Contenido

[1. Introducción 3](#_Toc10996851)

[2. Justificación 4](#_Toc10996852)

[3. Descripción del problema 4](#_Toc10996853)

[4. Formulación del problema 5](#_Toc10996854)

[5. Objetivos 5](#_Toc10996855)

[5.1. Objetivo general 5](#_Toc10996856)

[5.2. Objetivos específicos 5](#_Toc10996857)

[6. Alcance 6](#_Toc10996858)

[7. Técnicas orientadas a mejorar la productividad 8](#_Toc10996859)

[7.1. Uso de una metodología ágil de desarrollo de software 8](#_Toc10996860)

[7.2. Uso de herramientas durante el ciclo de vida 8](#_Toc10996861)

[7.3. Desarrollo de software basado en componentes 8](#_Toc10996862)

[8. Actividades adicionales para asegurar la calidad 9](#_Toc10996863)

[9. Arquitectura del software 9](#_Toc10996864)

[10. Ingeniería inversa y re-ingeniería 11](#_Toc10996865)

[10.1. Ingeniería inversa 11](#_Toc10996866)

[10.2. Re-ingeniería 13](#_Toc10996867)

[11. Descripción de herramientas de desarrollo 15](#_Toc10996868)

[11.1. PhpStorm 15](#_Toc10996869)

[11.2. Apache 15](#_Toc10996870)

[11.3. Enterprise Architect 16](#_Toc10996871)

[11.4. Git 17](#_Toc10996872)

[11.5. Visual Studio Code 17](#_Toc10996873)

[11.6. Postman 18](#_Toc10996874)

[12. Bitácora del proceso del software 19](#_Toc10996875)

[12.1. Modelo de requisito 19](#_Toc10996876)

[12.1.0. Diagrama de actividades móvil 19](#_Toc10996877)

[12.1.1. Identificar actores y casos de uso 21](#_Toc10996878)

[12.1.2. Detalle de casos de uso 22](#_Toc10996879)

[12.1.3. Diagrama general de casos de Uso 29](#_Toc10996880)

[13.2. Modelo de arquitectura del software a nivel del código 31](#_Toc10996881)

[12.3. Modelo de arquitectura del software a nivel de despliegue 32](#_Toc10996882)

[12.4. Modelo de persistencia 33](#_Toc10996883)

[12.4.1. Diagrama de clases 33](#_Toc10996884)

[12.4.2. Tabla de volumen 34](#_Toc10996885)

[12.4.3. Script SQL 37](#_Toc10996886)

[13. Referencias 43](#_Toc10996887)

# 1. Introducción

En la actualidad la importancia de las tecnologías de información está presente en todos los aspectos de nuestras vidas. Hemos adoptado una cultura que depende en gran parte al funcionamiento que estas nos brindan, como consecuencia de esto las organizaciones han tenido que evolucionar para mantenerse a la par de los constantes cambios que las TI generan en nuestro entorno, automatizando procesos industriales y requiriendo una infraestructura de TI sólida para poder funcionar, así como personal capacitado en TI que pueda prevenir riesgos y diseñar productos o servicios basados en las oportunidades que la tecnología brinda.

Teniendo en cuenta todo lo anteriormente mencionado, el proyecto a desarrollar busca implementar un sistema de control de personal utilizando las tecnologías de información actuales, el cual consta de una aplicación móvil para las plataformas Android e IOs y utiliza el GPS del dispositivo, el lector de huella, reconocimiento facila y el internet de este para marcar su ingreso y salida de su jornada laboral.

Actualmente según lo establecido en el código del trabajo, el empleador se encuentra obligado a mantener un registro para controlar las horas trabajadas de sus trabajadores. Dentro del marco legal, las empresas tienen derecho a implementar diferentes sistemas.

# 2. Justificación

Dentro de algunas organizaciones donde el pago de los empleados se calcula mediante el número de horas trabajadas, incluyendo el pago de horas extra; es importante tener un registro exacto de las entradas y salidas del personal, sin embargo, dichas asistencias actualmente se manejan por medio de tarjetones donde se registra tanto la hora de entrada como de salida, prolongando el proceso de verificación de asistencias y horas trabajadas, lo que genera demasiados cálculos manuales, que originan problemas por la pérdida de tarjetas y cálculos erróneos, lo cual provoca atrasos en la información. El problema se agrava si son demasiados empleados lo que lleva a contratar más personal para realizar el proceso.

Con este proyecto deseamos permitir acelerar el proceso de registro de asistencias para obtener información veraz y oportuna, generar reportes y poder distribuirla a todos los niveles de una organización.

# 3. Descripción del problema

Una empresa desea gestionar de una mejor manera la asistencia laboral de su personal. Ya que actualmente presentan las siguientes dificultades:

* No todos los trabajadores son oficinistas, sino que son encargados a visitar clientes. Y por ende ellos deben acercarse primero a la oficina asignada antes de hacer el viaje hasta donde el cliente. Lo mismo ocurre al momento de marcar salida, ya que el trabajador debe salir con anticipación de su lugar asignado fuera de la oficina para poder llegar a tiempo.
* Realización manual de los informes en cuanto al total de horas trabajadas, horas de retraso y horas extras de cada trabajador.

# 4. Formulación del problema

Se pretende desarrollar una aplicación web y otra móvil que gestione todo aquello que tenga que ver con el control de asistencia del personal de trabajo. La parte móvil funcionará en cualquier dispositivo con sistemas Android o IOS, capturando la ubicación GPS de la persona que vaya a marcar asistencia y una validación de identidad a través del reconocimiento fácil o a través de la huella dactilar. La parte web por su parte gestionará a los usuarios, sus asignaciones y capturará toda la información enviadas por los dispositivos móviles generando automáticamente informes sobre cada empleado.

# 5. Objetivos

## 5.1. Objetivo general

Desarrollar una aplicación web (PHP con Laravel) y otra móvil (multiplataforma Android & IOS) que gestione todo aquello que tenga que ver con el control de asistencia del personal de trabajo de la empresa mencionada.

## 5.2. Objetivos específicos

* Diseñar e implementar una aplicación web que sea capaz de gestionar a los empleados de la empresa, los clientes, lugares y horarios que pueden ser asignados además de realizar informes individuales en cuanto a sus horas trabajadas.
* Diseñar e implementar una aplicación móvil multiplataforma para dispositivos Android & IOS que permita iniciar sesión y marcar la asistencia y la salida de la jornada laboral.
* Utilizar recursos de los dispositivos móviles para incrementar la veracidad de los datos al momento de marcar asistencia, como ser: GPS, lector de huella y reconocimiento facial a través de la camara.

# 6. Alcance

La aplicación web debe ser capaz de realizar lo siguiente:

Registrar, editar y eliminar las ubicaciones (oficinas o sucursales) que posee la empresa. Donde los empleados pueden ser asignados y donde ellos deben marcar su asistencia. Se debe poder registrar el nombre de la ubicación, la dirección exacta y un teléfono. Además de señalar en un mapa el lugar exacto de su ubicación.

Registrar, editar y eliminar los horarios que existirán en la empresa, para los cuales se necesitará el nombre del horario, el turno al cual se aplicará y un detalle de los días y la hora de entrada y salida de cada día.

Registrar, editar y eliminar los clientes (personas, empresas o negocios) que posee la empresa. Donde los empleados pueden ser asignados y donde ellos deben marcar su asistencia. Se debe poder registrar el nombre de los clientes, la dirección exacta y un teléfono. Además de señalar en un mapa el lugar exacto de su ubicación.

Registrar, editar y eliminar los roles que tiene la empresa, requiriendo de ellos únicamente su nombre.

Registrar, editar y eliminar a los empleados de la empresa, guardando de ellos la siguiente información: nombre completo, teléfono, dirección, rol en la empresa, y una ubicación por defecto a la cual él está asignado.

Como parte principal se debe poder realizar asignaciones de los horarios, ubicaciones y clientes a los empleados. Pudiendo solo asignar una ubicación, ninguno o varios clientes y uno o muchos horarios.

También se debe mostrar un reporte con la información de la asistencia del empleado mensual. Que posea el nombre del empleado, la cantidad de horas asignadas, el total de horas trabajadas, señalando además las horas de retraso y las horas extras realizadas.

La aplicación Móvil debe realizar lo siguiente:

Un inicio y cierre de sesión, la pantalla principal debe estar preparada para el marcaje de la asistencia o salida del empleado, para validar la marcación de asistencia la aplicación capturará los datos del GPS, la fecha y hora exacta de ese momento y además de validar con la huella dactilar o a través del reconocimiento facial. Si los datos son correctos se enviarán directamente al servicio web, si no hubiera conexión, se debería guardar la información localmente para luego ser enviados al momento de reestablecerse la conexión. Debe también poder mostrar todo lo referente a sus asignaciones, o sea debería poder ver un detalle de la ubicación asignada, de cada uno de sus clientes y todos los horarios que tiene asignado. Además, debe mostrar un historial donde se mostrarán las veces que ha marcado asistencia y en qué momento.

# 7. Técnicas orientadas a mejorar la productividad

## 7.1. Uso de una metodología ágil de desarrollo de software

El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.

## 7.2. Uso de herramientas durante el ciclo de vida

Hoy en día se encuentra una vasta cantidad de tecnologías y herramientas que facilitan y agilizan el desarrollo de software, ayudando en todo el ciclo de vida de un proyecto, desde el diseño y modelado hasta la implementación, puesta en marcha y prueba. Ayudando de manera significativa a la reducción de tiempo y proporcionando agilidad en el proceso de desarrollo.

## 7.3. Desarrollo de software basado en componentes

La complejidad de los sistemas computacionales actuales nos ha llevado a buscar la reutilización del software existente. El desarrollo de software basado en componentes permite reutilizar piezas de código pre-elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.

El uso de este paradigma posee algunas ventajas:

* Reutilización del software.
* Simplifica las pruebas.
* Simplifica el mantenimiento del sistema.
* Mayor calidad

# 8. Actividades adicionales para asegurar la calidad

Se realizó una gran cantidad de distintas pruebas verificando en todos los posibles ambientes imaginados verificando de esta manera que se están cumpliendo con todos los requisitos solicitados.

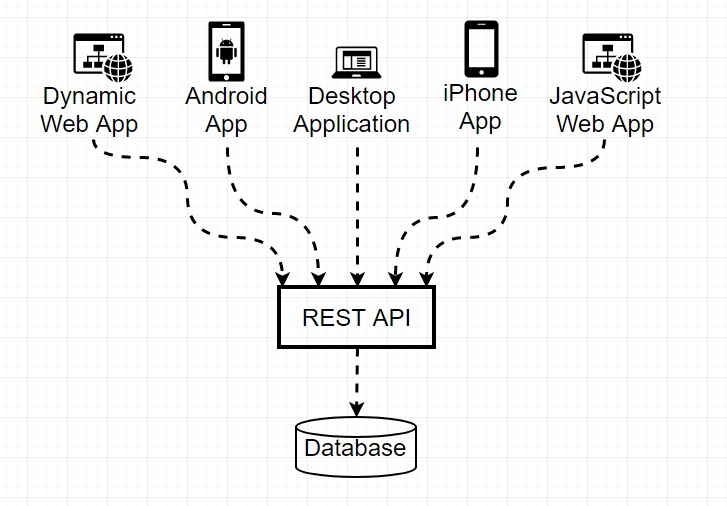
Se cuenta con un servicio de hosting estable y seguro, siendo que nuestra aplicación web y móvil funcionará únicamente con conexión a internet es muy necesario que esta se encuentre alojada en un lugar confiable, seguro y libre de caídas, ya que un problema en el servidor ocasionaría fallas al funcionamiento de la aplicación.

Se trabajó con una considerable población de datos para la verificación de la velocidad de respuesta del servicio.

# 9. Arquitectura del software

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. Es considerada el nivel más alto en el diseño de la arquitectura de un sistema puesto que establecen la estructura, funcionamiento e interacción entre las partes del software.

La arquitectura de software forma la columna vertebral para construir un sistema de software, es en gran medida responsable de permitir o no ciertos atributos de calidad del sistema entre los que se destacan la confiabilidad y el rendimiento del software. Además, es un modelo abstracto reutilizable que puede transferirse de un sistema a otro y que representa un medio de comunicación y discusión entre participantes del proyecto, permitiendo así la interacción e intercambio entre los desarrolladores con el objetivo final de establecer el intercambio de conocimientos y puntos de vista entre ellos.



El término REST (Representational State Transfer) un servicio REST es un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.

Hoy en día la mayoría de las empresas utilizan API REST para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web.

Según Fielding las restricciones que definen a un sistema RESTful serían:

* Cliente-servidor: esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y el servidor se “despreocupa” de cómo son usados los datos que envía al cliente.
* Sin estado: aquí decimos que cada petición que recibe el servidor debería ser independiente, es decir, no es necesario mantener sesiones.
* Cacheable: debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.
* Interfaz uniforme: define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección, “URI”.
* Sistema de capas: el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

# 10. Ingeniería inversa y re-ingeniería

## 10.1. Ingeniería inversa

La ingeniería inversa es el proceso llevado a cabo con el objetivo de obtener información o un diseño a partir de un producto, con el fin de determinar cuáles son sus componentes y de qué manera interactúan entre sí y cuál fue el proceso de fabricación.

El método se denomina así porque avanza en dirección opuesta a las tareas habituales de ingeniería, que consisten en utilizar datos técnicos para elaborar un producto determinado. En general, si el producto u otro material que fue sometido a la ingeniería inversa fue obtenido en forma apropiada, entonces el proceso es legítimo y legal. De la misma forma, pueden fabricarse y distribuirse, legalmente, los productos genéricos creados a partir de la información obtenida de la ingeniería inversa.

La ingeniería inversa es un método de resolución. Aplicar ingeniería inversa a algo supone profundizar en el estudio de su funcionamiento, hasta el punto de que podamos llegar a entender, modificar y mejorar dicho modo de funcionamiento.

La aplicación de ingeniería inversa nunca cambia la funcionalidad del producto objeto de la aplicación, sino que permite obtener productos que indican cómo se ha construido el mismo. Su realización permite obtener los siguientes beneficios:

* Reducir la complejidad del sistema: al intentar comprender el sistema se facilita su mantenimiento y la complejidad existente disminuye.
* Generar diferentes alternativas: del punto de partida del proceso, principalmente código fuente, se generan representaciones gráficas lo que facilita su comprensión.
* Recuperar y/o actualizar la información.
* Detectar efectos laterales: los cambios que se puedan realizar en un sistema puede conducirnos a que surjan efectos no deseados, esta serie de anomalías puede ser detectados por la ingeniería inversa.
* Facilitar la reutilización: por medio de la ingeniería inversa se pueden detectar componentes de posible reutilización de sistemas existentes, pudiendo aumentar la productividad, reducir los costes y los riesgos de mantenimiento.

## 10.2. Re-ingeniería

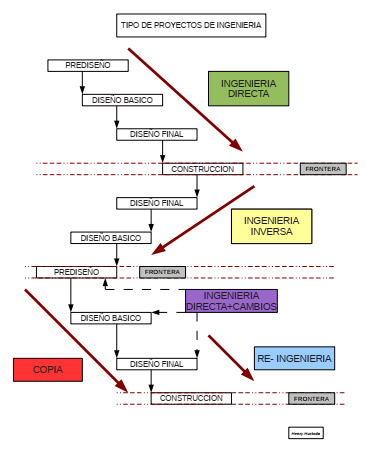
Reingeniería en un concepto simple es el rediseño de un proceso en un negocio o un cambio drástico de un proceso. A pesar que este concepto resume la idea principal de la reingeniería esta frase no envuelve todo lo que implica la reingeniería. Reingeniería es comenzar de cero, es un cambio de todo o nada, además ordena la empresa alrededor de los procesos. La reingeniería requiere que los procesos fundamentales de los negocios sean observados desde una perspectiva transfuncional y en base a la satisfacción del cliente.

Una definición rápida de reingeniería es "comenzar de nuevo". Reingeniería también significa el abandono de viejos procedimientos y la búsqueda de trabajo que agregue valor hacia el consumidor.

La definición más aceptada actualmente es la siguiente "La Reingeniería es el replanteamiento fundamental y el rediseño radical de los procesos del negocio para lograr mejoras dramáticas dentro de medidas críticas y contemporáneas de desempeño, tales como costo, calidad, servicio y rapidez". (Hammer 1994)

La implementación paulatina de un enfoque basado en procesos permite a una organización:

* Establecer indicadores de gestión para los procesos básicos de la organización e indicadores de resultados
* Eliminar actividades sin valor agregado
* Mejorar los flujos de información
* Reducir tiempos de operación
* Mantener los procesos focalizados en el cliente
* Mejorar la calidad del servicio



# 11. Descripción de herramientas de desarrollo

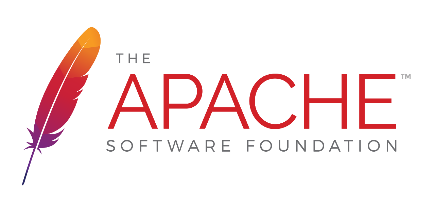
## 11.1. PhpStorm



PhpStorm de JetBrains es un PHP IDE inteligente y profesional que proporciona a los desarrolladores una mezcla de potentes herramientas inteligentes, hábitos útiles y mejores prácticas para el desarrollo de PHP, todos orientados hacia sacar provecho a la productividad del desarrollador.

* Asistencia de codificación inteligente
* Análisis de Calidad del código
* Entorno de desarrollo
* HTML/CSS/JavaScript Editor
* Fácil depuración y pruebas
* Refactorización rápida y segura

## 11.2. Apache



El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual según la normativa RFC 2616. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que alguien quería que tuviese la connotación de algo que es firme y enérgico, pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de Estados Unidos, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además, Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, a patchy server (un servidor "parcheado") suena igual que Apache Server.

* Modular
* Código abierto
* Multi-plataforma
* Extensible
* Popular (fácil conseguir ayuda/soporte)

## 11.3. Enterprise Architect



Sparx Systems Enterprise Architect es una herramienta de modelado y diseño visual basada en OMG UML. La plataforma admite: el diseño y la construcción de sistemas de software; modelar procesos comerciales; y modelado de dominios basados ​​en la industria. Las empresas y organizaciones lo usan no solo para modelar la arquitectura de sus sistemas, sino también para procesar la implementación de estos modelos en todo el ciclo de vida de desarrollo de aplicaciones.

* Modelado y análisis de negocios y datos
* Gestión de pruebas y simulación
* Análisis de ejecución visual
* Gestión del proyecto
* Integración con otras herramientas

## 11.4. Git



Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

* Compartición selectiva
* Velocidad y seguridad
* Ramificación
* Convergencia

## 11.5. Visual Studio Code



Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows , Linux y macOS. Incluye soporte para la depuración, control integrado de Git , resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor , los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software propietario.

* IntelliSense, una función que nos permite desde resaltar la sintaxis de nuestros proyectos.
* Cuenta con un avanzado depurador de código.
* Se integra en Git. Nos permite versionar nuestros trabajos fácilmente desde esta plataforma.
* Gracias a sus módulos, es compatible con prácticamente cualquier lenguaje de programación.

## 11.6. Postman



Postman surgió originariamente como una extensión para el navegador Google Chrome. A día de hoy dispone de aplicaciones nativas para MAC y Windows y están trabajando en una aplicación nativa para Linux (disponible en versión beta).

Está compuesto por diferentes herramientas y utilidades gratuitas (en la versión free) que permiten realizar tareas diferentes dentro del mundo API REST: creación de peticiones a APIs internas o de terceros, elaboración de tests para validar el comportamiento de APIs, posibilidad de crear entornos de trabajo diferentes (con variables globales y locales), y todo ello con la posibilidad de ser compartido con otros compañeros del equipo de manera gratuita (exportación de toda esta información mediante URL en formato JSON).

# 12. Bitácora del proceso del software

## 12.1. Modelo de requisito

### 12.1.0. Diagrama de actividades móvil

Iniciar Sesión



Marcar Asistencia



### 12.1.1. Identificar actores y casos de uso

Actores



Lista de casos de uso

*Aplicación Web*

* CU01 Gestionar Roles
* CU02 Gestionar Ubicaciones
* CU03 Gestionar Horarios
* CU04 Gestionar Clientes
* CU05 Gestionar Empleados
* CU06 Realizar Asignaciones
* CU07 Generar Reportes

*Aplicación Móvil*

* CU01 Iniciar Sesión
* CU02 Marcar Asistencia
* CU03 Ver Horarios
* CU04 Ver Ubicación
* CU05 Ver Clientes
* CU06 Ver Historial

### 12.1.2. Detalle de casos de uso



|  |  |
| --- | --- |
| Caso de uso | CU01 Gestionar Roles |
| Propósito | Registrar, editar y eliminar roles de los empleados de la empresa. |
| Descripción | Todos los empleados que pertenecen a la empresa deben tener un rol establecido el cual detallará la función que el realiza. El rol será asignado al momento de crear un nuevo empleado. Para crear un nuevo rol, solo se requiere el nombre que se le asignará. Ej. Administrador, Contador, Oficinista, etc. |
| Actores | Administrador |
| Actor iniciador | Administrador |
| Pre-condición | Ninguna |
| Flujo principal | 1. Registrar    1. Se accede al formulario y se inserta el nombre del rol.    2. Se validan los datos y se guardan. 2. Editar    1. Se seleccionar el rol a editar.    2. Se editan los campos.    3. Se validan los datos y se guardan. 3. Eliminar    1. Se selecciona el rol a eliminar.    2. Se verifica la acción.    3. Se guarda. |
| Post-condición | Ninguna. |
| Excepciones | 1b. Se deben llenar todos los campos.  2c. Se deben llenar todos los campos. |



|  |  |
| --- | --- |
| Caso de uso | CU02 Gestionar Ubicaciones |
| Propósito | Registrar, editar y eliminar las ubicaciones de la empresa. Conociendo como ubicación a los lugares de trabajo como ser una oficina o una sucursal. |
| Descripción | Las ubicaciones de la empresa señalan donde los empleados deben asistir para poder marcar su asistencia o salida. De las ubicaciones se guardarán los siguientes datos: nombre del lugar, dirección exacta, teléfono y se debe señalar la ubicación en un mapa. |
| Actores | Administrador |
| Actor iniciador | Administrador |
| Pre-condición | Ninguna |
| Flujo principal | 1. Registrar    1. Se accede al formulario y se inserta el nombre, la dirección, un número de teléfono y se coloca un marcador en el mapa.    2. Se validan los datos y se guardan. 2. Editar    1. Se seleccionar la ubicación a editar.    2. Se editan los campos.    3. Se validan los datos y se guardan. 3. Eliminar    1. Se selecciona la ubicación a eliminar.    2. Se verifica la acción.    3. Se guarda. |
| Post-condición | Ninguna. |
| Excepciones | 1b. Se deben llenar todos los campos.  2c. Se deben llenar todos los campos. |



|  |  |
| --- | --- |
| Caso de uso | CU03 Gestionar Horarios |
| Propósito | Registrar, editar y eliminar los horarios que se manejaran en la empresa. Siendo estos los que se podrán asignar a los empleados. |
| Descripción | Registrar, editar y eliminar los horarios que existirán en la empresa, para los cuales se necesitará el nombre del horario, el turno al cual se aplicará y un detalle de los días y la hora de entrada y salida de cada día |
| Actores | Administrador |
| Actor iniciador | Administrador |
| Pre-condición | Ninguna |
| Flujo principal | 1. Registrar    1. Se accede al formulario y se inserta el nombre del horario, se selecciona el turno al cual pertenecerá y se agrega todos los días que compondrán el horario, se agrega hora de entrada y de salida.    2. Se validan los datos y se guardan. 2. Editar    1. Se seleccionar el horario a editar.    2. Se editan los campos.    3. Se validan los datos y se guardan. 3. Eliminar    1. Se selecciona el horario a eliminar.    2. Se verifica la acción.    3. Se guarda. |
| Post-condición | Ninguna. |
| Excepciones | 1b. Se deben llenar todos los campos.  2c. Se deben llenar todos los campos. |



|  |  |
| --- | --- |
| Caso de uso | CU04 Gestionar Clientes |
| Propósito | Registrar, editar y eliminar los clientes de la empresa. |
| Descripción | Los clientes de la empresa son lugares donde los empleados también podrán marcar su asistencia o salida. De los clientes se guardarán los siguientes datos: nombre del cliente, dirección exacta, teléfono y se debe señalar la ubicación en un mapa. |
| Actores | Cliente |
| Actor iniciador | Cliente |
| Pre-condición | Ninguna |
| Flujo principal | 1. Registrar    1. Se accede al formulario y se inserta el nombre, la dirección, un número de teléfono y se coloca un marcador en el mapa.    2. Se validan los datos y se guardan. 2. Editar    1. Se seleccionar el cliente a editar.    2. Se editan los campos.    3. Se validan los datos y se guardan. 3. Eliminar    1. Se selecciona el cliente a eliminar.    2. Se verifica la acción.    3. Se guarda. |
| Post-condición | Ninguna. |
| Excepciones | 1b. Se deben llenar todos los campos.  2c. Se deben llenar todos los campos. |



|  |  |
| --- | --- |
| Caso de uso | CU05 Gestionar Empleados |
| Propósito | Registrar, editar y eliminar a los empleados de la empresa. |
| Descripción | Todos los empleados que pertenecen a la empresa deben ser registrados con la siguiente información: nombre completo, dirección y teléfono, una foto, se debe también seleccionar el rol que desempeña y se debe seleccionar una ubicación por defecto a la que él debe asistir para marcar su asistencia. |
| Actores | Empleado |
| Actor iniciador | Empleado |
| Pre-condición | Ninguna |
| Flujo principal | 1. Registrar    1. Se accede al formulario y se inserta el nombre completo, se selecciona el rol y la ubicación, se agrega una imagen desde los archivos propios y se proporciona una dirección y teléfono.    2. Se validan los datos y se guardan. 2. Editar    1. Se seleccionar el empleado a editar.    2. Se editan los campos.    3. Se validan los datos y se guardan. 3. Eliminar    1. Se selecciona el empleado a eliminar.    2. Se verifica la acción.    3. Se guarda. |
| Post-condición | Ninguna. |
| Excepciones | 1b. Se deben llenar todos los campos.  2c. Se deben llenar todos los campos. |



|  |  |
| --- | --- |
| Caso de uso | CU06 Realizar Asignaciones |
| Propósito | Relacionar al empleado con uno o más horarios de trabajo, una ubicación y con ningún o muchos clientes. |
| Descripción | Como parte principal se debe poder realizar asignaciones de los horarios, ubicaciones y clientes a los empleados. Pudiendo solo asignar una ubicación, ninguno o varios clientes y uno o muchos horarios. |
| Actores | Administrador |
| Actor iniciador | Administrador |
| Pre-condición | Ninguna |
| Flujo principal | 1. Asignar Ubicación    1. Se selecciona el lugar de trabajo en el que desea que el empleado marque su asistencia o salida.    2. Se validan los datos y se guardan. 2. Asignar Horarios    1. Se agrega un horario luego de seleccionarlo y aceptar.    2. Se validan los datos y se guardan. 3. Asignar Clientes    1. Se selecciona el cliente de la empresa en el que desea que el empleado marque su asistencia o salida.    2. Se validan los datos y se guardan. |
| Post-condición | Ninguna. |
| Excepciones | 2a No se puede asignar un horario por choque de horarios. |



|  |  |
| --- | --- |
| Caso de uso | CU07 Generar Reportes |
| Propósito | Mostrar un reporte mensual de horas trabajadas de cada empleado. |
| Descripción | Mostrar un reporte con la información de la asistencia del empleado mensual. Que posea el nombre del empleado, la cantidad de horas asignadas, el total de horas trabajadas, señalando además las horas de retraso y las horas extras realizadas. |
| Actores | Administrador |
| Actor iniciador | Administrador |
| Pre-condición | Ninguna |
| Flujo principal | 1. Generar Reporte    1. Se selecciona el mes del cual ser quiere ver el reporte de asistencias y horas trabajadas.    2. Se realiza la consulta y se muestra el reporte en una nueva pestaña o se imprime un PDF. |
| Post-condición | Ninguna. |
| Excepciones | Ninguna. |

### 12.1.3. Diagrama general de casos de Uso

Aplicación Web



Aplicación Móvil



## 13.2. Modelo de arquitectura del software a nivel del código



## 12.3. Modelo de arquitectura del software a nivel de despliegue



## 12.4. Modelo de persistencia

### 12.4.1. Diagrama de clases



### 12.4.2. Tabla de volumen

Rol:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| id | Numérico | 11 | PK | No | Identificador numérico. |
| Nombre | Texto | 255 | - | No | Nombre del rol. |

Ubicacion:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| id | Numérico | 11 | PK | No | Identificador numérico. |
| Nombre | Texto | 255 | - | No | Nombre de la ubicación. |
| Latitud | Numérico | 11 | - | No | Coordenada de ubicación. |
| Longitud | Numérico | 11 | - | No | Coordenada de ubicación. |
| Direccion | Texto | 255 | - | No | Dirección de la ubicación. |
| Telefono | Texto | 255 | - | No | Teléfono de la ubicación. |

Cliente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| id | Numérico | 11 | PK | No | Identificador numérico. |
| Nombre | Texto | 255 | - | No | Nombre del cliente. |
| Latitud | Numérico | 11 | - | No | Coordenada de ubicación. |
| Longitud | Numérico | 11 | - | No | Coordenada de ubicación. |
| Direccion | Texto | 255 | - | No | Dirección del cliente. |
| Telefono | Texto | 255 | - | No | Teléfono del cliente. |

Horario:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| id | Numérico | 11 | PK | No | Identificador numérico. |
| Nombre | Texto | 255 | - | No | Nombre del horario. |
| Turno | Texto | 255 | - | No | Turno del horario. |

Dia:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| id | Numérico | 11 | PK | No | Identificador numérico. |
| Nombre | Texto | 255 | - | No | Nombre del día. |
| Entrada | Texto | 5 | - | No | Hora de entrada. |
| Salida | Texto | 5 | - | No | Hora de salida. |
| Horario\_id | Numérico | 11 | FK | No | Identificador del horario. |

Empleado:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| id | Numérico | 11 | PK | No | Identificador del usuario |
| nombre | Texto | 255 | - | No | Nombre del usuario |
| Email | Texto | 255 | - | No | Email del usuario |
| password | Texto | 255 | - | No | Password codificada del usuario |
| huella | Texto | 255 | - | No | Huella del empleado. |
| foto | Texto | 255 | - | No | Foto del empleado. |
| dirección | Texto | 255 | - | No | Dirección del empleado. |
| teléfono | Texto | 8 | - | No | Teléfono del empleado. |
| Rol\_id | Numérico | 11 | FK | No | Identificador del rol. |
| ubicacion\_id | Numérico | 11 | FK | No | Identificador de la ubicación. |

Asistencia:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| Id | Numérico | 11 | PK | No | Identificador numérico. |
| Fecha | Texto | 255 | - | No | Fecha de la asistencia. |
| Entrada | Texto | 5 | - | No | Hora de entrada. |
| Salida | Texto | 5 | - | No | Hora de salida. |
| Estado | Texto | 255 | - | No | Estado de la asistencia. |
| user\_id | Numérico | 11 | FK | No | Identificador del empleado. |

Asignacion Cliente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| Id | Numérico | 11 | PK | No | Identificador numérico. |
| Cliente\_id | Numérico | 11 | FK | No | Identificador cliente. |
| user\_id | Numérico | 11 | FK | No | Identificador empleado. |

Asignacion Horarios:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE** | **TIPO DE DATO** | **AMPLITUD** | **LLAVE** | **NULO** | **DESCRIPCIÓN** |
| Id | Numérico | 11 | PK | No | Identificador numérico. |
| horario\_id | Numérico | 11 | FK | No | Identificador del horario. |
| user\_id | Numérico | 11 | FK | No | Identificador del empleado. |

### 12.4.3. Script SQL

DROP TABLE IF EXISTS `asistencia`;

CREATE TABLE IF NOT EXISTS `asistencia` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`fecha` date NOT NULL,

`entrada` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`salida` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`estado` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`user\_id` int(10) UNSIGNED NOT NULL,

PRIMARY KEY (`id`),

KEY `asistencia\_user\_id\_foreign` (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `a\_clientes`;

CREATE TABLE IF NOT EXISTS `a\_clientes` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`cliente\_id` int(10) UNSIGNED NOT NULL,

`user\_id` int(10) UNSIGNED NOT NULL,

PRIMARY KEY (`id`),

KEY `a\_clientes\_cliente\_id\_foreign` (`cliente\_id`),

KEY `a\_clientes\_user\_id\_foreign` (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `a\_horarios`;

CREATE TABLE IF NOT EXISTS `a\_horarios` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`horario\_id` int(10) UNSIGNED NOT NULL,

`user\_id` int(10) UNSIGNED NOT NULL,

PRIMARY KEY (`id`),

KEY `a\_horarios\_horario\_id\_foreign` (`horario\_id`),

KEY `a\_horarios\_user\_id\_foreign` (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `cliente`;

CREATE TABLE IF NOT EXISTS `cliente` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`nombre` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`direccion` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`telefono` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`latitud` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`longitud` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`visible` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `dia`;

CREATE TABLE IF NOT EXISTS `dia` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`nombre` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`entrada` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`salida` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`horario\_id` int(10) UNSIGNED NOT NULL,

PRIMARY KEY (`id`),

KEY `dia\_horario\_id\_foreign` (`horario\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `horario`;

CREATE TABLE IF NOT EXISTS `horario` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`nombre` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`turno` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`visible` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `migrations`;

CREATE TABLE IF NOT EXISTS `migrations` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`migration` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`batch` int(11) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `oauth\_access\_tokens`;

CREATE TABLE IF NOT EXISTS `oauth\_access\_tokens` (

`id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`user\_id` int(11) DEFAULT NULL,

`client\_id` int(11) NOT NULL,

`name` varchar(191) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`scopes` text COLLATE utf8mb4\_unicode\_ci,

`revoked` tinyint(1) NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

`expires\_at` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `oauth\_access\_tokens\_user\_id\_index` (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `oauth\_auth\_codes`;

CREATE TABLE IF NOT EXISTS `oauth\_auth\_codes` (

`id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`user\_id` int(11) NOT NULL,

`client\_id` int(11) NOT NULL,

`scopes` text COLLATE utf8mb4\_unicode\_ci,

`revoked` tinyint(1) NOT NULL,

`expires\_at` datetime DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `oauth\_clients`;

CREATE TABLE IF NOT EXISTS `oauth\_clients` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`user\_id` int(11) DEFAULT NULL,

`name` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`secret` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`redirect` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`personal\_access\_client` tinyint(1) NOT NULL,

`password\_client` tinyint(1) NOT NULL,

`revoked` tinyint(1) NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `oauth\_clients\_user\_id\_index` (`user\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `oauth\_personal\_access\_clients`;

CREATE TABLE IF NOT EXISTS `oauth\_personal\_access\_clients` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`client\_id` int(11) NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `oauth\_personal\_access\_clients\_client\_id\_index` (`client\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `oauth\_refresh\_tokens`;

CREATE TABLE IF NOT EXISTS `oauth\_refresh\_tokens` (

`id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`access\_token\_id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`revoked` tinyint(1) NOT NULL,

`expires\_at` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `oauth\_refresh\_tokens\_access\_token\_id\_index` (`access\_token\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `rol`;

CREATE TABLE IF NOT EXISTS `rol` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`nombre` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`visible` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `ubicacion`;

CREATE TABLE IF NOT EXISTS `ubicacion` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`nombre` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`direccion` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`telefono` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`latitud` double NOT NULL,

`longitud` double NOT NULL,

`visible` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

DROP TABLE IF EXISTS `users`;

CREATE TABLE IF NOT EXISTS `users` (

`id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

`nombre` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`email` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`password` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`huella` varchar(191) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`foto` varchar(191) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`direccion` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`telefono` varchar(191) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`visible` tinyint(1) NOT NULL DEFAULT '1',

`remember\_token` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`rol\_id` int(10) UNSIGNED NOT NULL,

`ubicacion\_id` int(10) UNSIGNED NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `users\_email\_unique` (`email`),

KEY `users\_rol\_id\_foreign` (`rol\_id`),

KEY `users\_ubicacion\_id\_foreign` (`ubicacion\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

ALTER TABLE `asistencia`

ADD CONSTRAINT `asistencia\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`);

ALTER TABLE `a\_clientes`

ADD CONSTRAINT `a\_clientes\_cliente\_id\_foreign` FOREIGN KEY (`cliente\_id`) REFERENCES `cliente` (`id`),

ADD CONSTRAINT `a\_clientes\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`);

ALTER TABLE `a\_horarios`

ADD CONSTRAINT `a\_horarios\_horario\_id\_foreign` FOREIGN KEY (`horario\_id`) REFERENCES `horario` (`id`),

ADD CONSTRAINT `a\_horarios\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`);

ALTER TABLE `dia`

ADD CONSTRAINT `dia\_horario\_id\_foreign` FOREIGN KEY (`horario\_id`) REFERENCES `horario` (`id`);

ALTER TABLE `users`

ADD CONSTRAINT `users\_rol\_id\_foreign` FOREIGN KEY (`rol\_id`) REFERENCES `rol` (`id`),

ADD CONSTRAINT `users\_ubicacion\_id\_foreign` FOREIGN KEY (`ubicacion\_id`) REFERENCES `ubicacion` (`id`);

COMMIT;

# 13. Referencias

*EcuRed*. (s.f.). Obtenido de Arquitectura de software: https://www.ecured.cu/Arquitectura\_de\_software

*Instinto Binario*. (s.f.). Obtenido de Desarrollo Software Basado en Componentes: https://instintobinario.com/890/

*Monografias*. (s.f.). Obtenido de Reingeniería y gestión de la calidad: https://www.monografias.com/trabajos76/reingenieria-gestion-calidad/reingenieria-gestion-calidad2.shtml

*Wikipedia*. (s.f.). Obtenido de Ingeniería inversa: https://es.wikipedia.org/wiki/Ingenier%C3%ADa\_inversa

*Wikipedia*. (s.f.). Obtenido de Desarrollo ágil de software: https://es.wikipedia.org/wiki/Desarrollo\_%C3%A1gil\_de\_software