

Las conexiones persistentes son buenas para el rendimiento en casos donde existan gran cantidad de mensajes con poco contenido. Para mensajes más grandes, la diferencia no es tan considerable.

Las conexiones de flujo continuo (streaming) son buenas para el rendimiento en casos donde existen mensajes con gran cantidad de contenido. La codificación HTTP "chunked" es un tipo de flujo continúo soportado por HTTP/1.1.

En ocasiones, efectuar una codificación binaria de elemento contenidos en el mensaje puede resultar beneficiosa. En el caso de tener una capa de transporte lenta o de que el procesamiento sea complejo, es indispensable disponer de un modelo de mensajería asíncrona. (MADEJA, s.f.)

6.7 Arquitectura de la Información

La Arquitectura de la Información se puede separar en cuatro componentes: la organización, navegación, rotulado y sistemas de búsqueda. Cada uno de estos componentes cumple un papel fundamental en la arquitectura general de la aplicación, y la deficiencia de uno de ellos puede ocasionar grandes problemas de usabilidad.

Organización

Existen diferentes esquemas de organización, en las cuales se puede dividir en exactas o subjetivas y ambiguas. La organización exacta se refiere a aquellas que tienen una sola interpretación, como pueden ser las que se organizan en forma alfabética (diccionarios, directorios y listados ordenados), cronológicas (revistas, periódicos, publicaciones), geográficas (agencias y sucursales, portales organizados geográficamente). Mientras la organización subjetiva se basa en diversos criterios, como son las temáticas (portales horizontales, tiendas organizadas por rubros), funcionales (intranets corporativas), audiencia específica y la metafórica.

Navegación

El sistema de navegación es uno de los temas más importantes en la accesibilidad y usabilidad de la aplicación. Proveer opciones para ir de un lado a otro, poder regresar a la página anterior o ir hacia otras secciones con el menor esfuerzo, puede brindar al usuario cierta placentera comodidad. Existe barra de navegación horizontal, vertical, desplegable, permanentes. La navegación se puede clasificar en globales (acceso a las secciones principales), locales (acceso a las secciones internas) y ad hoc (acceso a secciones relacionadas). Se recomienda presentar información que permita conocer la ubicación exacta del navegante, como opciones de subir o bajar cuando existen textos grandes. En la

navegación externa, se puede apoyar la navegación utilizando tablas de contenido, índices, mapas del sitio o visitas guiadas.

Rotulado

La rotulación es una forma de representación de la información, que describe el contenido de una aplicación. Los sistemas de rotulación pueden ser como enlace, encabezados, como iconos, y además cumple una función fundamental en la indización de documentos.

Sistema de Búsqueda

En algunas aplicaciones la posibilidad de explorar el contenido puede ser un pasatiempo placentero, sin embargo, cuando un se cuenta con más de 50,000 contenidos puede convertirse en una pesadilla. Los sistemas de búsqueda permiten encontrar rápidamente la información, y algunas interfaces permiten realizar opciones de filtrado por secciones o por tipo de documento. En el caso de contenidos dinámicos, es necesario implementar un buscador interno, más aún cuando los robots y arácnidos de indización, no pueden clasificar la información en los grandes motores de búsqueda. (Celso, 2003)

6.8 Arquitectura de la Interfaz

Model-View-Controller (MVC) es un patrón de arquitectura de software cuya principal función es separar en componentes la interfaz de la aplicación, de la lógica de negocio y de los datos de la misma. Además, sobre su función también recae la gestión de las comunicaciones entre estos componentes, así como gestionar los eventos que desencadenarán la actividad de los componentes.

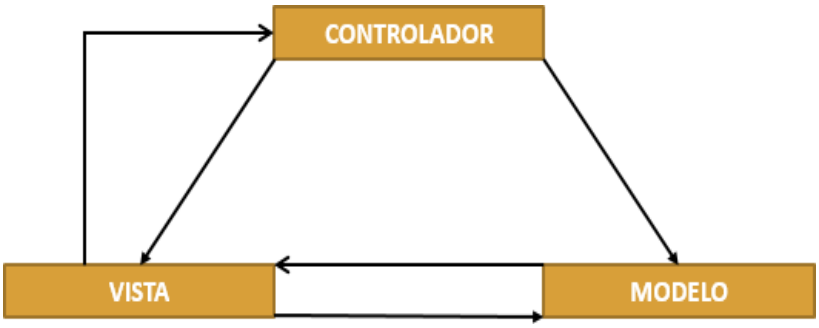

Ilustración 10 Arquitectura de la interfaz

Fuente: Autoría propia

Busca separar conceptos y la reutilización de código mediante tres componentes que son:

Modelo: es la representación de la información del sistema. Gestiona el acceso a esa información.

Controlador: responde a los eventos del usuario e invoca peticiones al modelo. También determina la vista a mostrar (puede tener varias) y se comunica con ella si la información se tiene que presentar de una determinada forma.

Vista: se encarga de la lógica de presentación la información del modelo adecuándola a la visión del usuario permitiéndole interactuar con su interfaz gráfica.

Hay cierto acoplamiento entre la Vista y el Modelo.

La interacción entre el usuario y los distintos componentes del sistema se realiza de la siguiente manera:

- El usuario interactúa con la interfaz gráfica y solicita una acción mediante una petición. Esta acción corresponde a una llamada a un método del controlador
- El controlador gestiona la petición.

- El controlador accede al modelo para modificar cierta información en función de la petición.
- El controlador delega a la vista la acción de desplegar la interfaz de usuario reflejando los cambios del modelo. De esta forma la Vista accede al modelo. El controlador no pasa datos del modelo a la vista.

La interfaz de usuario queda activa esperando nuevas acciones (Universidad de Alicante, s.f.).

6.9 Arquitectura Tecnológica

El avance de las tecnologías y las nuevas tendencias globales exigen un cambio en las aplicaciones. Los ámbitos son impulsados por las nuevas necesidades de los usuarios, y estas se centran principalmente en facilidades de uso, colaboración, entre otras. Las aplicaciones de hoy en día deben cumplir con estándares web 2.0. Que permitan la usabilidad, tener un diseño centrado en el usuario y ambientes de colaboración, entre otras características. Teniendo en cuenta a PHP con el framework Laravel como tecnología de implementación, se puede identificar que a la fecha esta herramienta en versión 5.4 está bien posicionada en el mercado.

6.9.1 Elección del lenguaje backend

Se tienen en cuenta las tendencias del mercado según los índices de Google Trends teniendo en cuenta los lenguajes de programación más aceptados en la CORHUILA según los lenguajes adoptados para las aplicaciones existentes en la organización

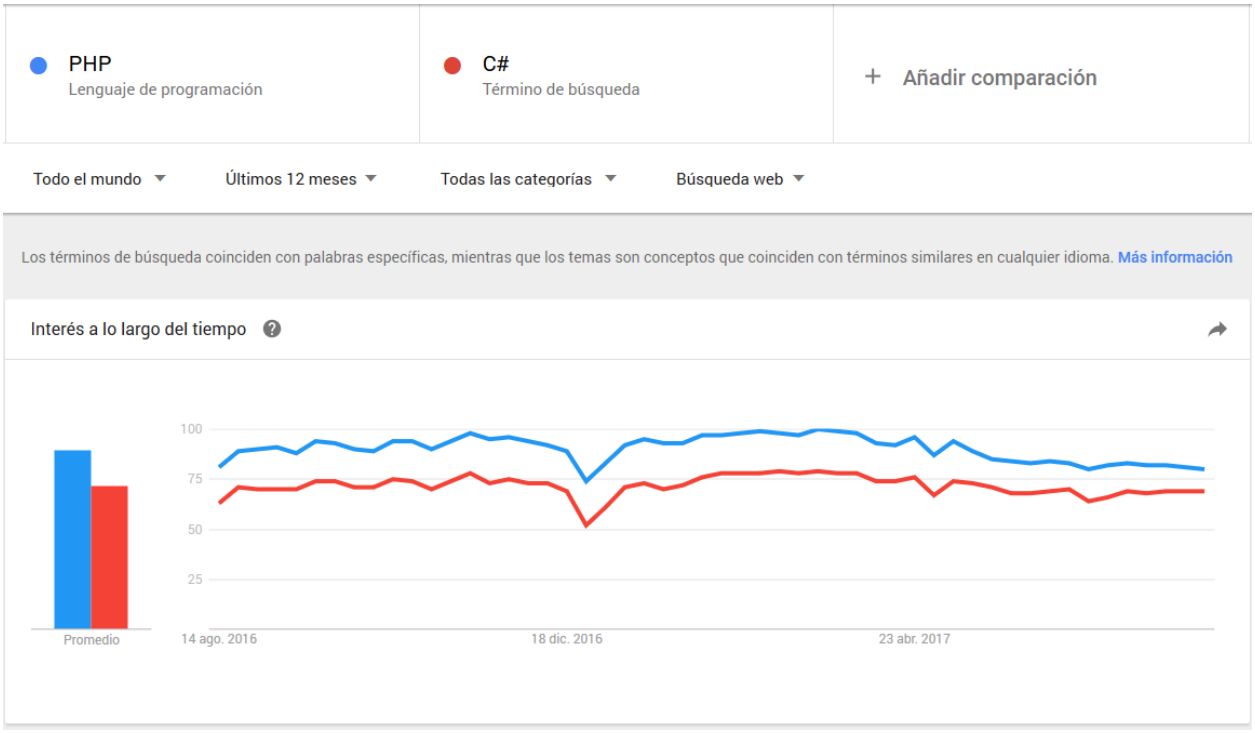

Ilustración 11Eleccion del lenguaje backend

Fuente: https://trends.google.es/trends/explore?geo=ES&q=%2Fm%2F060kv,C%23

El índice de lenguaje de programación de PYPL se crea analizando la frecuencia con la que se buscan los tutoriales del lenguaje en Google.

Cuanto más se busca un tutorial del lenguaje, más popular se supone que es el lenguaje. Es un indicador adelantado. Los datos en bruto provienen de Google Trends.

Si se cree en la sabiduría colectiva, el índice PYPL de popularidad del lenguaje de programación puede ayudar a decidir qué lenguaje estudiar o cuál usar en un nuevo proyecto de software.

Worldwide, Jun 2017 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.7 %	-1.2 %
2		Python	16.1 %	+3.8 %
3		PHP	9.3 %	-0.9 %
4		C#	8.2 %	-0.6 %
5		Javascript	7.9 %	+0.5 %
6		C++	6.8 %	-0.3 %
7		C	6.6 %	-0.1 %
8		Objective-C	3.7 %	-1.0 %
9		R	3.6 %	+0.4 %
10		Swift	2.8 %	-0.2 %

Ilustración 12PYPL

Fuente: http://pypl.github.io/PYPL.html

Teniendo en cuenta la popularidad de los lenguajes de programación de lado del servidor se toma la decisión de desarrollarla con el lenguaje PHP con Laravel como framework teniendo en cuenta las tendencias en el uso de los frameworks PHP más populares del mercado según el índice de Google Trends.

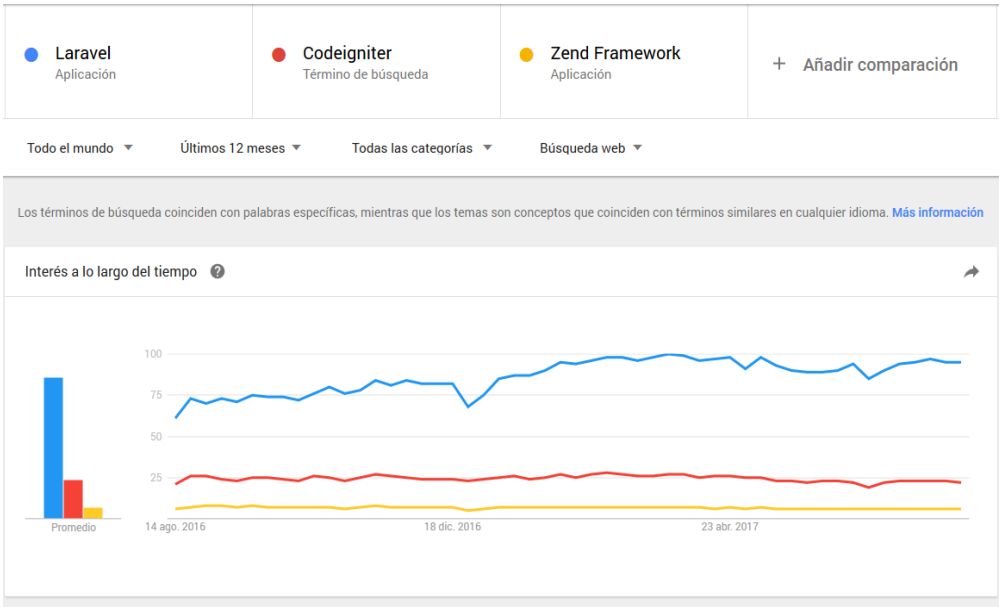

Ilustración 13 Framework Laravel

Fuente: https://trends.google.es/trends/explore?geo=ES&q=%2Fm%2F0jwy148,Codeigniter,%2Fm%2F0cdvjh

6.9.2 Elección del lenguaje frontend

Ya que se opta por la implementación de una aplicación web el lenguaje de programación del lado del cliente será Javascript, en este aspecto la toma de decisión recae en el framework a utilizar; el framework debe ser capaz de implementar de manera nativa las necesidades más recurrentes en el consumo de servicios RESTful, del mismo modo que para el backend se realizó un estudio de tendencias para elegirlo, teniendo en cuenta los dos más populares a la fecha.

Es importante resaltar que el framework de desarrollo de Javascript, debe permitir que la aplicación sea compatible con diferentes tipos y versiones de navegadores y así mismo, que integre otras tecnologías de tendencia para web como HTML5 o CSS3, preprocesadores de CSS como SASS o LESS y funcionalidades de fácil creación de interfaces responsive. Algunas de las funcionalidades son:

- Diseño gráfico personalizable.
- Diseño responsivo.
- Capacidad de personalización en tiempo de ejecución.
- Administración del portal.
- Seguridad reconfigurada del sitio.
- Conjunto de servicios Web 2.0.

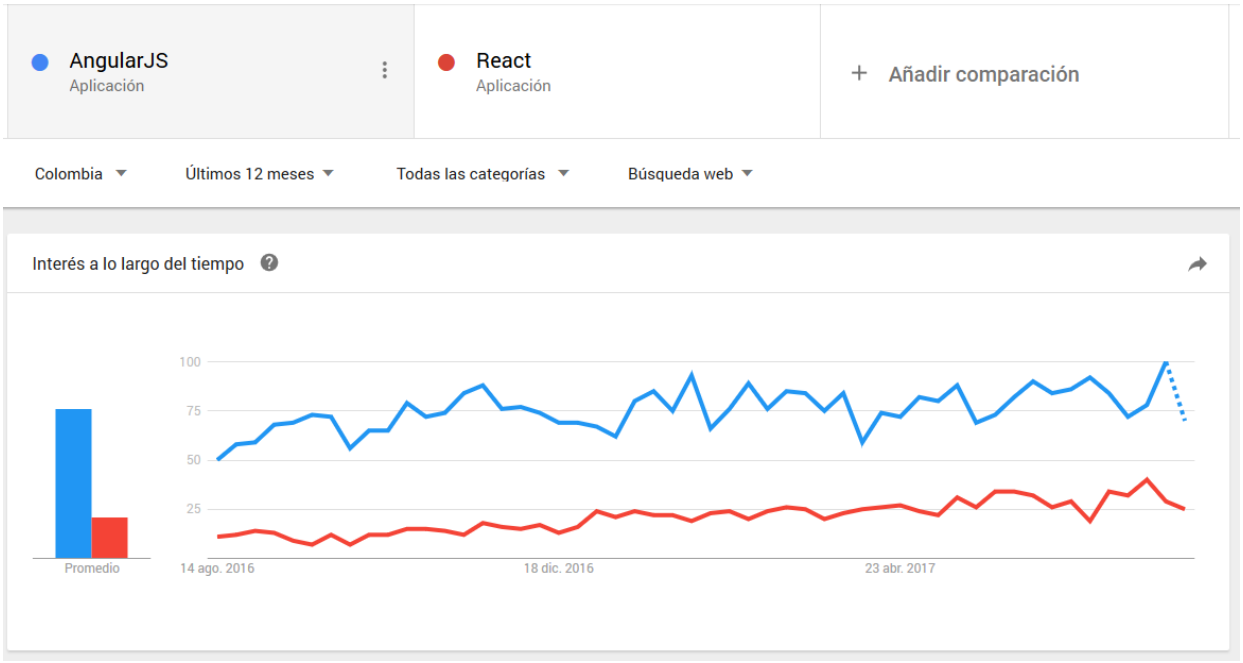

Ilustración 14 Framework Angular

Fuente: https://trends.google.es/trends/explore?geo=ES&q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9

6.9.3 Plataforma

La plataforma de software base adoptada por la entidad está basada en componentes Microsoft, por lo que los componentes de las capas de paralización, acceso, aplicaciones, interoperabilidad y persistencia de datos deben ser compatibles en forma nativa con dichos componentes.

Los componentes del software de la aplicación web deben desplegarse en el servidor de la aplicación. Los componentes deben construirse de forma genérica para que puedan ejecutarse en cualquier servidor, ej.: Los componentes deben ser portátiles. El servidor de la aplicación proporciona el contenedor web donde se ejecuta la aplicación. Por lo tanto, la aplicación se debe ejecutar en cualquier servidor de aplicación que tenga un contenedor web.

El servidor de la aplicación debe tener características que soporten portales. El servidor de la aplicación soporta la aplicación web con características adicionales para concurrencia,

balance de carga, agrupación de conexiones, escalabilidad. Estos son los requisitos generales de cualquier aplicación web. (Salud, 2017)

Apache 2.4 es el servidor de aplicación que se sugiere implementar, ya que Apache es un proyecto de código abierto y de uso gratuito, multiplataforma (hay versiones para todos los sistemas operativos más importantes), muy robusto y que se destaca por su seguridad y rendimiento. (Digital Learning, 2012)

6.9.4 Alojamiento el sistema

Las herramientas tecnológicas para garantizar una optimización y flexibilidad en la administración de la infraestructura de servidores.

En este grupo de componentes se deben integrar:

- Administración dinámica de ambientes de sistemas operativos
- Monitoreo al desempeño y capacidad de los equipos y configuración de las máquinas

Esta es la capa de hardware básica para ejecutar la aplicación.

Los sistemas operativos seleccionados fueron Ubuntu Server 16.04 TS.

Considerando el alto costo de los componentes, la virtualización puede traer efectividad en los costos. La virtualización puede reducir los costos en instalaciones, energía, enfriamiento y hardware. Puede simplificar la administración y mantenimiento y promete un perfil TI más verde.

La virtualización permite ejecutar múltiples máquinas virtuales en una física y compartir los recursos de ese computador físico en varios ambientes. Se pueden ejecutar diferentes

máquinas virtuales (VM) en diferentes sistemas operativos y varias aplicaciones en un mismo computador físico. El propósito de la virtualización, como componente intermedio y como estrategia del sistema integrado de información, es permitir una administración flexible de la infraestructura y mejorar la rentabilidad de su adquisición. Microsoft Hyper-V es el componente de máquina virtual de Microsoft que adoptó la Superintendencia Nacional de Salud.

6.10 Arquitectura de seguridad y privacidad

6.10.1 Autenticación

Se analizan las vulnerabilidades que pueden darse en las aplicaciones a la hora de identificar sus usuarios y los permisos que estos poseen. Recogiendo una serie de recomendaciones para el desarrollo de aplicaciones, que, tenidas en cuenta, ayuden a mitigar los riesgos de producirse situaciones como el escalado de privilegios o la suplantación de identidad.

La debilidad de seguridad más común en aplicaciones web es la falta de validación apropiada de las entradas del cliente o del entorno. Teniendo en cuenta una serie de indicaciones y consejos a la hora de codificar nuestras aplicaciones podremos evitar problemas como la inyección de código SQL. (MADEJA, s.f.)

La autenticación de Laravel proporciona de forma sencilla el registro e inicio de sesión además el cierre de sesión y el restablecimiento de la contraseña del usuario, de forma rápida y sencilla.

Agregar autenticación a una aplicación Angular y Laravel no es lo más directo, especialmente si tomamos el enfoque de crear aplicaciones front-end y backend independientes y conectarlas con una API servida por Laravel.

Laravel viene con la autenticación fácil de usar, pero es basada en sesión y por lo tanto es más útil para aplicaciones tradicionales de cliente-servidor.

Para aplicaciones de una sola página (SPA) que dependen de una API, una forma mejor de manejar la autenticación es con JSON Web Tokens, o JWTs.

En pocas palabras, un JWT es un objeto JSON con tres partes distintas que se utilizan juntas para transmitir información entre dos partes. Los JWT consisten en un encabezado, una carga útil y una firma que están codificadas.

La autenticación tradicional requiere que el servidor almacene la información de autenticación del usuario que se comprueba cada vez que el usuario hace una solicitud. Este método crea desafíos cuando la aplicación crece y necesita escalar, especialmente si se distribuye a través de varios servidores diferentes. También se convierte en problemático cuando queremos utilizar nuestra API para otros fines, como para aplicaciones móviles

6.10.2 Autorización

Laravel puede controlar el acceso a ciertas secciones del sitio o servicios, o activar o desactivar partes particulares de una página para los no administradores, o asegurarse de que alguien solo puede editar sus propios contactos, es necesario que la aplicación cuente con una herramienta como BeatSwitch Lock o Hand-roll, que sería algo llamado ACL: Access Control Lists, o básicamente la habilidad de definir la capacidad de un usuario para hacer y ver ciertas cosas basadas en los atributos de su registro de usuario (roles y permisos).

6.10.3 Despliegue

Despliegue Continuo es un concepto salido de las mejores prácticas de administración de la configuración, pruebas automatizadas, integración continua, gestión de datos y gestión de versiones.

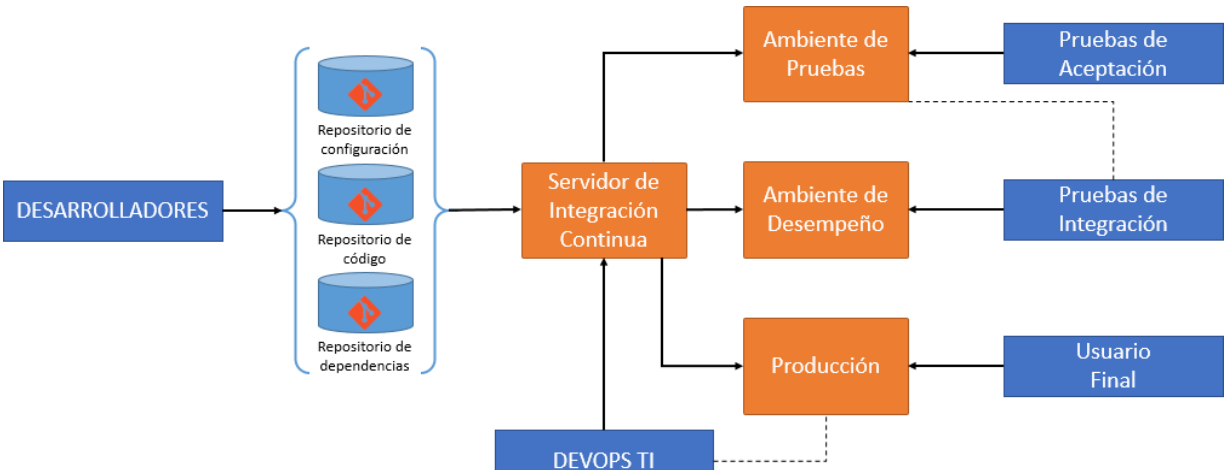

Ilustración 15 Despliegue

Fuente Propia

La infraestructura consiste de hardware y herramientas o frameworks que agilizarán el ciclo de vida de desarrollo de software (develop-test-release):

Desarrollo es el ambiente local donde trabajarán los programadores y se generarán la codificación y configuración de la aplicación, así como los documentos, manuales y demás artefactos producidos durante el desarrollo. Este ambiente contiene a la aplicación funcionando de manera local. Una vez que se haya construido una historia de usuario o caso de uso, el código, configuraciones y documentos asociados serán subidos a los repositorios de código y configuración para su correcta administración de versiones. Éstos pueden reposar en un solo repositorio, pero deben localizarse en una máquina independiente de las demás, de preferencia en una locación segura. Por otro lado, en muchos desarrollos existen librerías

comunes a varios proyectos; en otros casos puede tenerse un “proyecto base” que tenga baja variabilidad y sólo cambie muy de vez en cuando. Para no compilar nuevamente todo cada vez que se realiza un despliegue es necesario contar con un repositorio de dependencias que facilite su gestión.

Como parte de las buenas prácticas de desarrollo, es necesario contar con un servidor de integración continua, que de forma periódica descarga la última versión de la aplicación y ejecuta automáticamente una serie de pruebas unitarias para conocer la “salud general” del código encontrado en el repositorio de versiones. Si estas pruebas unitarias se ejecutan correctamente, se puede utilizar el mismo servidor de integración para generar un despliegue que será depositado en el ambiente de pruebas: un ambiente lo más parecido a producción donde el equipo de calidad, así como el usuario o gerente de producto podrán verificar la funcionalidad de la aplicación.

Más tarde es necesario realizar pruebas de capacidad o performance. Esto se logra en un ambiente de desempeño, que es una copia fiel del ambiente de producción con la capacidad de ser estresado por el equipo de calidad. La principal diferencia entre éste y el ambiente de pruebas es la escalabilidad: el ambiente de desempeño ya tiene configuradas funcionalidades como balanceo de carga o migración de sesiones de usuario.

Finalmente, previa autorización, se sube la aplicación y su configuración a un ambiente de producción.

Para desplegar la nueva versión o regresar a una anterior, debe existir un proceso automatizado por medio de versionamiento que pueda ser ejecutado desde el servidor de integración continua. (EVERAC, 2011)

7 Proceso de desarrollo

Los procesos de desarrollo con enfoques agiles permiten construir aplicaciones en ciclos cortos establecidos dentro del proceso, esto con el propósito de finalizar los proyectos en el menor tiempo posible y orientados hacia la satisfacción del cliente, en procura del éxito en los proyectos de desarrollo de software con los acuerdos contractuales establecidos. Se destacan como métodos y procesos ágiles: Scrum, XP (eXtreme Programming), TDD (Test Driven Development) y DAD (Disciplined Agile Delivery).

Para la elaboración del presente trabajo se utiliza como base referencial el proceso de desarrollo DAD, que según (Ambler & Lines, 2012) es un enfoque híbrido, que extiende Scrum con estrategias probadas del modelado ágil (AM), programación extrema (XP) y el proceso unificado (UP); DAD incluye prácticas que no se encuentran incluidas en el SCRUM como es el modelado y la documentación, ofreciendo la posibilidad de adaptar estrategias funcionales. Este proceso se identifica como la mejor opción para plantear un proceso de desarrollo ágil sin dejar a un lado la documentación que para el trabajo que se pretende desarrollar es una actividad relevante en la entrega de productos de calidad.

El enfoque del DAD es la entrega de valor para el cliente, sin dejar a un lado los aspectos del ciclo de vida del sistema como los requisitos, arquitectura y todas las actividades de desarrollo que este implica a partir de la idea hasta la entrega del producto final. El proceso de entrega ágil disciplinado (DAD), está caracterizado por:

- Está estructurado bajo un enfoque híbrido, que permite la adaptación de diversas estrategias del ciclo de desarrollo de Software.

- Hace referencia en las personas como elemento importante dentro del proceso de desarrollo.
- Asegura la entrega de soluciones tecnológicas.
- Minimiza riesgos.
- Está orientado al sector empresarial.
- Es escalable permitiendo cumplir con los objetivos trazados al inicio del proyecto del desarrollo.
- No propietario; es de libre acceso.

En el enfoque del proceso disciplinado de entregas ágiles, va más allá que entregar una solución, las opciones que ofrece este marco en el entorno empresarial son:

- Equipos de trabajo agiles: Involucra un recurso humano especializado que trabaja del inicio al fin en todas las actividades que relacionadas con la construcción de Software (Análisis, Diseño, Arquitectura, Pruebas, etc.), haciendo participe a personas fuera de este equipo, como directivos, arquitectos, gestores de datos y de cualquier otra disciplina; esta flexibilidad es con el propósito de que el equipo de trabajo tenga éxito en el proyecto de desarrollo.
- Estrategia coherente para la agilidad de TI: Permite la participación del equipo del departamento de TI, para hacer fácil el desarrollo de las aplicaciones.
- Apoyo a la empresa: elimina procesos que no aportan ningún valor agregado al cliente; anticipando y respondiendo rápidamente a los cambios en el mercado.
- Contexto: Cada organización, persona y cada equipo es única; DAD proporciona la posibilidad de adaptar opciones y que estas evolucionen enfrentándose a situaciones que se presenta en la práctica.

7.1 Tecnologías y Entorno de Desarrollo

Para el desarrollo del software, se hizo la utilización de diversas herramientas que ayudan y facilitan el proceso de desarrollo del sistema

7.1.1 Tecnologías aplicadas al Frontend

7.1.1.1 Javascript

Javascript es un lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario y la realización de automatismos sencillos. En ese contexto podríamos decir que nació como un "lenguaje de scripting" del lado del cliente, sin embargo, hoy Javascript es mucho más. Las necesidades de las aplicaciones web modernas y el HTML5 ha provocado que el uso de Javascript que encontramos hoy haya llegado a unos niveles de complejidad y prestaciones tan grandes como otros lenguajes de primer nivel. (Desarrollo Web, s.f.)

7.1.1.2 Angular

Es una framework que permite desarrollar aplicaciones web en la sección cliente utilizando HTML y JavaScript para que el cliente asuma la mayor parte de la lógica y descargue al servidor con la finalidad de que las aplicaciones ejecutadas a través de Internet sean más rápidas. (Boada Oriols & Gomez Guiterrez, 2019)

7.1.1.3 NodeJS

NodeJS es un código abierto de JavaScript (razón por la cual se incluye el distintivo JS) que está diseñado para generar aplicaciones web de forma altamente optimizada. es un entorno

Javascript del lado del servidor, basado en eventos. Node ejecuta javascript utilizando el motor V8, desarrollado por Google para uso de su navegador Chrome. Aprovechando el motor V8 permite a Node proporciona un entorno de ejecución del lado del servidor que compila y ejecuta javascript a velocidades increíbles. El aumento de velocidad es importante debido a que V8 compila Javascript en código de máquina nativo, en lugar de interpretarlo o ejecutarlo como bytecode. Node es de código abierto, y se ejecuta en Mac OS X, Windows y Linux. (Net Consulting, 2019)

7.1.1.4 NPM

Node Package Manager o simplemente npm es un gestor de paquetes, el cual nos facilitará trabajar con Node, ya que gracias a él podremos tener cualquier librería disponible con solo una línea de código, npm nos ayudará a administrar nuestros módulos, distribuir paquetes y agregar dependencias de una manera sencilla. (Guevara Benites, 2017)

7.1.1.5 Gulp

Gulp es una herramienta, en forma de script en NodeJS, que te ayuda a automatizar tareas comunes en el desarrollo de una aplicación, como pueden ser: mover archivos de una carpeta a otra, eliminarlos, minificar código, sincronizar el navegador cuando modificas tu código, validar sintáxis y un largo etcétera. (Torres, 2016)

7.1.1.6 *Bower*

Bower es un complemento ideal para el desarrollo web. Es un sencillo programa que nos sirve para tener al día las dependencias de un proyecto para la web, en lo que respecta al desarrollo frontend, con Javascript o incluso CSS. Se trata de un programa basado en NodeJS que se ejecuta desde la consola y que tiene un sencillo API de comandos útiles para realizar

tareas de mantenimiento y administración de paquetes necesarios para construir un proyecto web, concretamente la parte del lado del cliente. (Alvarez, Desarrollo Web, 2015)

7.1.2 Tecnologías aplicadas al Diseño

7.1.2.1 Material Design

Material Design es un concepto, una filosofía, unas pautas enfocadas al diseño utilizado en Android, pero también en la web y en cualquier plataforma. El encargado de crear Material Design y máximo responsable de diseño en Google es el chileno Matías Duarte. Está basado en objetos materiales. Piezas colocadas en un espacio (lugar) y con un tiempo (movimiento) determinado. Es un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal. (Perez, 2014)

7.1.2.2 Bootstrap

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías. Desde la aparición de Bootstrap 3 el framework se ha vuelto bastante más compatible con desarrollo web responsive. (Fontela, 2015)

7.1.3 Tecnologías aplicadas al Backend

7.1.3.1 PHP

es un lenguaje interpretado del lado del servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos e PHP son embebidos directamente en el código HTML y ejecutados por el servidor web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en forma de código HTML. (Cobo, Gomez, Perez, & Rocha, 2005)

7.1.3.2 Oracle

Oracle es un sistema de gestión de base de datos relacional (Relational Data Base Management System), desarrollado por Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando Sopore de transacciones, Estabilidad, Escalabilidad y Soporte Multiplataforma. (Isaac, 2015)

7.1.3.3 MySQL

MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte, es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle. Actualmente, es la base de datos de código abierto más famosa y utilizada en el mundo entero.

Como él, podemos encontrar otras como la propia Oracle o Microsoft SQL Server. Todas tienen la misma finalidad y se utilizan en el mismo entorno, que no es más que el desarrollo web, y son las que más se utilizan actualmente para dar forma y facilitar la comunicación entre webs y servidores. (Neo Attack, s.f.)

7.1.3.4 Laravel

es un framework de código abierto para aplicaciones y servicios web PHP escrito por Taylor Otwell. (Rees & Laguna, 2013)

7.1.3.5 Composer

Composer es un gestor de dependencias en proyectos, para programación en PHP. Eso quiere decir que nos permite gestionar (declarar, descargar y mantener actualizados) los paquetes de software en los que se basa nuestro proyecto PHP. Se ha convertido en una herramienta de cabecera para cualquier desarrollador en este lenguaje que aprecie su tiempo y el desarrollo ágil. (Alvarez, Desarrollo Web, 2014)

7.1.4 Tecnologías aplicadas al control de versiones

7.1.4.1 GIT

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún software o página que implique código el cual necesitemos hacerlo con un grupo de personas. (Codigo Facilito , s.f.)

7.1.4.2 GitLab

GitLab nació como un sistema de alojamiento de repositorios Git, es decir, un hosting para proyectos gestionados por el sistema de versiones Git. Sin embargo, alrededor de esta herramienta han surgido muchas otras herramientas muy interesantes para programadores y equipos de desarrollo, que envuelven todo el flujo del desarrollo y el despliegue de aplicaciones, test, etc. (Torrado, 2017)

7.2 Patrón de arquitectura del software

El patrón utilizado fue Modelo - Vista - Controlador:

El modelo vista controlador se usa en nuestro sistema para establecer un orden dentro de la aplicación

7.2.1 MVC en Frontend

7.2.1.1 Modelo

En un modelo se tienen los detalles de las clases que usaremos para llamar a la base de datos. El modelo sirve para enlazarse a la base de datos, empaquetar los datos que obtuvo en la consulta y mandárselos al controlador, donde el controlador se los enviara por último a la vista para ser mostrados al usuario.

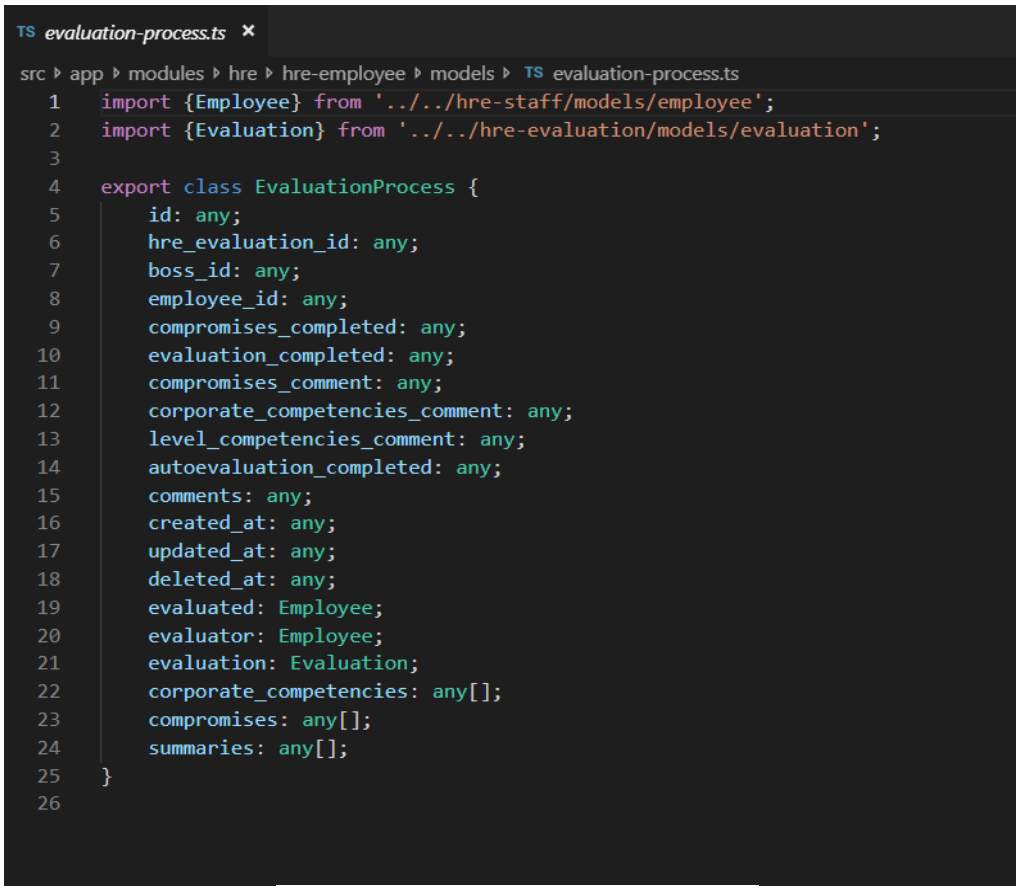

Ilustración 16 Modelo Frontend

Fuente Propia

7.2.1.2 Vista

En la vista manejamos cómo se mostrarán los datos al usuario en un formato adecuado, simple y entendible.

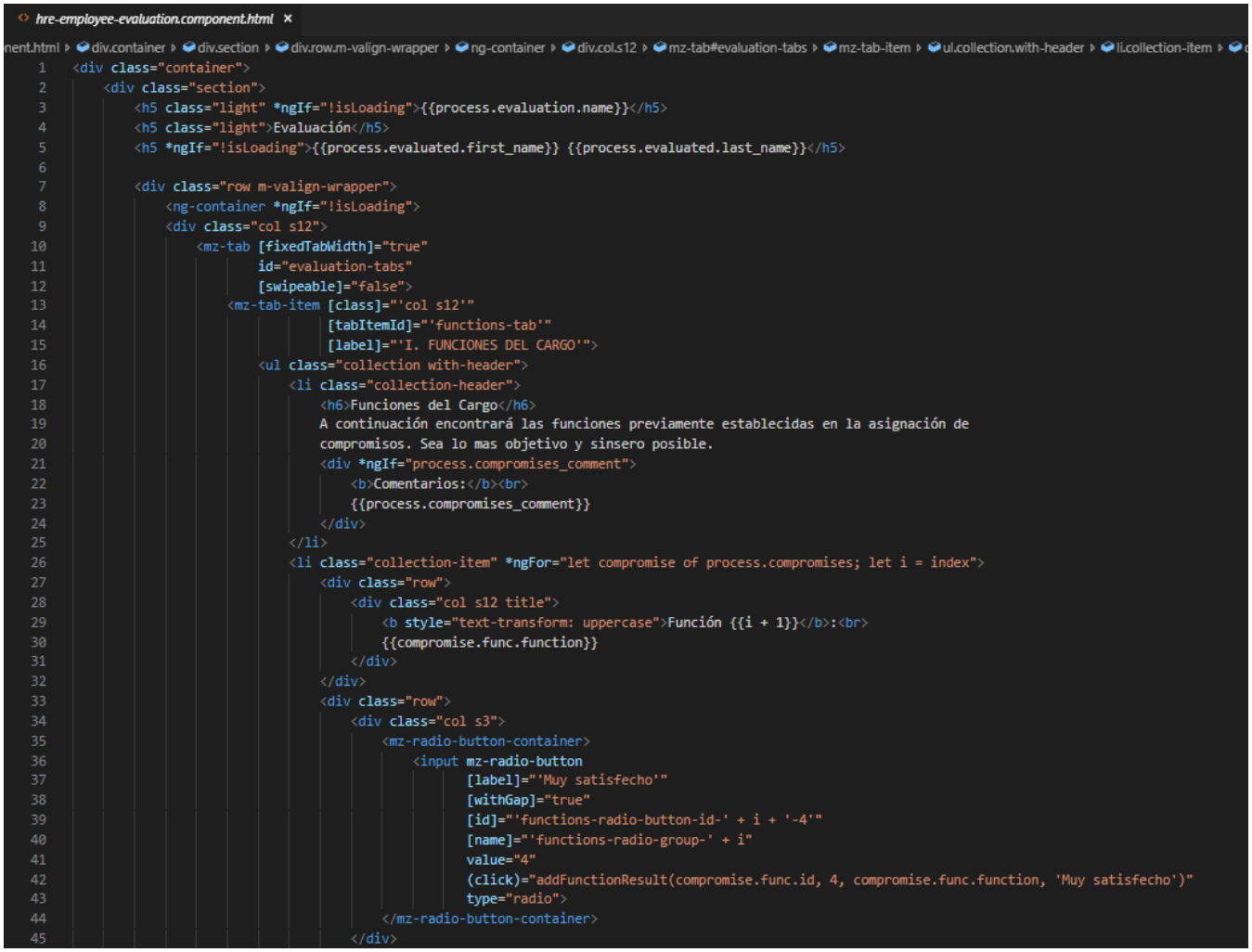

Ilustración 17 Vista Frontend

Fuente Propia

7.2.1.3 Controlador

El controlador es la parte que hace la mayoría del trabajo, En el controlador debemos recibir todas las peticiones que haya enviado el usuario (guardar, modificar, ver una lista, entre otras.) y revisar que las peticiones se filtren por varios requisitos antes de realizar una acción (permisos, parámetros, entre otros.) para un correcto funcionamiento. Cuando se cumpla con los requisitos, el controlador llama al modelo enviando los datos para que se los devuelva y los envía a la vista para que se muestre un resultado aceptable al usuario. Si no cumple, responde un mensaje indicando que las acciones que introdujo el usuario son inválidas.

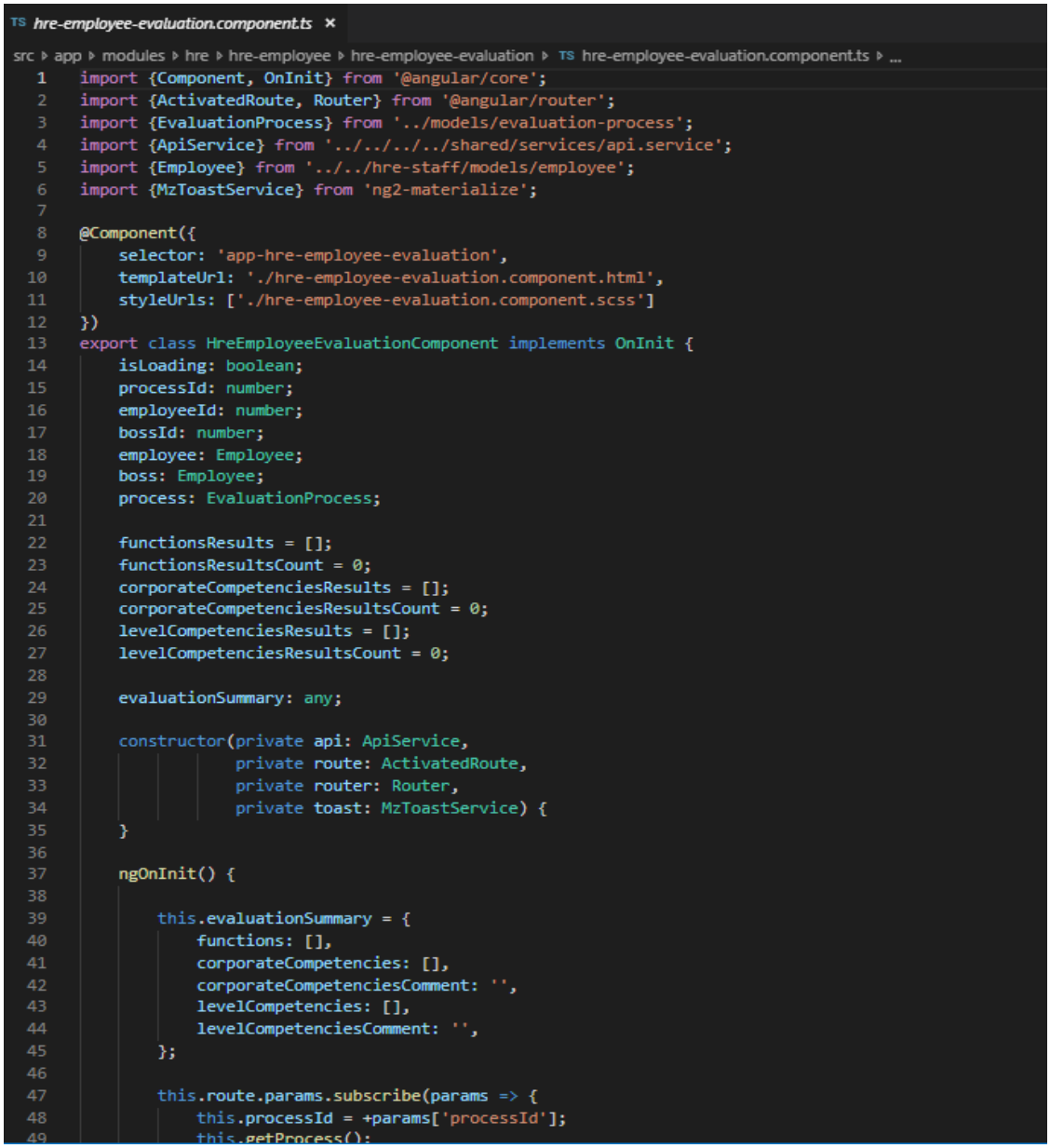

Ilustración 18 Controlador Frontend

Fuente Propia

7.2.2 MVC en Backend

7.2.2.1 Modelo

Los modelos son clases encargadas de trabajar con las consultas de la base de datos, esto quiere decir que por cada tabla tendremos una clase, cada registro será un objeto y las consultas se llamarán a través de métodos de esas clases. A su vez Laravel trabaja con Eloquent que es un ORM que nos facilitará el trabajo de las consultas a través de métodos ya establecidos, estos nos permitirán realizar las tareas más comunes y que más se repiten en una base de datos como insertar, recuperar registros por su id, modificar esos registros, listarlos, eliminarlos, entre otros

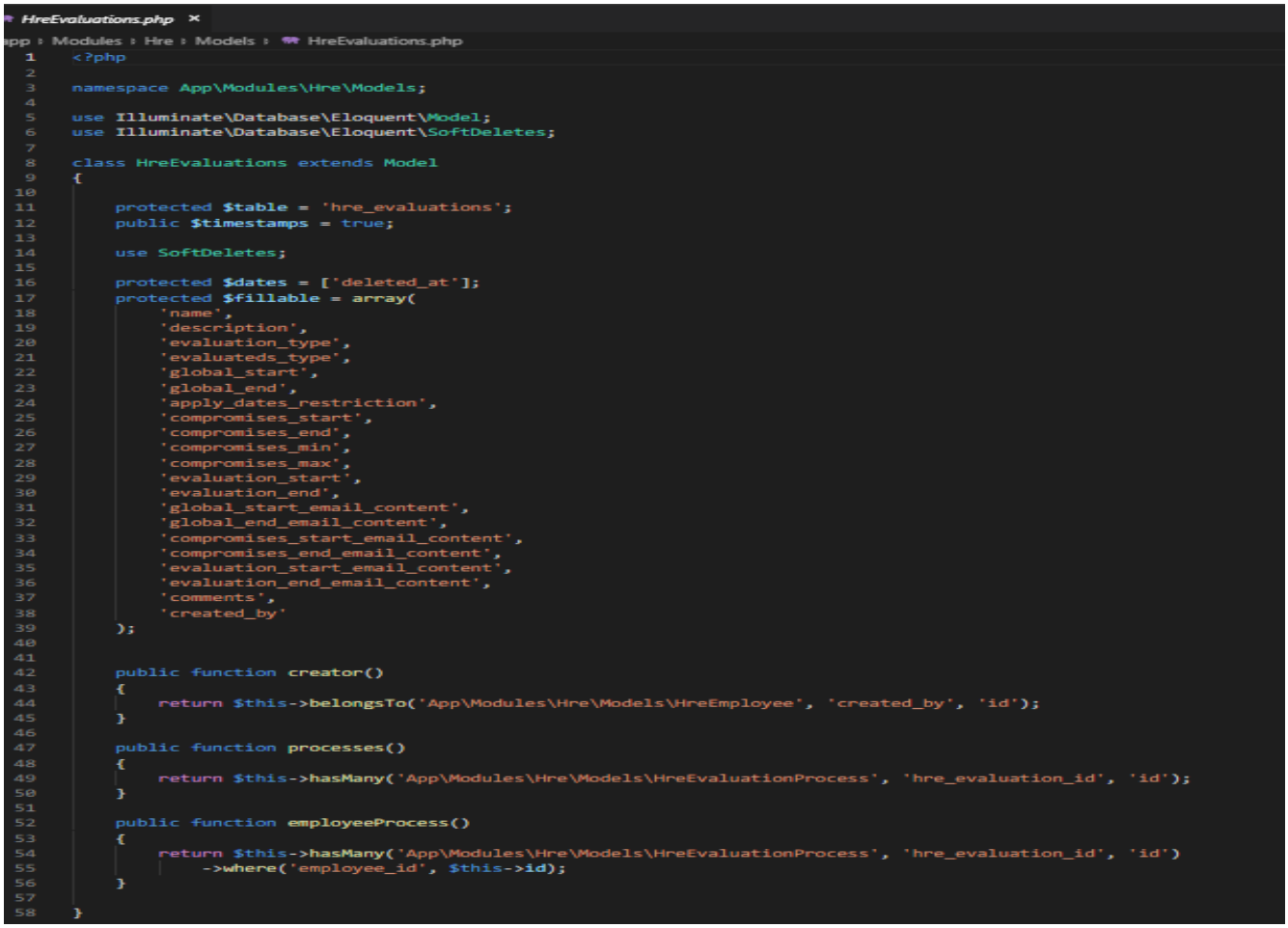

Ilustración 19 Modelo Backend

Fuente Propia

7.2.2.2 Controlador

Al igual que en el Frontend, los controladores se comunican con los modelos para hacer consultas a la base de datos, y con las vistas para devolver una respuesta al cliente. En los controladores creamos métodos que cumplen con acciones como insertar, modificar, eliminar, listar , entre otros

HreEvaluationsController.php

app › Modules › Hre › Http › Controllers › HreEvaluationsController.php

```php

class HreEvaluationsController extends Controller
{

    /**
     * Display a listing of the resource.
     *
     * @return Response
     */
    public function index(Request $request)
    {
        $keyword = $request->get('search');

        if ($keyword && $keyword == 'evaluations-menu') {

            $evaluations = HreEvaluations::with(
                'processes.evaluated.position.level.competencies.results',
                'creator',
                'processes.evaluated.user',
                'processes.evaluator.position.functions',
                'processes.evaluated.position.functions',
                'processes.evaluator.user',
                'processes.compromises.func',
                'processes.summaries')->get();

        } elseif ($keyword && $keyword != 'evaluations-menu') {

            $evaluations = HreEvaluations::with(
                'processes.evaluated.position.level.competencies.results',
                'creator',
                'processes.evaluated.user',
                'processes.evaluator.position.functions',
                'processes.evaluated.position.functions',
                'processes.evaluator.user',
                'processes.compromises.func',
                'processes.summaries')
                ->where('name', 'LIKE', "%$keyword%")
                ->where('description', 'LIKE', "%$keyword%")
                ->where('evaluation_type', 'LIKE', "%$keyword%")
                ->where('evaluateds_type', 'LIKE', "%$keyword%")
                ->where('global_start', 'LIKE', "%$keyword%")
                ->where('global_end', 'LIKE', "%$keyword%")
                ->paginate($this->perPage);

        } else {

            $evaluations = HreEvaluations::with(
                'processes.evaluated.position.level.competencies.results',
                'creator',
                'processes.evaluated.user',
                'processes.evaluator.position.functions',
                'processes.evaluated.position.functions',
                'processes.evaluator.user',
                'processes.compromises.func',
                'processes.summaries')->paginate($this->perPage);

        }

```

Ilustración 20 Controlador Backend

Fuente Propia

8 Módulos a desarrollar

8.1 Configurar organigrama

Es de vital importancia configurar el organigrama institucional, en correlación a los roles y poder identificar los actores que participaran en el proceso:

- **Trabajador:** quien en adelante se denominará evaluado, responderá objetivamente por su auto evaluación.
- **Jefe inmediato:** responsable de aplicar evaluación a los empleados administrativos y de cumplir con los tiempos de evaluación.
- **Pares**: o colegas son aquellos trabajadores que tienen el mismo nivel jerárquico del evaluado y son asignados para realizar la labor.
- **Colaboradores:** o subordinados son trabajadores que dependen directamente del evaluado.

8.2 Configurar manual de funciones

Para el proceso es necesario ingresar en la base de datos del sistema y al manual de funciones, con el propósito de conocer, seleccionar y evaluar las responsabilidades asignadas según el organigrama de la corporación y teniendo en cuenta la configuración del organigrama establecido con anterioridad en el sistema.

8.3 Ingresar contratación

Para lograr el proceso de desempeño, es necesario contener de manera digital los contratos de los administrativos según organigrama institucional, con el fin de identificar las funciones de los cargos que se evaluaran posteriormente.

8.4 Diseño de instrumentos

El sistema ofrece la opción de diseñar los instrumentos que se aplicaran en el proceso según formatos:

- FO-GH-01 FORMATO DE EVALUACIÓN PERIODO DE PRUEBA – Evaluador
- FO-GH-01 FORMATO DE EVALUACIÓN DEL DESEMPEÑO – Autoevaluación
- FO-GH-01 FORMATO DE EVALUACIÓN DEL DESEMPEÑO – Evaluadores.
- FO-GH-01 FORMATO DE COMPROMISOS EVALUACIÓN DEL DESEMPEÑO.
- FO-GH-01 FORMATO DE COMPROMISOS PERIODO DE PRUEBA.

8.5 Establecer Programación

A cargo del área de Talento Humano, programará el proceso de evaluación de desempeño por semestre a través de las opciones que están incluidas en este módulo, las cuales son:

- **Cronograma:** a través de este módulo permitirá establecer las fechas del proceso que se evaluará por semestre.
- **Configurar instrumentos:** Este módulo permitirá configurar los instrumentos (Prueba y permanencia) al funcionario a evaluar, en acuerdo con el jefe inmediato,

esta actividad se realizará con apoyo del sistema, seleccionado las funciones y compromisos “Según formatos configurados con anterioridad en el sistema.

- **Establecer muestra:** El sistema de forma aleatoria, mostrara una lista de los posibles evaluados y evaluadores, el jefe de talento humano tomara la decisión de seleccionar cada uno de estos a criterio propio.

8.6 Realizar evaluación

Los evaluadores a través de la aplicación calificaran a los evaluados según encuestas establecidas con anterioridad a través del sistema, este módulo contiene:

- **Evaluación de Permanencia:** Esta opción serán calificados los funcionarios con contratos de permanencia con CORHUILA, según formato configurado en el sistema con anterioridad.
- **Evaluación de prueba:** Esta opción podrán ser calificados los funcionarios que se encuentran en periodo de prueba con CORHUILA, según formato configurado en el sistema con anterioridad.
- **Control seguimiento**: El jefe de la división de talento humano, mediante este módulo podrá hacer seguimiento del proceso de la evaluación de desempeño, realizando una comparación con las fechas establecidas y de esta forma saber el dinamismo de la actividad, identificando aquellos evaluadores que están cerca de cumplírseles la fecha para la terminación del proceso de evaluación y así poder hacer una campaña más fuerte de sensibilización.

8.7 Análisis y medición de los datos

Finalizado el proceso de la evaluación, se procede a la etapa de recolección de datos y generación de informes mostrando los resultados de las encuestas en formato pdf y xls, este módulo comprende:

- **Elaboración de informe:** la información recolectada se consolida en un informe individual de manera estructurada, que representará gráficamente los resultados obtenidos con sus comparaciones estadísticamente, según formatos:FO-GH-01 FORMATO INFORME EVALUACIÓN DEL DESEMPEÑO y FO-GH-01 FORMATO INFORME PERIODO DE PRUEBA
- **Retroalimentación:** Los evaluados podrán ver sus resultados en línea, a través de una cuenta de usuario asignado por la división de talento humano.
- **Proceso de medición de desempeño:** El Proceso de Evaluación del Desempeño contiene la opción de seleccionar los modelos establecidos por el manual de procedimiento de desempeño, los cuales se utilizarán de acuerdo con las necesidades de la dirección y alineación con los objetivos estratégicos de la Corporación:
    - **Evaluación 90º**: La realizan dos actores: el evaluado (quien realiza una autoevaluación) y un evaluador que es el jefe inmediato (quien ha realizado seguimiento) directo a su desempeño y a sus actividades. Siempre se utilizará para el procedimiento de evaluación periodo de prueba (ver PC- GH- 01 PROCEDIMIENTO PERIODO DE PRUEBA).
    - **Evaluación 180º:** La evaluación de 180 grados, da a los trabajadores una visión amplia del desempeño al presentar una perspectiva evaluadora del jefe, la autoevaluación y los pares jerárquicos (compañeros del mismo nivel jerárquico del evaluado).

- **Evaluación 270º:** La evaluación de 270 grados, incluye la autoevaluación, la evaluación del Jefe, de los pares y subalternos.

Para el desarrollo de los informes anteriores, los resultados obtenidos del proceso se encontrarán basados en los criterios establecidos por el manual de proceso de desempeño.

	EVALUADO	ROLES DE EVALUADORES			
Criterios	Autoevaluación	Jefe	Pares	Subalternos	Clientes
Funciones	Peso 40%	Se promedia todas las evaluaciones -Peso 40%			
Competencias corporativas	Peso 40%	Se promedia todas las evaluaciones -Peso 40%			
Competencias por nivel jerárquico	Peso 20%	Se promedia todas las evaluaciones -Peso 20%			
Total	100%	100%			
Ponderación en la calificación final	**30%**	**70%**			

Tabla 1 Criterios manual desempeño

Fuente: Área Talento Humano CORHUILA

La sumatoria de la calificación final de la autoevaluación y la calificación final de los evaluadores daría el resultado final sobre el 100%

8.8 Rangos para interpretación de la calificación:

La calificación final tanto del periodo de prueba como de la evaluación del desempeño serán las siguientes:

La calificación aprobatoria de la misma será igual o superior al 60%. Inferior a ésta en el periodo de prueba será insumo para la desvinculación, teniendo en cuenta que el citado periodo es el instrumento de medición de ajuste del nuevo servidor tanto al cargo como a la organización. En la evaluación del desempeño será insumo para análisis de la continuidad de un trabajador por parte del jefe inmediato, jefe de personal y directivos.

- **Nivel adecuado:** Puntajes de 60 a 70 porciento, serán clasificados como aprobatorios.
- **Nivel satisfactorio:** Puntajes de 71 a 80 porciento, serán clasificados como aprobatorios.
- **Nivel óptimo:** Puntajes de 81 a 100.

8.9 Indicadores para evaluar la Medición del Desempeño

Las actividades del proceso de medición del desempeño se evaluarán con base en los siguientes indicadores

Qué se quiere medir	Qué evalúa el indicador	Fórmula	Unidad de Medida	Medios de verificación	Frecuencia de Medición
Cobertura	Alcance de las actividades de medición del desempeño	Nº trabajadores evaluados/No total de trabajadores *100	Nº participantes en el proceso de medición del desempeño	Informes de evaluación periodo de prueba y de desempeño	Semestral
Cumplimiento	% anual de cumplimiento de las actividades programadas	Nº Actividades realizadas/No Actividades programadas*100	Actividades de medición del desempeño	Registros de actividades realizadas	Semestral
Nivel de desempeño	Nivel de desempeño óptimo, satisfactorio y adecuado	Comparativo desempeño óptimo, satisfactorio y adecuado	Consolidación de resultados evaluación periodo de prueba y de desempeño	Bases de datos	Semestral

Tabla 2 Indicadores de desempeño

Fuente: Área Talento Humano CORHUILA

8.10 Modulo administrativo

A través de este módulo el administrador del sistema podrá tener control sobre este, facilitando:

- Crear usuarios.
- Ingresar parámetros que requiere la aplicación para su funcionamiento.

9 Diseño del sistema

9.1 Modelo Relacional

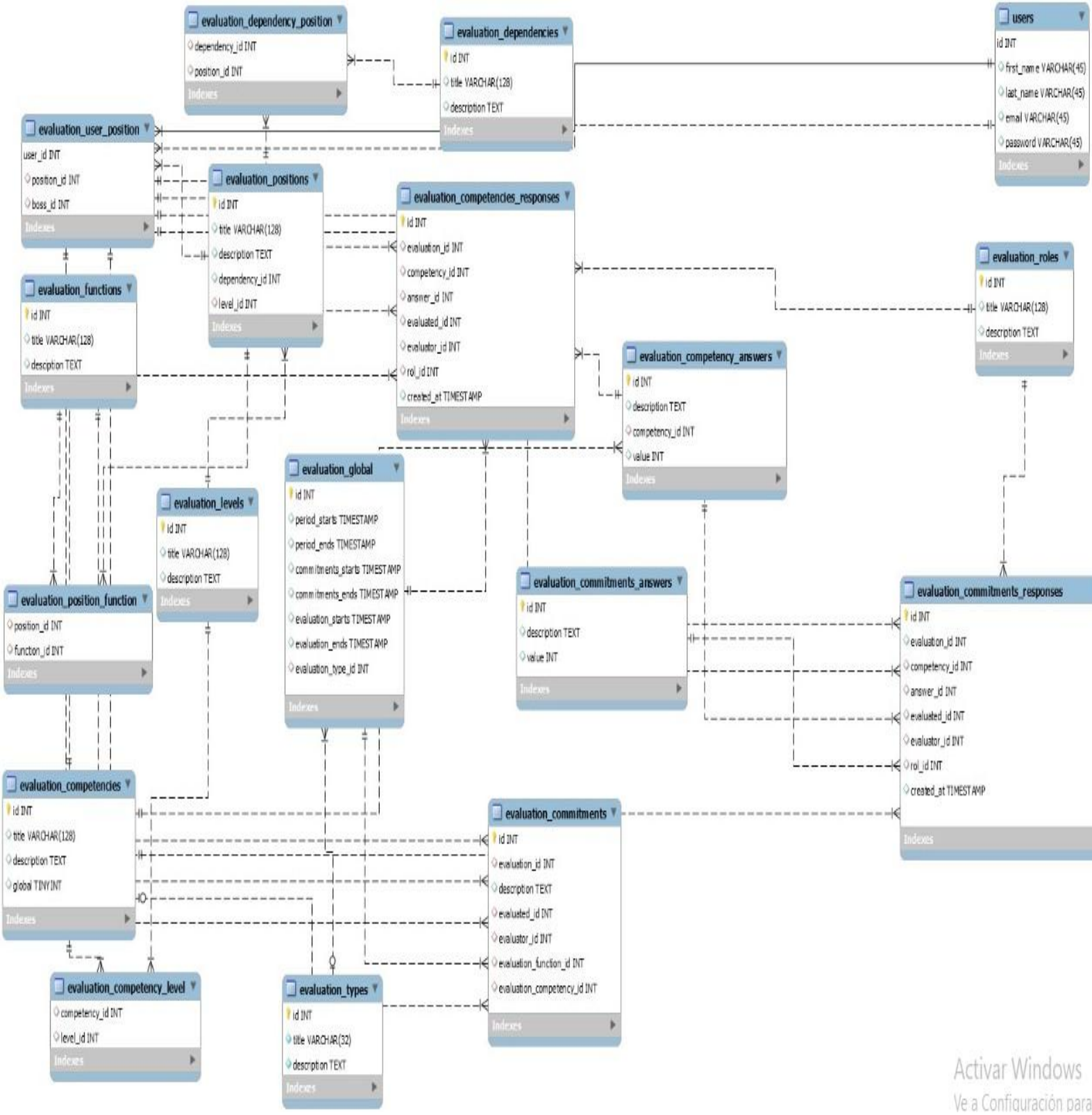

Ilustración 21 Modelo entidad relación

Fuente Propia

9.2 Diagrama casos de uso

Nombre	CU_01 Ingresar al sistema
Versión	1.0
Actores	Usuario
Descripción	Se valida el ingreso del usuario al sistema

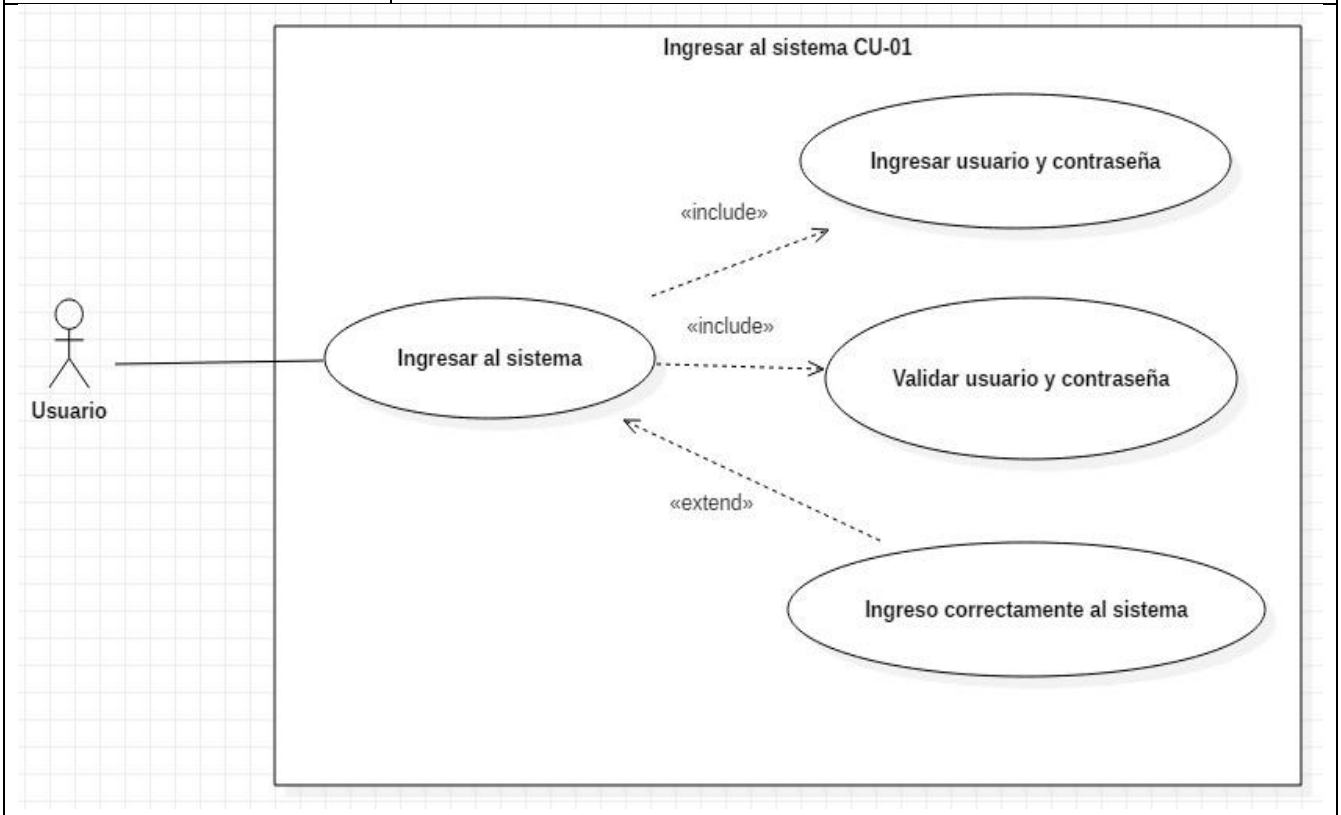

Ilustración 22 CU_01 Ingresar al sistema

Tabla 3 Caso de uso 01

Nombre	CU_02 Ingresar parámetros al sistema
Versión	1.0
Actores	Usuario
Descripción	Al ingresar el sistema , se ingresan los parámetros de la función , dependencia , rol , nivel y competencia del usuario

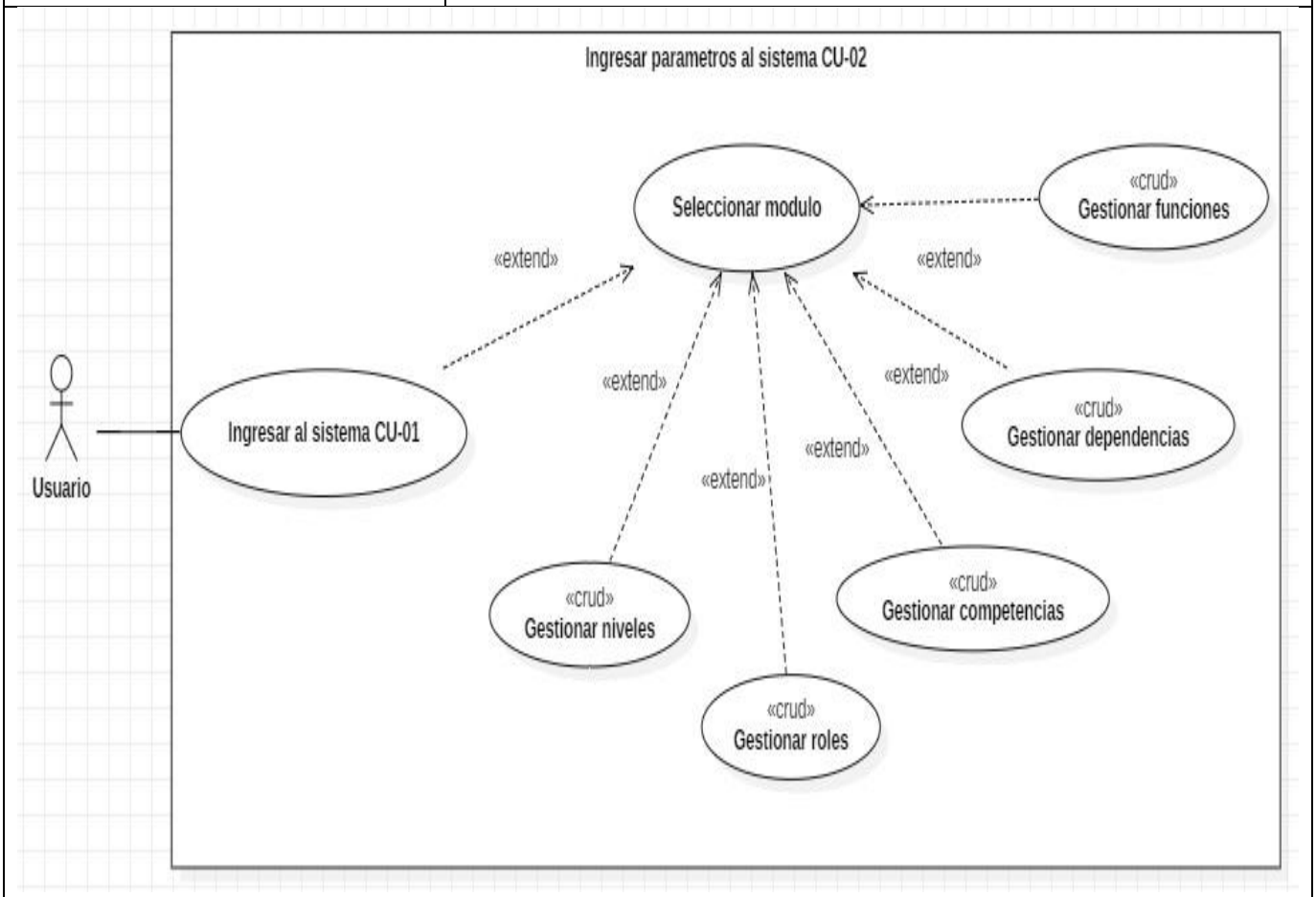

Ilustración 23 CU_02 Ingresar parámetros al sistema

Tabla 4 Caso de uso 02

Nombre	CU_03 Crear compromisos
Versión	1.0
Actores	Jefe inmediato
Descripción	Creación de los compromisos para la evaluación de desempeño

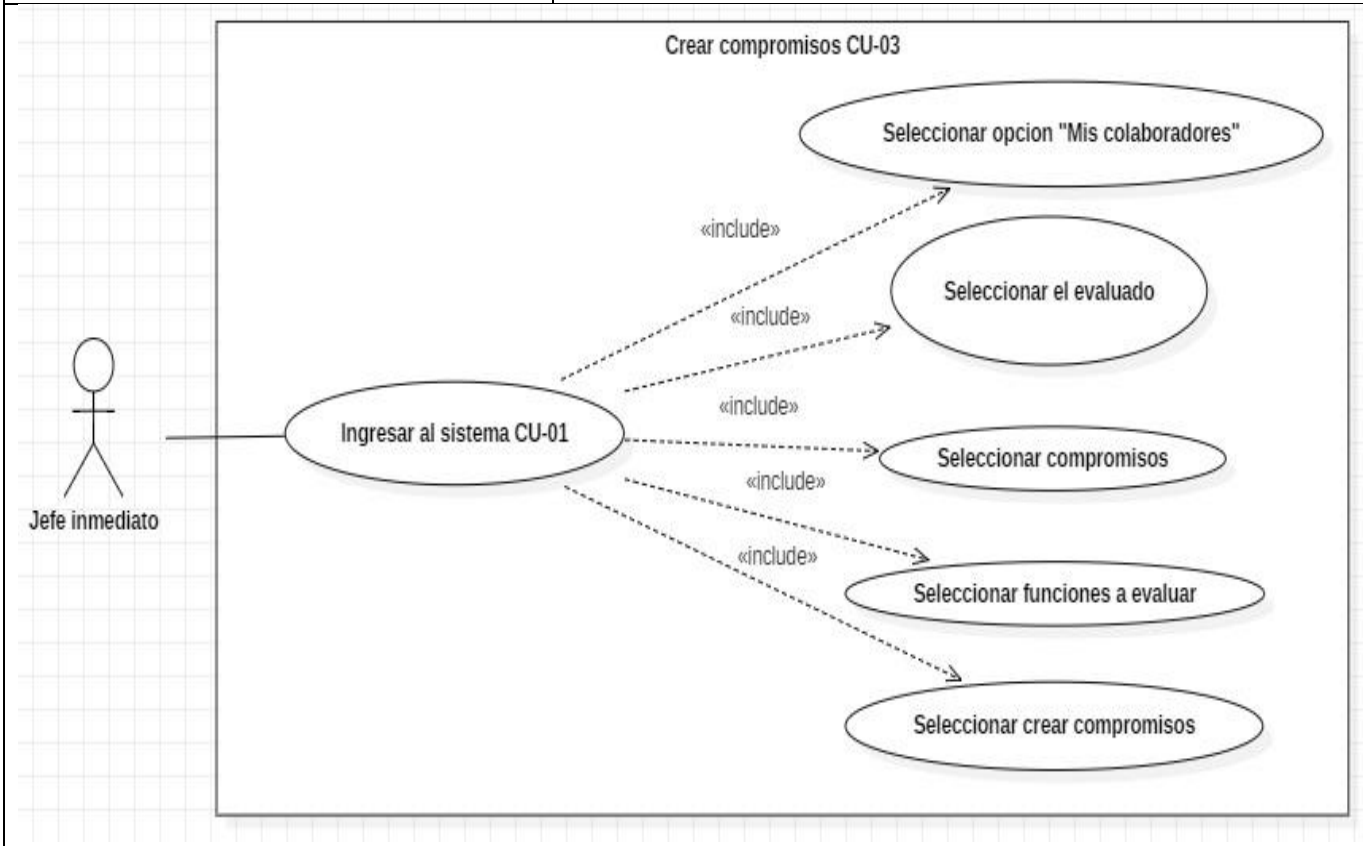

Ilustración 24 CU_03 Crear compromisos

Tabla 5 Caso de uso 03

Nombre	CU_04 Crear evaluación
Versión	1.0
Actores	Jefe Talento Humano
Descripción	Creación de la evaluación de desempeño

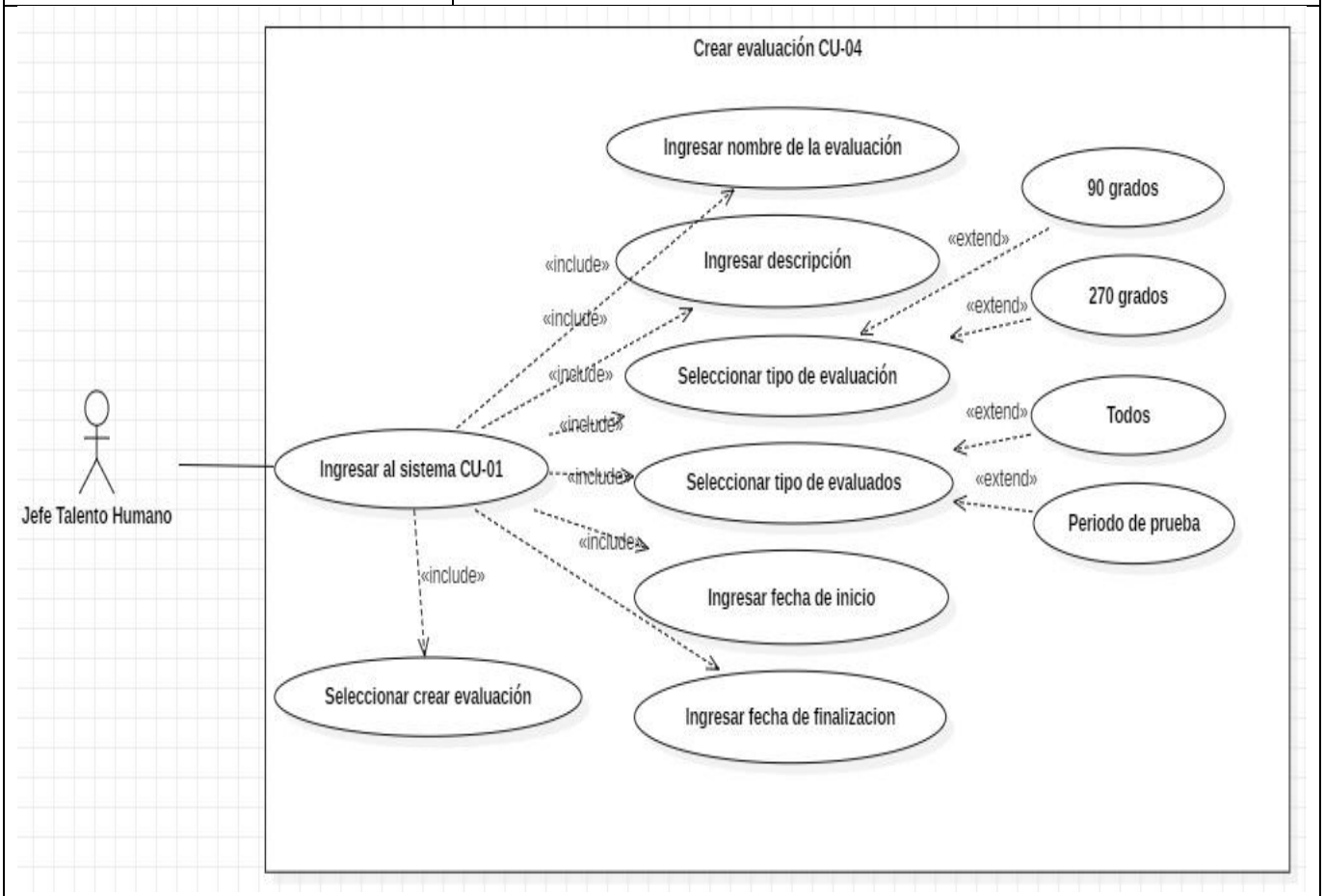

Ilustración 25 CU_04 Crear evaluación

Tabla 6 Caso de uso 04

Nombre	CU_05 Realizar evaluación
Versión	1.0
Actores	Jefe Inmediato
Descripción	Realizar la evaluación de desempeño

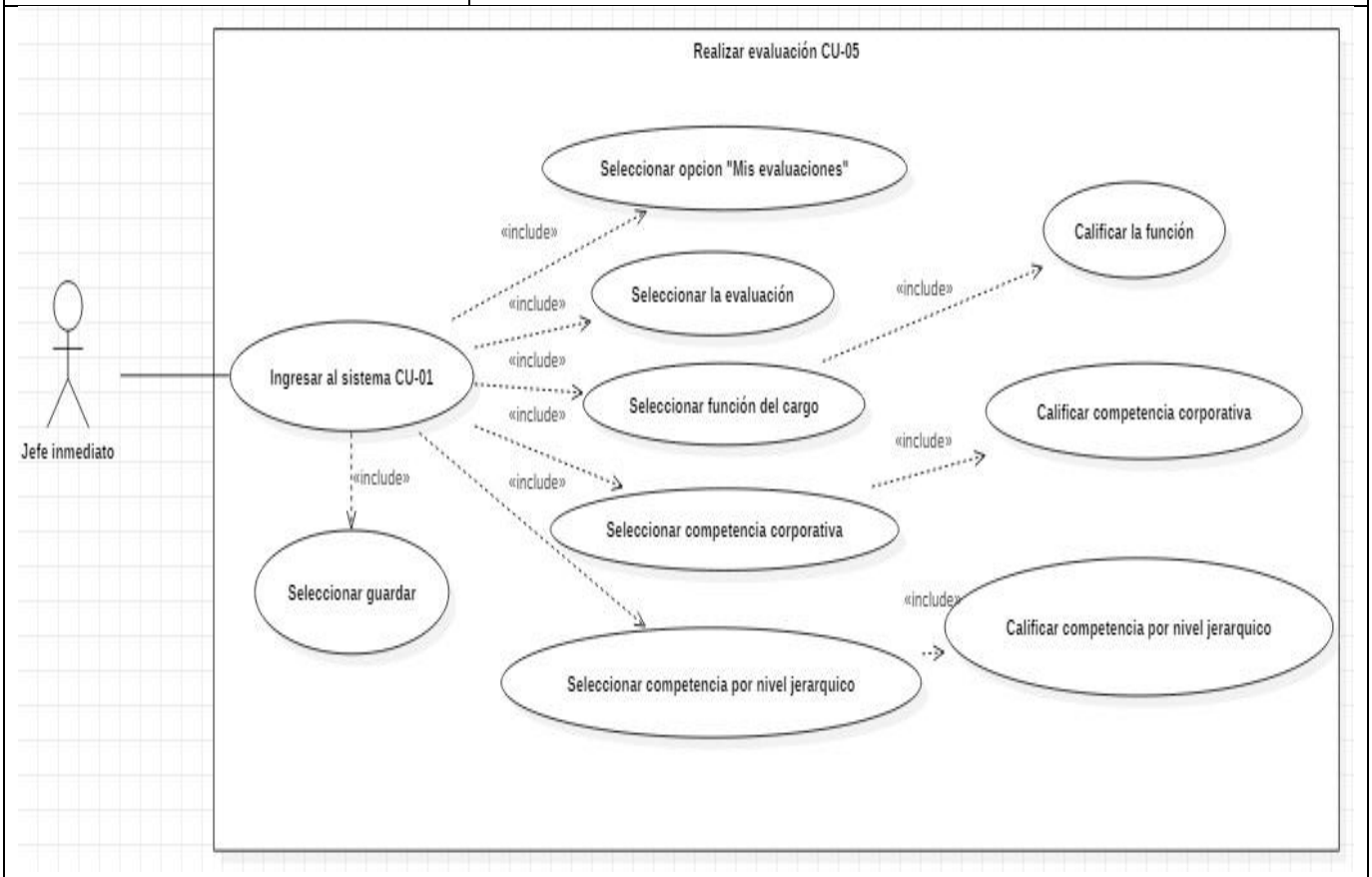

Ilustración 26 CU_05 Realizar evaluación

Tabla 7 Caso de uso 05

10 Cronograma general de actividades

ACTIVIDAD		DURACIÓN EN SEMANAS (Días Hábiles)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
No.	**Descripción**	**Fase**																														
1	Levantamiento de Información y definición de módulos.	Análisis																														
2	Ingeniería de requisitos. (Análisis y diseño).	Análisis y diseño																														
3	Preparación del ambiente, de desarrollo.	Diseño																														
4	Desarrollo.	Desarrollo																														
5	Despliegue (Instalación del Software en el entorno de producción)	Implementación																														
6	Pruebas (Primera evaluación de desempeño) y ajustes de los módulos en el entorno de pruebas	Pruebas																														
8	Soporte.																															
9	Entrega del Software																															

Tabla 8 Cronograma de actividades

Fuente Propia

11 Cronograma entrega de módulos

		Duración en semanas contados en días habiles																									
Id	Modulo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	Configurar organigrama																												
2	Configurar manual de funciones																												
3	Ingresar contratación																												
4	Diseño de instrumentos																												
5	Establecer Programación																												
6	Realizar evaluación:																												
7	Control seguimiento:																												
8	Análisis y medición de los datos																												
9	Módulo de Administrativo:																												
10	Margen de riesgos.																												

Tabla 9 Cronograma entrega de módulos

Fuente Propia

12 Condiciones de la propuesta

12.1 Costos

El desarrollo del Software tiene un costo de Veintidós millones seiscientos mil pesos ($22’600.000).

	Cantidad	Valor mes x Cantidad	Costos x mes	Tiempo en meses	sub total
Programadores	2	500000	1000000	5	5000000
Product owner	1	700000	700000	8	5600000
Director proyecto	1	1500000	1500000	8	12000000
			$3.200.000,00	**VALOR TOTAL PROYECTO**	**$22.600.000,00**

Tabla 10 Costos

Fuente Propia

12.2 Tiempo de ejecución del proyecto

El desarrollo del Software cuenta con una inversión de treinta y dos meses (32) semanas (días hábiles) de desarrollo e instalación en el entorno de producción y pruebas en la primera evaluación de desempeño (Prueba piloto).

12.3 Tiempo de garantía y soporte de la aplicación

Se ofrece la garantía por el periodo de la segunda evaluación de desempeño, en este tiempo no se adicionarán funcionalidades al Software; se harán correcciones de los posibles defectos de la aplicación que se puedan presentar y el periodo de soporte está comprendido entre las semanas 28 a la 32 que corresponde al acompañamiento y capacitación a usuarios.

12.4 Entrega del producto

El Software se entrega con lo siguiente:

1. Manual de usuario.
2. Documento arquitectónico, para TI.
3. CD con código fuente.

13 Aspectos legales

- CORHUILA es responsable de la licencia del motor de base de datos Oracle 11g.
- CORHUILA es responsable de la licencia de plataforma del servidor de pruebas y producción.
- Se utilizará herramientas de Software libre para el desarrollo de las interfaces de la aplicación, el cual no implica algún tipo de licenciamiento.
- Se hará entrega del código fuente, con un certificado por parte del director del proyecto especificando la propiedad de CORHUILA para su uso libre.
- La información que sea suministrada por parte del cliente (CORHUILA), será de total reserva por parte del grupo de desarrollo.

14 Bibliografía

ABC Consultorio. (16 de Febrero de 2015). Obtenido de https://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html

Alvarez, M. A. (17 de Octubre de 2014). *Desarrollo Web*. Obtenido de https://desarrolloweb.com/articulos/composer-gestor-dependencias-para-php.html

Alvarez, M. A. (15 de Octubre de 2015). *Desarrollo Web*. Obtenido de https://desarrolloweb.com/articulos/uso-bower-gestor-dependencias.html

Alvarez, M. A. (29 de Noviembre de 2016). *Desarrollo Web*. Obtenido de https://desarrolloweb.com/articulos/que-es-una-spa.html

Ambler, S., & Lines, M. (2012). *Disciplined Agile Delivery: A practicioner's guide to agile software delivery in the enterprise.* IBM Press.

AngularJS: A Modern MVC Framework in JavaScript. (2014). *Journal of Global Research in Computer Science*, 17-23.

Aplicación del modelado de procesos en un curso de ingeniería de software. (2001). *REDIE. Revista Electrónica de Investigación Educativa*.

ArcGis Pro. (s.f.). Obtenido de https://pro.arcgis.com/es/pro-app/help/data/query-layers/what-is-a-query-layer-.htm

Bahit, E. (s.f.). *POO y MVC en PHP.*

Boada Oriols, M., & Gomez Guiterrez, J. A. (2019). *El gran libro de Angular.* Alfaomega.

Caules, C. Á. (14 de Junio de 2013). *Arquitectura Java.* Obtenido de https://www.arquitecturajava.com/servicios-rest/

Celso, G. (2003). *Arquitectura de la Informacion: diseño e implementación* . Lima .

Cobo, A., Gomez, P., Perez, D., & Rocha, R. (2005). *PHP y MySQL Tecnologias para el desarrollo de aplicaciones web.* Diaz de Santos.

Codigo Facilito . (s.f.). Obtenido de https://codigofacilito.com/articulos/que-es-git

Conceptualización del proceso de implementación de software: perspectivas ágil y disciplinada. (2010). *Ciencia e Ingenieria*, 143-152.

Cubo Velasquez, A. (s.f.). *Representational State Transfer (REST). Un estilo de arquitectura* .

Desarrollo Web. (s.f.). Obtenido de https://desarrolloweb.com/javascript/

Develapps. (20 de 07 de 2016). Obtenido de http://www.develapps.com/es/noticias/modelo-vista-presentador-mvp-en-android

Digital Learning. (17 de Marzo de 2012). Obtenido de https://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/

Estudio y análisis de los framework en php. (2013). *Universidad Simon Bolivar*.

EVERAC. (23 de Junio de 2011). Obtenido de https://everac99.wordpress.com/2011/06/23/despliegue-continuo-continuous-delivery/

Fontela, A. (16 de Julio de 2015). *Raiola Networks*. Obtenido de https://raiolanetworks.es/blog/que-es-bootstrap/

Garcia Sanchez, M. d., Reyes Añorve, J., & Godinez Alarcon, G. (2017). Revista Iberoamericana de las ciencias sociales y humanisticas. *Las TIC en la educacion superior, innovaciones y retos*.

Gomez, J. C. (2015). Aplicacion web y movil usando Android y AngularJS contra Spring Web Services. *UPM*, 6.

Gonzalez, A. (2013). *Tu Programacion*. Obtenido de http://www.tuprogramacion.com/glosario/que-es-un-orm/

Guevara Benites, A. (20 de Abril de 2017). *DevCode*. Obtenido de https://devcode.la/blog/que-es-npm/

Guia al cuerpo de conocimiento de la ingenieria de software SWEBOK. (2004).

Guillermo Solarte, L. M. (2009). Modelos de Calidad Para Procesos de Software. *Scientia Et Technica*.

IEEE Computer Society. (2014). *SWEBOK: Guide to the Software Engineering Body of Knowledge.* IEEE.

Isaac, J. (2 de Octubre de 2015). *Conocimiento Digital*. Obtenido de https://grupo4herramientasinformatica.blogspot.com/2015/10/que-es-la-oracle.html

Jose H. Canos, P. L. (2012). Metodologias agiles en el desarrollo de software. *Universidad plitecnica de valencia*.

MADEJA. (s.f.). Obtenido de Marco de Desarrollo de la Junta de Andalucia: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/818

MADEJA. (s.f.). Obtenido de Marco de Desarrollo de la Junta de Andalucia: http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/seguridad

MADEJA. (s.f.). Obtenido de Marco de Desarrollo de la Junta de Andalucia: http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/71#libro-pautas-toc

Mateo, J. (2014). DICIPLINED AGILE DELIVERY (DAD). *Blog*.

Montero Ortega, J. M. (4 de Febrero de 2019). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/la-arquitectura-mvvm-y-sus-componentes/

Neo Attack. (s.f.). Obtenido de https://neoattack.com/neowiki/mysql/

Net Consulting. (27 de Enero de 2019). Obtenido de https://www.netconsulting.es/blog/nodejs/

Perez, E. (9 de Noviembre de 2014). *El Androide Libre*. Obtenido de https://elandroidelibre.elespanol.com/2014/11/que-es-material-design.html

Pressman, R. (2010). *Ingenieria del software un enfoque practico.* The McGraw-Hill.

Rees, D., & Laguna, A. (2013). *Laravel: Code Happy (ES).* Lean Publishing.

Rosa Moncayo, J. M. (17 de Mayo de 2018). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/

Ruiz Ruso, C., & Alvarez, M. A. (14 de Septiembre de 2015). *Desarrollo Web*. Obtenido de https://desarrolloweb.com/articulos/introduccion-modelos-laravel.html

Saffirio, M. (5 de Febrero de 2006). *Mario Saffirio*. Obtenido de https://msaffirio.wordpress.com/2006/02/05/%C2%BFque-son-los-web-services/

Santos Hernadez, D., & Serrano Parreño, A. (2017). *Desarrollo de un API REST para aplicaciones web y movil.*

Stavru, S. (2014). A critical examination of recent industrial surveys on agile method usage. *Journal of Systems and Software*, 87-97.

Torrado, J. (12 de Octubre de 2017). *Desarrollo Web*. Obtenido de https://desarrolloweb.com/articulos/introduccion-gitlab.html

Torres, G. (30 de Julio de 2016). *ReturnGis*. Obtenido de https://www.returngis.net/2016/07/que-es-gulp-y-para-que-sirve/

Universidad de Alicante. (s.f.). Obtenido de https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Yanes Enriquez, O., & Gracia del Busto, H. (2011). Mapeo Objeto/Relacional (ORM). *Revista Telematica*, 1-7.