|  |  |
| --- | --- |
|  |  |
| Memoria Proyecto DAMLLER |  |
|  |  |
|  | Álvaro Del Val  Mario Bueno  Mohamad El Sayed |

Contenido

[Memoria Proyecto DAMLLER 0](#_Toc136640919)

[1- Introducción 2](#_Toc136640920)

[2 - Objetivos de nuestro proyecto 2](#_Toc136640921)

[3 - Metodología y Desarrollo del proyecto 3](#_Toc136640922)

[4 – Planificación de tareas y de responsabilidades 3](#_Toc136640923)

[5 - Tecnologías utilizadas 4](#_Toc136640924)

[6 - Desarrollo e Implementación Web 5](#_Toc136640925)

[Diseño inicial de la web 5](#_Toc136640926)

[Descripción de la Arquitectura: 5](#_Toc136640927)

[7 - Diseño de bases de datos 7](#_Toc136640928)

[Modelo entidad relación: 7](#_Toc136640929)

[Modelo físico 8](#_Toc136640930)

[Tablas Implementadas en DataGrip 9](#_Toc136640931)

[Scripts 9](#_Toc136640932)

[Script para la creación de la base de datos: 9](#_Toc136640933)

[Script para la creación de procedimientos, disparadores y eventos: 10](#_Toc136640934)

[8 - Diagramas y desarrollo del programa 12](#_Toc136640935)

[Diagrama de caso de usos 12](#_Toc136640936)

[Diagrama de secuencia 13](#_Toc136640937)

[Diagrama para crear una nueva cita 13](#_Toc136640938)

[Diagrama para actualizar el trabajador de una cita 13](#_Toc136640939)

[Diagrama para imprimir un informe 14](#_Toc136640940)

[Diagrama de clases de modelo 14](#_Toc136640941)

[9 - Aplicación de Escritorio Gestión de Citas: 15](#_Toc136640942)

[Controladores : 16](#_Toc136640943)

[ModelViews : 16](#_Toc136640944)

[Repositorios 18](#_Toc136640945)

[Servicios 19](#_Toc136640946)

[Storages 19](#_Toc136640947)

[DatabaseManger 20](#_Toc136640948)

[Railway oriented programming 20](#_Toc136640949)

[Inyección de Dependencias con Koin 21](#_Toc136640950)

[Dependencias: 22](#_Toc136640951)

[Interfaz Grafica 23](#_Toc136640952)

[Vista Principal 23](#_Toc136640953)

# 1- Introducción

DAMLLER busca ocuparse de proporcionar un Sistema Informático completo para la gestión de distintas sedes de ITV.

Este proyecto estará dividido en 3 grandes bloques, una página web, un sistema de bases de datos para almacenar información y una aplicación de escritorio para que los trabajadores gestionen las citas de los clientes.

# 2 - Objetivos de nuestro proyecto

El Objetivo principal de este proyecto es poder recrear el proceso de planificación, de creación, de diseño y de implementación de un proyecto para una ITV para distintos tipos de vehículo ,con la idea de poder recrear un proceso de Sprint de una semana de duración.

Dados los requisitos proporcionados por un cliente deberemos :

* Saber poder identificar los requisitos funcionales, no funcionales y de información del proyecto.
* Realizar la correcta organización del proyecto a través de un sistema de gestión de tareas y responsabilidades con la metodología Kanban, organizando las tareas de la forma más eficiente posible y tener un seguimiento de estas.
* Saber utilizar un sistema de control de versiones como Git a través de GitHub incluyendo el correcto uso de aplicaciones como GitKraken y GitHub Desktop.
* Diseñar e implementar una página web que se encargue de dar a conocer nuestra empresa al público y que permita al usuario pedir una Cita con facilidad.
* Diseñar e implementar una Base de datos que se encargue de almacenar toda la información de las Citas, incluyendo los vehículos y los clientes junto a los informes de estas, incluyendo un historial de cambios a largo del tiempo de los informes almacenados y un sistema de borrado de las citas con más de dos de antigüedad.
* Una aplicación de escritorio que permita a nuestros trabajadores gestionar las citas de los usuarios, encargándose de generar informes con la información de las distintas inspecciones.
* Logotipo

  Descripción generada automáticamenteRealizar un análisis tecnológico para el completo desarrollo del proyecto.

# 3 - Metodología y Desarrollo del proyecto

Para llevar a cabo el desarrollo de este proyecto se harán uso de distintos tipos de metodologías ágiles y buenas prácticas en el mundo del desarrollo del software.

Se utilizará la metodología Kanban para la correcta gestión de tareas y responsabilidades de todos los miembros del equipo.

El empleo de los Principios SOLID para la correcta implementación del código y el futuro mantenimiento y expansión de este, lo que permitirá arreglar los posibles errores y ampliar de una forma cómoda el código a nuevas funcionalidades en el futuro .

El uso de Railway oriented programming para el control y gestión de errores.

# 4 – Planificación de tareas y de responsabilidades

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente  
Todo proyecto consta de varias partes y etapas de su desarrollo. Para poder trabajar de una forma cómoda y organizada usaremos Trello , una aplicación web que utiliza la metodología Kanban que nos permite organizar cada parte del proyecto, asignándoles prioridades, fechas de entrega, tags personalizadas , advertencias y mensajes entre otras cosas.

Tendremos distintos tableros, cada uno orientado a una de las grandes partes del proyecto y una individual para los apartados generales. Las tarjetas estarán organizadas en las distintas columnas :

TO DO : Una tarea que todavía no ha comenzado ni esta asignada, a la espera que otras tareas sean completadas

DOING : Una tarea que se está realizando. Estará asignada a uno o varios integrantes del equipo y se podrán ir haciendo actualizaciones de su estado

DONE : Una tarea que se ha terminado, a la espera de revisión.

Revisión : Una tarea que se está comprobando, buscando errores en esta y comprobando su correcta Implementación.

Merged : Una tarea que ha sido revisada y ya está publicada y fusionada con el resto de las tareas

Captura de pantalla de un celular

Descripción generada automáticamente

# 5 - Tecnologías utilizadas

Durante el desarrollo de este proyecto hemos utilizado los siguientes programas y tecnologías:

- Draw.io como programa principal para la creación y diseño de todos los diagramas del proyecto, incluyendo el Modelo E/R, el diagrama de Clases de Modelo, los diagramas de Secuencia entre otros.  
 - Visual Studio Code como editor de Código para toda la parte de desarrollo web, incluyendo el desarrollo del HTML, del código JavaScript y de los estilos CSS.  
- DataGrip y WorkBench para la creación de la base de datos y sus Scripts .

- Kotlin como lenguaje principal para el desarrollo de la Aplicación de escritorio.

- Intellij IDEA como el IDE principal para el desarrollo de la Aplicación de Escritorio.

- Programas de control de versiones como GitKraken y GitHub Desktop y GitHub para el repositorio remoto.

Logotipo

Descripción generada automáticamente con confianza media- Trello como sistema de Planificación de tareas y responsabilidades.

# 6 - Desarrollo e Implementación Web

## Imagen de la pantalla de un celular con letras Descripción generada automáticamente con confianza bajaDiseño inicial de la web

## Descripción de la Arquitectura:

Captura de pantalla de computadora

Descripción generada automáticamente

La página web se compone :

Un encabezado o Header en el que mostraremos un menú con enlaces a nuestro repositorio, un enlace a Google Maps con la ubicación de nuestro instituto y el logo como enlace a la página principal

Un Slider de imágenes programado con JavaScript cambiante con un temporizador y flechas funcionales.

Iconos con una leve descripción de nuestra empresa

Una imagen SVG que sirve de enlace a nuestro formulario.

Un Footer con enlaces a nuestros GitHub personales , nuestro logo Principal y un enlace a nuestra página de Políticas de privacidad

Interfaz de usuario gráfica

Descripción generada automáticamente

Un formulario para que el usuario pueda pedir una cita con nosotros con facilidad.

Todo los campos del formulario están validados con JavaScript.

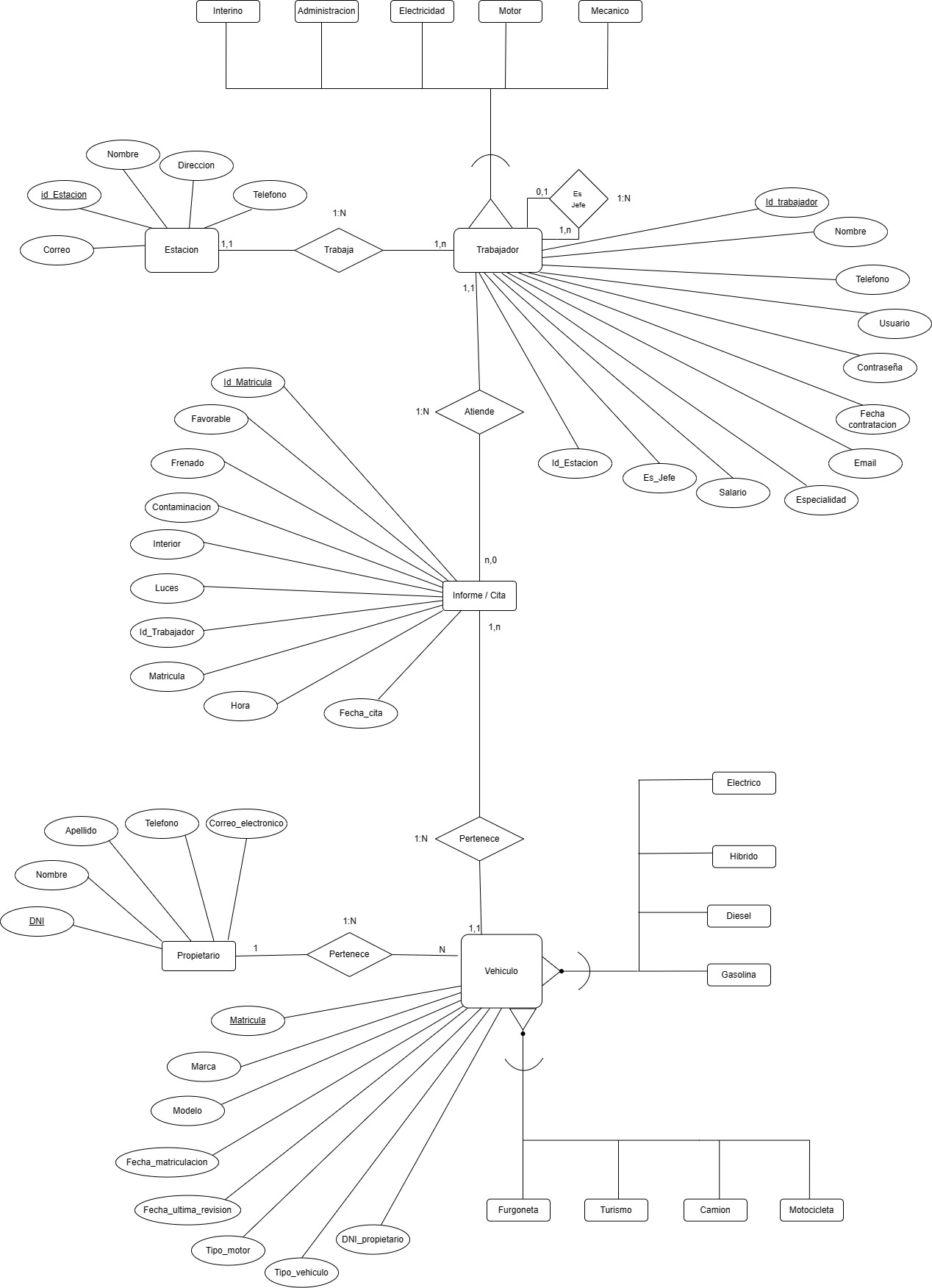
Los datos del formulario serán enviados por correo electrónico para los trabajadores puedan crear una cita

# 7 - Diseño de bases de datos

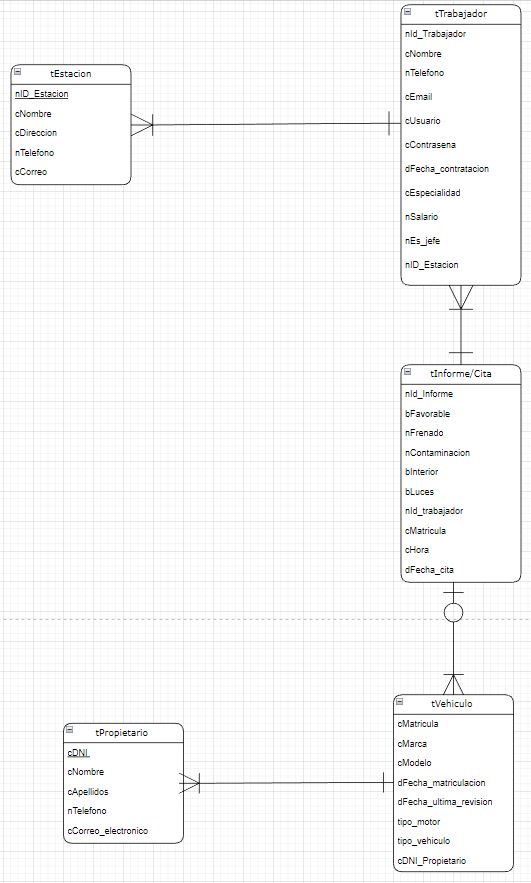
## Modelo entidad relación:

Se ha creado la base de datos y el modelo entidad relación según el enunciado propuesto y se han establecido las siguientes restricciones:

-Un trabajador pertenece a una única estación, y en una estación trabajan varios trabajadores.  
-Un trabajador tiene un jefe y un jefe es jefe de varios trabajadores.  
-Un trabajador atiende varias citas/informes y una cita/informe solo puede ser atendida por un único trabajador.  
-Un informe/cita solo pertenece a un único coche, y un coche puede tener varios informes/citas.  
-un vehículo pertenece a un único propietario y un propietario le pueden pertenecer varios vehículos.



## Modelo físico



## Tablas Implementadas en DataGrip

Imagen que contiene Diagrama

Descripción generada automáticamente

## Scripts

### Script para la creación de la base de datos:

Texto

Descripción generada automáticamenteSe han credo las sentencias SQL para la creación de la tablas con sus respectivos campos y restricciones siguiendo el modelo entidad relación creado, añadiendo restricciones a los campos en los campos que sean necesarios,

Script para la creación de procedimientos, disparadores y eventos:

#### Texto Descripción generada automáticamente 1-Procedimiento que liste los trabajadores por estación:

Texto

Descripción generada automáticamente2-Disparador para Controlar inspecciones, guardando la información previa y la información que se ha modificado.

#### Texto Descripción generada automáticamente3-Evento para el borrado de citas cada dos meses

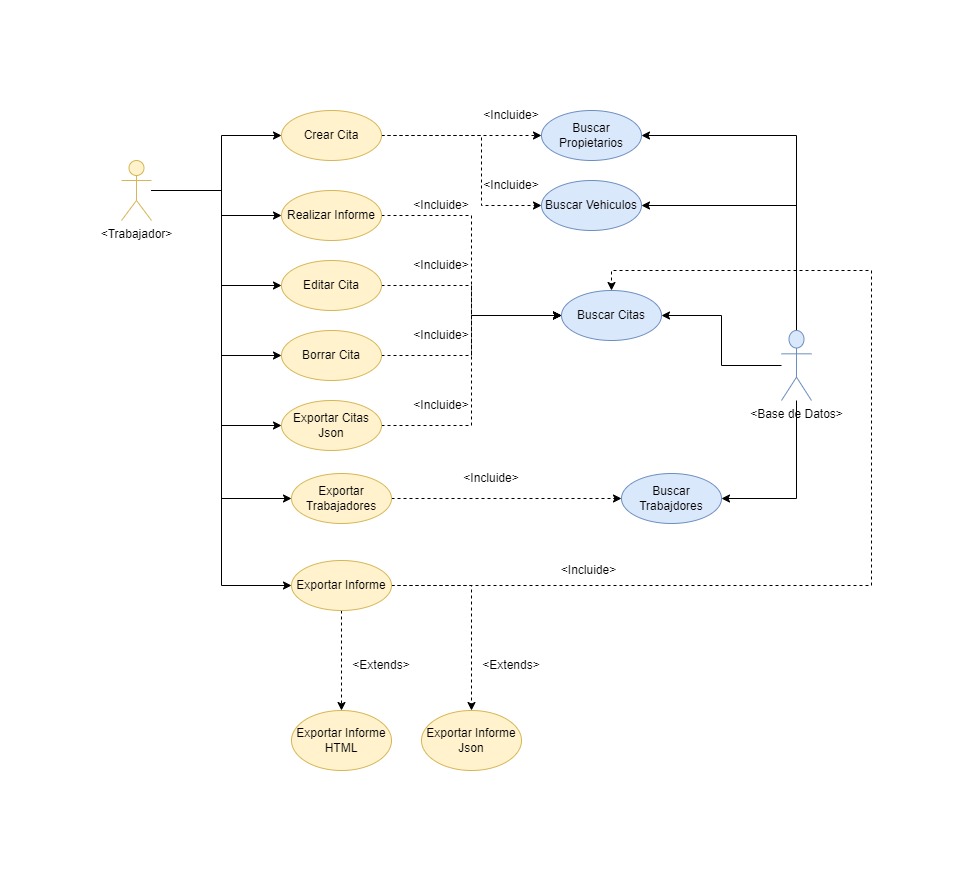
#### 4-Trigger adicional para el cálculo del salario según la especialidad del trabajador

Texto

Descripción generada automáticamente

# 8 - Diagramas y desarrollo del programa

## Diagrama de caso de usos

Diagrama de casos de uso de un trabajador :

En este diagrama podemos comprobar como un trabajador que usa la aplicación podrá :

* Crear una nueva cita : Para crear una nueva cita el trabajador necesitara los datos de los propietarios y sus trabajadores. Estos serán proporcionados por nuestra base de datos
* Realizar un informe de una cita : El trabajador rellenará los datos del informe y los guardará en nuestra base de datos
* Editar una cita : El trabajador solo podrá cambiar la hora, el día y el trabajador asignado de la cita. En una cita nunca se modificará el vehículo asignado.
* Borrar una cita : El trabajador borrará una cita y su informe (aunque no su historial) de nuestra base de datos
* Exportar todas las citas a JSON : Todas las citas de nuestra base de datos , incluyendo la información de la cita, el propietario , el vehículo y el nombre y correo electrónico del trabajador asignado, serán exportados en formato JSON.
* Exportar un informe concreto a HTML o JSON : El trabajador exportará en formato HTML o JSON toda la información del informe, incluyendo la cita en la que se hizo, el vehículo y propietario y el nombre y correo electrónico del trabajador asignado.
* Exportar todos los trabajadores en formato CSV : Se exportará toda la información de todos los trabajadores de la Base de datos , incluyendo los campos privados excepto la contraseña.

## 

## Diagrama de secuencia

Los diagramas de secuencia nos sirven para mostrar las llamadas que ocurrirán dentro de nuestra aplicación y como se va recorriendo por las distintas clases y funciones de nuestro programa desde que el trabajador inicia una acción hasta que ha sido completada o se termina.

### Diagrama de Secuencia Nueva CitaDiagrama para crear una nueva cita

### Escala de tiempo Descripción generada automáticamenteDiagrama para actualizar el trabajador de una cita

### Diagrama Descripción generada automáticamente con confianza bajaDiagrama para imprimir un informe

### Diagrama Descripción generada automáticamenteDiagrama de clases de modelo

9 - Aplicación de Escritorio Gestión de Citas:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTexto

Descripción generada automáticamente  
Para el desarrollo de la Aplicación hemos utilizado la Arquitectura MVVM (Model View ViewModel o Modelo Vista VistaModelo ) que nos permite organizar las clases y funciones del código de una forma controlada y lógica, haciendo que el código esté lo menos acoplado posible, permitiendo su fácil ampliación y mantenimiento en el tiempo.

Como podemos ver todas las clases están organizadas por funcionalidad en su respectivo paquete, lo que mejora la navegabilidad por el código y el fácil acceso a todas las clases y funciones de este igual que su fácil lectura para todas las personas que tengan que mantener este código en un futuro.

## Controladores :

Texto

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamenteLos controladores se encargarán de gestionar la interfaz gráfica y las partes con las que interactuara el usuario. Todo Controlador pertenece a una vista (un archivo .fxml) que se encarga de definir la parte visual mientras que el controlador gestiona las acciones de la vista junto a su funcionalidad interna.

Texto

Descripción generada automáticamente

## ModelViews :

Los modelViews se encargan de interconectar la interfaz gráfica y la lógica interna del programa, a la vez que se encarga de controlar los cambios que se mostrarán en la interfaz gráfica.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamentePor ejemplo, en este modelView tenemos unas cuantas Listas Observables que estarán unidas con un Binding a las distintas tablas, columnas y selectores de la Interfaz gráfica. Al ser observables permiten que cada vez que haya cambios en las listas la interfaz reacciona a estos sin necesidad de ninguna llamada. También tenemos un objeto MainState() que será una representación de la interfaz gráfica hecha clase.

Esta clase se encarga de mostrar los cambios del viewmodel al controlador a través de bindings.

Texto

Descripción generada automáticamente con confianza mediaTambién , el viewModel al ser la interconexión entre los distintos puntos de la aplicación es el sitio ideal para poder validar los datos a través de distintos validators.

Este validador comprobara que el trabajador no ha dejado ningún campo vacío. Gracias al uso de los Result (Railway oriented programming) podemos mostrar al usuario los errores deseados.

## Repositorios

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza mediaLos repositorios se encargan de gestionar las funcionalidades más importantes de la aplicación como la comunicación directa con la Base de datos. Cada repositorio estará asociado un tipo de Modelo concreto . Todos los repositorios implementarán una interfaz (un contrato) que obligará a los repositorios a implementar las funcionalidades deseadas. Si un repositorio concreto tiene funcionalidades extras hará otra interfaz mas especializada.

Texto

Descripción generada automáticamenteEsta interfaz obligara a implementar sus funciones. Debido a que nuestra aplicación solo gestiona citas y sus informes y no otros modelos esta interfaz solo obligara a implementar las funciones comunes. Esta interfaz hace uso de genéricos para T – Tipo , ID – id y ERROR – Error especializado.

Captura de pantalla de computadora

Descripción generada automáticamenteEsta interfaz obliga a implementar 3 funciones de gestión que solo usarán las citas.

Todos los repositorios tendrán una conexión directa con la Base de datos

## Servicios

Texto

Descripción generada automáticamenteLos servicios se encargarán de dar funcionalidades extras.

Tenemos dos tipos de servicios :

### Storages

Texto

Descripción generada automáticamenteSe encargarán de guardar objeto y clases de nuestro código en distintos archivos

Texto

Descripción generada automáticamente

### DatabaseManger

Texto

Descripción generada automáticamenteEl databaseManager se encargará de conectar nuestra aplicación con nuestra base de datos

En el caso de nuestro databaseManager aparte tendremos funciones que se encargarán de crear las tablas si estás no existiesen en la base de datos.

## Railway oriented programming

Texto

Descripción generada automáticamenteEl railway oriented programming nos permite programar nuestro código como si fuese un camino con distintas intersecciones. Siempre que el programa funcione a la perfección decimos que vamos por el camino correcto o Happy Path ,pero si durante el camino ocurre algún error, cambiaremos de camino y devolveremos al usuario el error que ha ocurrido en el momento de desviarse del camino.

En esta función, si no se encuentra el Informe con la id buscada se devolverá el error de que no existe (Camino Incorrecto).

Si lo encuentra se devolverá el valor borrado (Camino Correcto o Happy Path).

## Inyección de Dependencias con Koin

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteEn todo el proyecto hemos necesitado múltiples dependencias para nuestras Clases. Para poder hacerlo de una forma más cómoda hemos utilizado Koin, debido a que es fácil de usar y nos permite inyectar las dependencias de una forma rápida y sin mucha complicación, ya que solo necesitaremos declarar nuestras dependencias en un modulo

## Texto Descripción generada automáticamenteDependencias:

* Result : Aunque Kotlin nos proporciona un Result de base, esta dependencia nos proporciona el uso de errores propios y personalizados.
* Logger : Nos permite poder depurar mejor nuestro código, ya que en el momento de ejecución podremos ver todos nuestros loggers según se van activando.
* Open : Nos permite abrir enlaces en nuestra interfaz gráfica.
* Mockito : Dependencia para poder testear con dobles
* Conector MariaDB y mysql : Dependencias necesarias para poder conectar nuestra aplicación con nuestra Base de datos.
* Gson : Dependencia para exportar e importar con comodidad los JSON. Hemos utilizado Gson en vez de Moshi por un problema con el modulo de javaFx, ya que Moshi-Kotlin y Moshi normal hacen uso de dependencias comunes y no pueden funcionar a la vez de una forma comoda.
* Koin : Inyector de dependencias comodo y muy fácil de usar.

## Interfaz Grafica

Para la creación de nuestra interfaz gráfica hemos usado la aplicación de SceneBuilder y JavaFX, ya que nos encontramos cómodos con estos y existe una gran cantidad de documentación en internet para arreglar posibles errores y añadir diversas funcionalidades.

### Interfaz de usuario gráfica, Aplicación Descripción generada automáticamenteVista Principal

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTenemos una tabla con todos las citas , los datos del propietario y vehículo al que está asignado al igual que el resultado del informe. Al pulsa una de las citas se rellenan los datos de la parte inferior con toda la información.