Anexo III

Especificación de Análisis y Diseño

Trabajo de fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

VNiVERSiDAD
D SALAMANCA

Marzo de 2023

Autor

Rodrigo de la Calle Alonso

Tutor/a

Álvaro Lozano Murciego

Tabla de contenido

Tabla de Ilustraciones

1 Análisis

1.1 Introducción

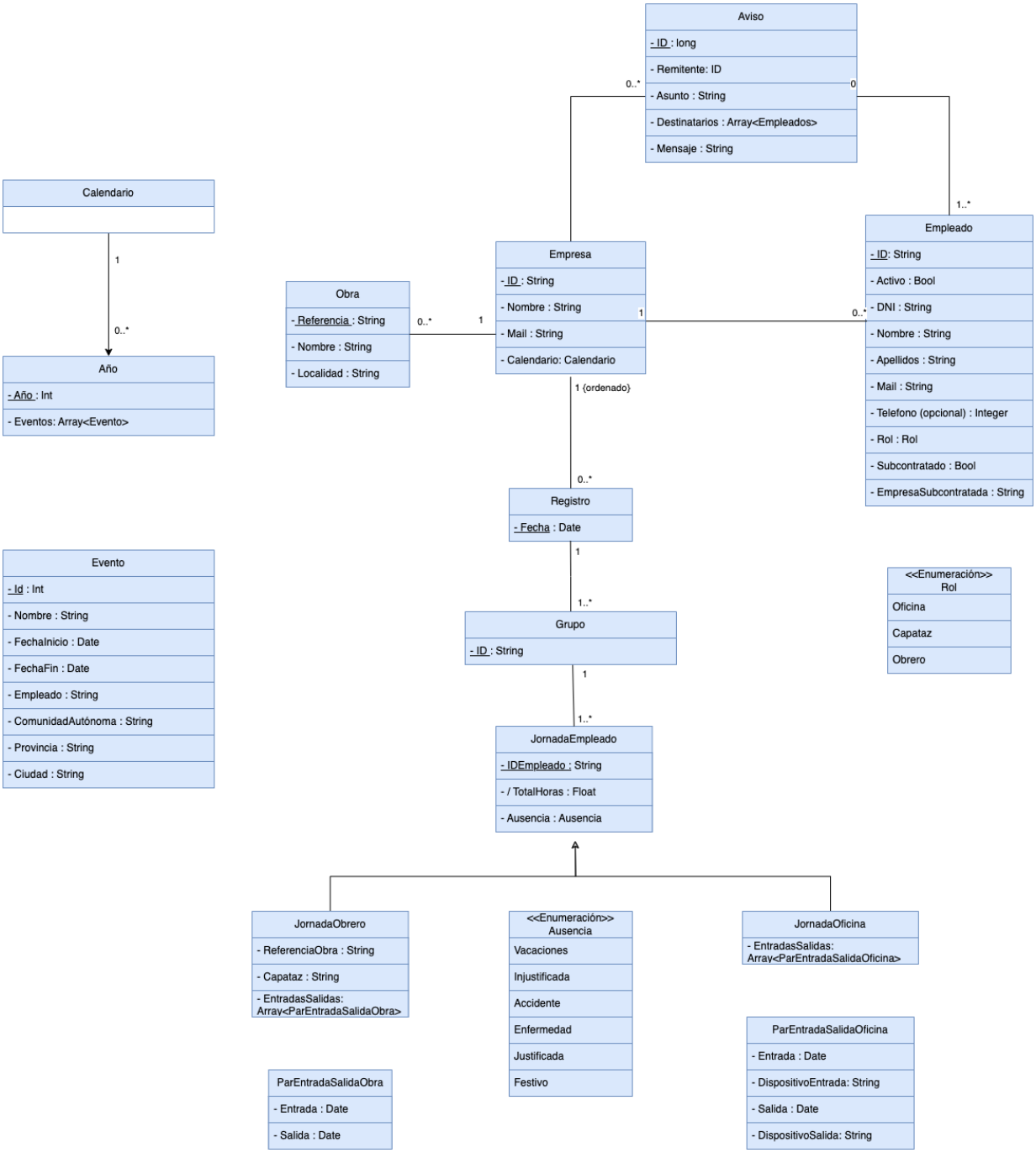
En esta parte del anexo se detallará el análisis del sistema. Se mostrará la arquitectura del sistema y los diagramas de secuencia de los casos de uso de la aplicación. También se podrá ver el modelo del domino del sistema.

Para la creación de los diagramas del sistema se ha empleado la herramienta Visual Paradigm [1] y draw.io [2].

1.2 Modelo del dominio

En este apartado se pueden ver las distintas clases y atributos que forman parte del proyecto.

Ilustración 1 Diagrama de Clases



A continuación, se muestra una breve explicación de lo que representa cada clase:

- **Empresa:** representa al usuario que se ha registrado por sí mismo en la plataforma identificándose como una empresa.

- **Empleado:** representa a los usuarios que un usuario empresa registra en el software y que estarán asociados a la empresa que los ha registrado.

- **Obra:** representa a las construcciones en las que trabaja la empresa.

- **Aviso:** representa a las notificaciones que se pueden intercambiar de forma interna entre empleados y su empresa.

- **Registro:** representa a los documentos diarios donde se almacenan las jornadas.

- **Grupo:** representa la forma de agrupar las jornadas por cuadrillas de trabajo o por oficina.

- **JornadaEmpleado:** representa la asistencia o ausencia de un empleado para un día junto con las horas totales que ha trabajado. En el caso de ser un obrero o capataz almacenará un atributo con la referencia de la obra en la que trabaja, otro con el capataz que le ha marcado la asistencia y un array de ParEntradaSalidaObra. En el caso de ser un empleado con rol de oficina, almacenará un array de ParEntradaSalidaOficina.

- **ParEntradaSalidaObra:** representa una entrada y salida que ha tenido un empleado durante su jornada.

- **ParEntradaSalidaOficina:** representa una entrada y salida que ha tenido un empleado durante su jornada y en que dispositivos ha fichado.

- **Calendario:** representa una colección de años.

- **Año:** representa una colección que contiene todos los eventos y festivos de un año.

- **Evento:** representa un evento junto con su fecha y localidad.

1.3 Vista de Arquitectura

En este apartado se muestran las diferentes interfaces, controles y entidades que hay en el sistema, así como las relaciones que tienen. Para facilitar la comprensión y el orden se han introducido los elementos en paquetes. En la última foto se puede ver las relaciones entre los paquetes.

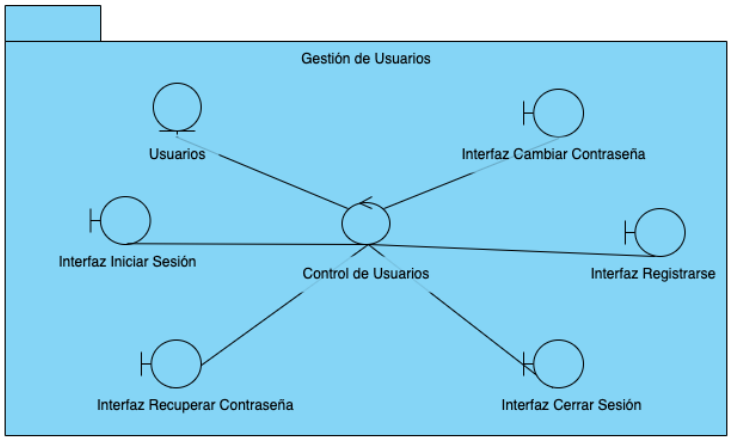


Ilustración 2 Arquitectura Gestión de Usuarios

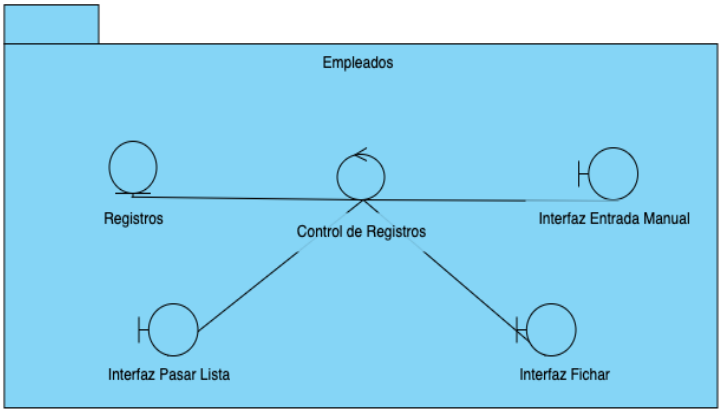


Ilustración 3 Arquitectura Empleados

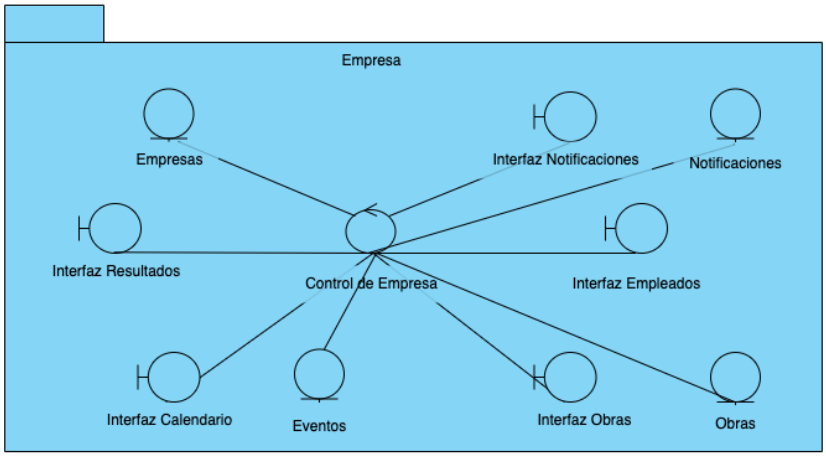


Ilustración 4 Arquitectura Empresa

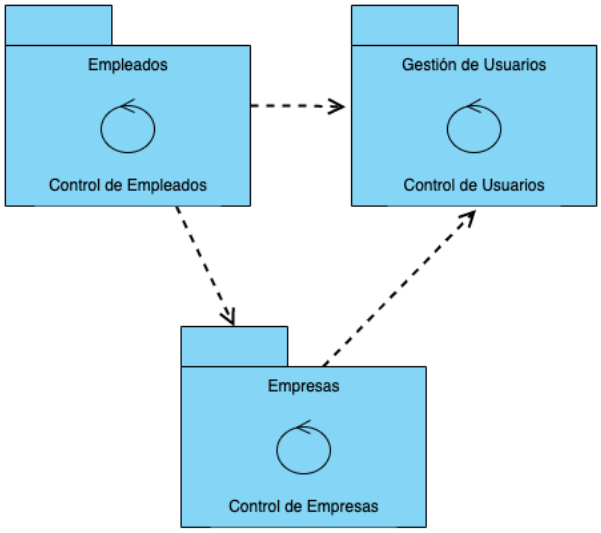


Ilustración 5 Arquitectura Relaciones

1.4 Diagramas de secuencia

Con estos diagramas podremos ver de forma ordenada las interacciones entre los elementos del sistema mediante el paso de mensajes.

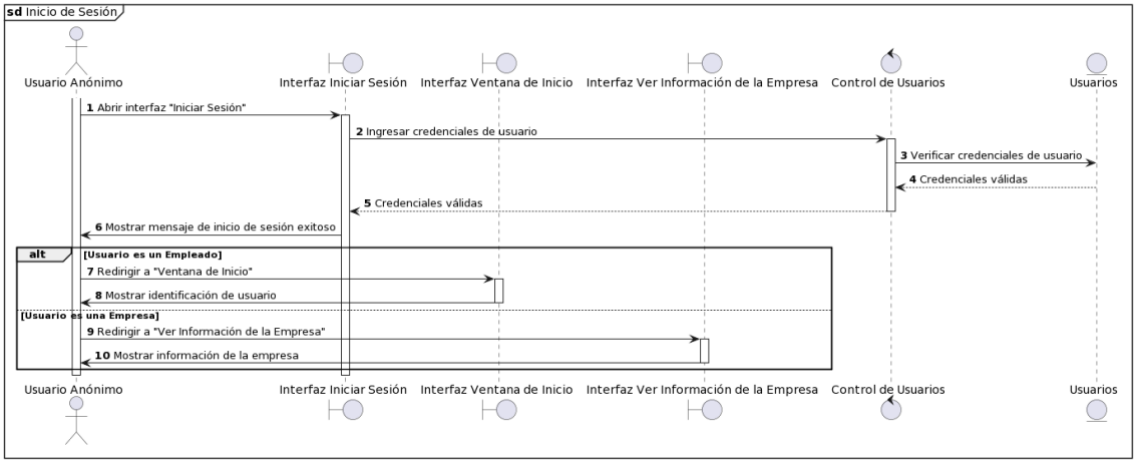


Ilustración 6 Análisis, SD Iniciar Sesión

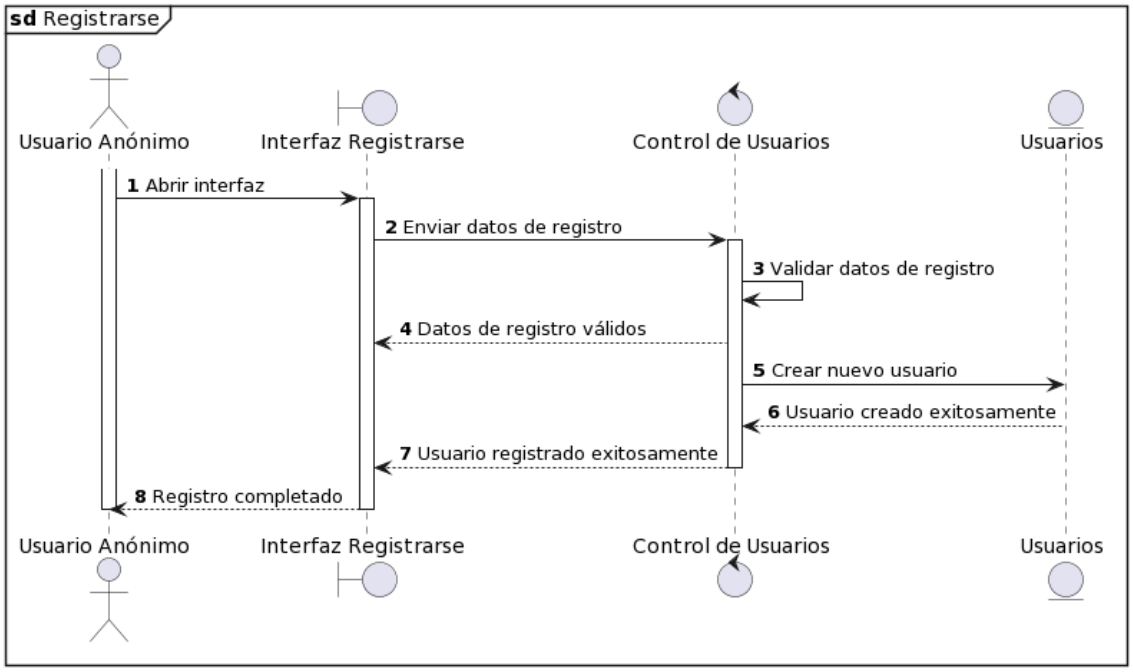


Ilustración 7 Análisis, SD Registrarse

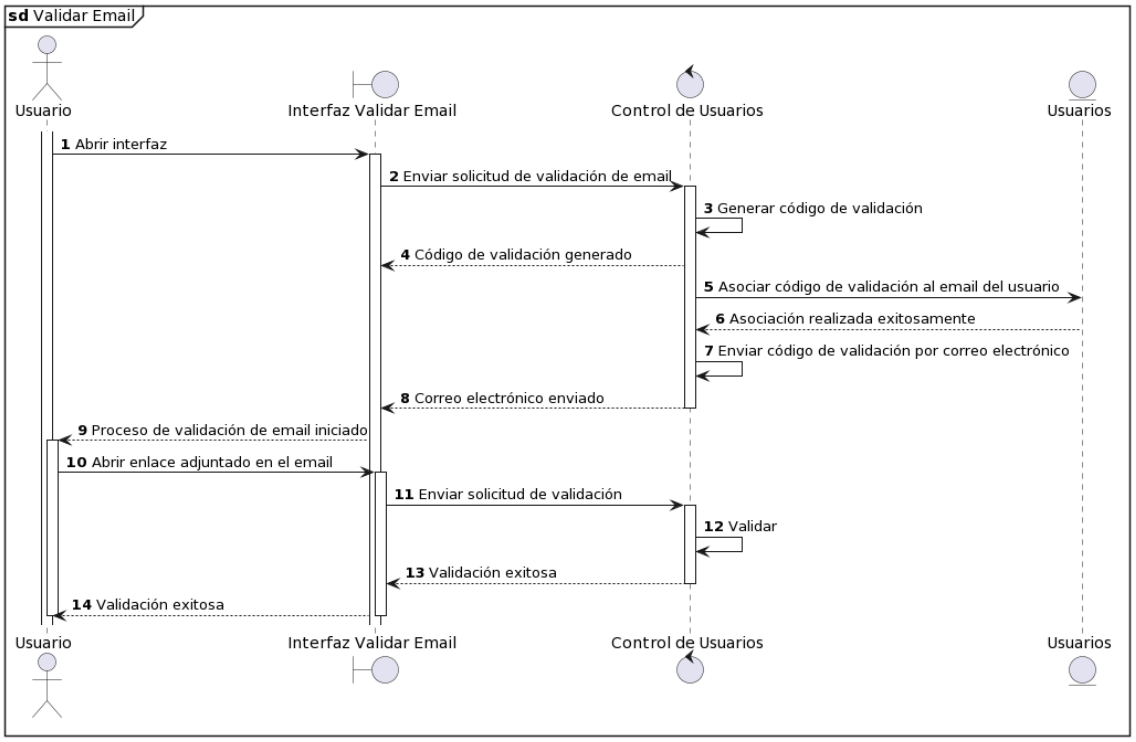


Ilustración 8 Análisis, SD Validar Email

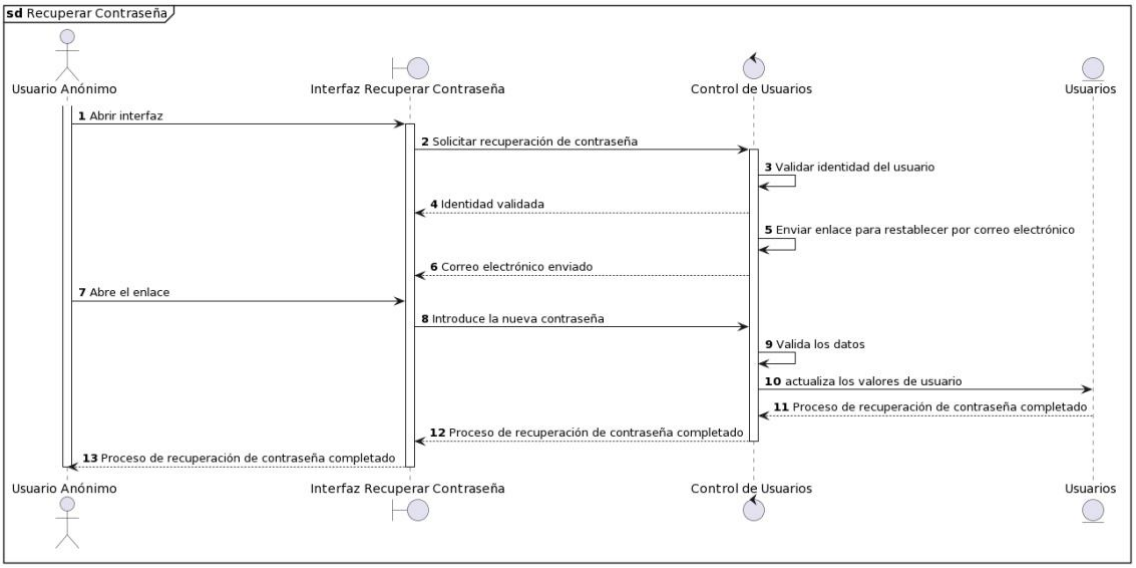


Ilustración 9 Análisis, SD Recuperar Contraseña

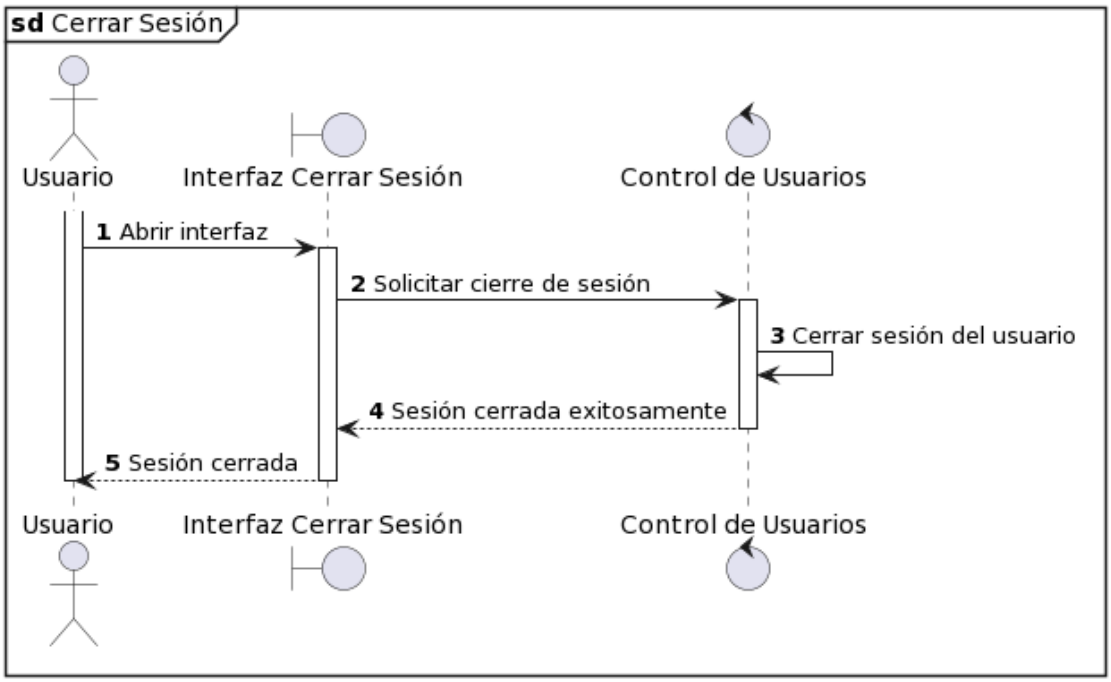


Ilustración 10 Análisis, SD Cerrar Sesión

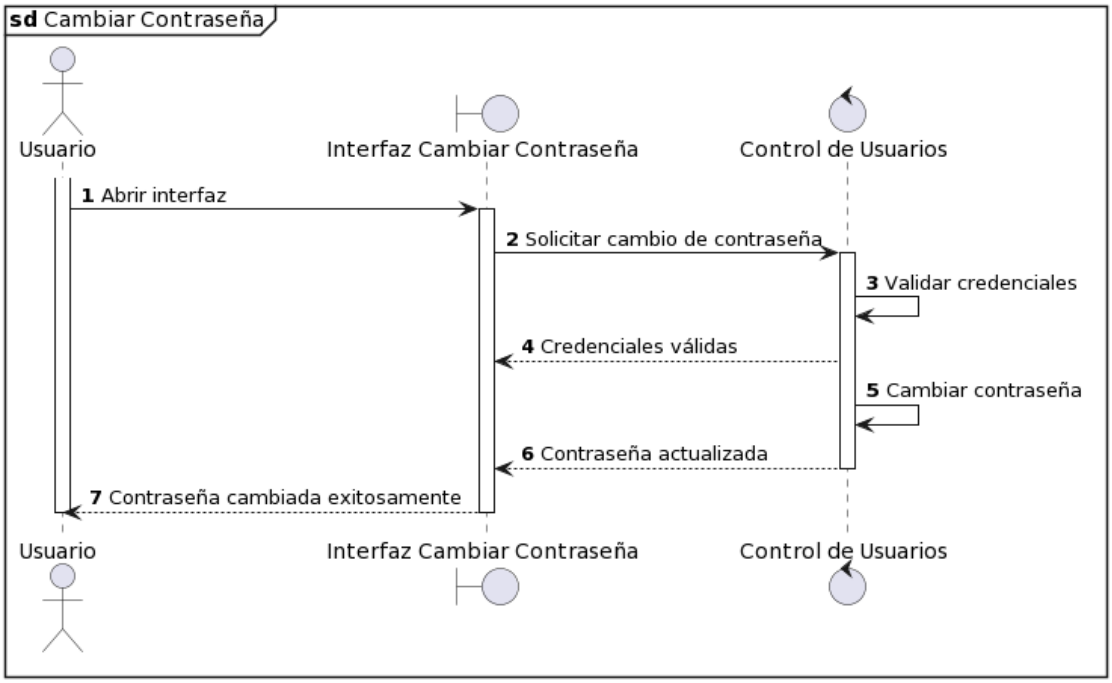


Ilustración 11 Análisis, SD Cambiar Contraseña

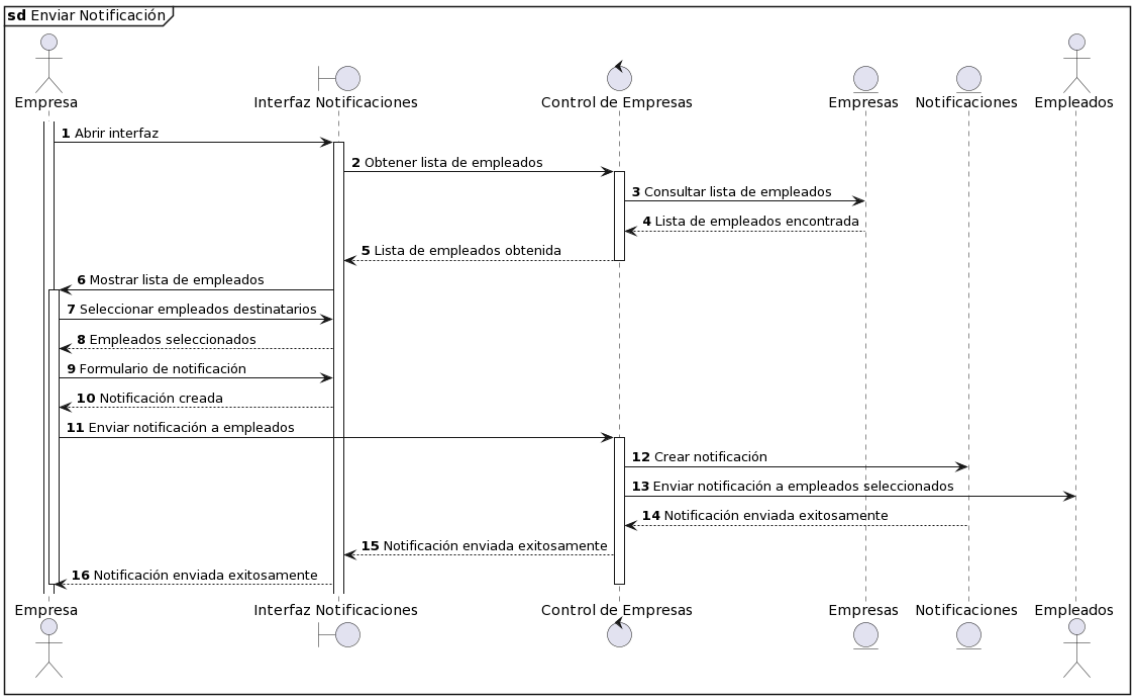


Ilustración 12 Análisis, SD Enviar Notificación

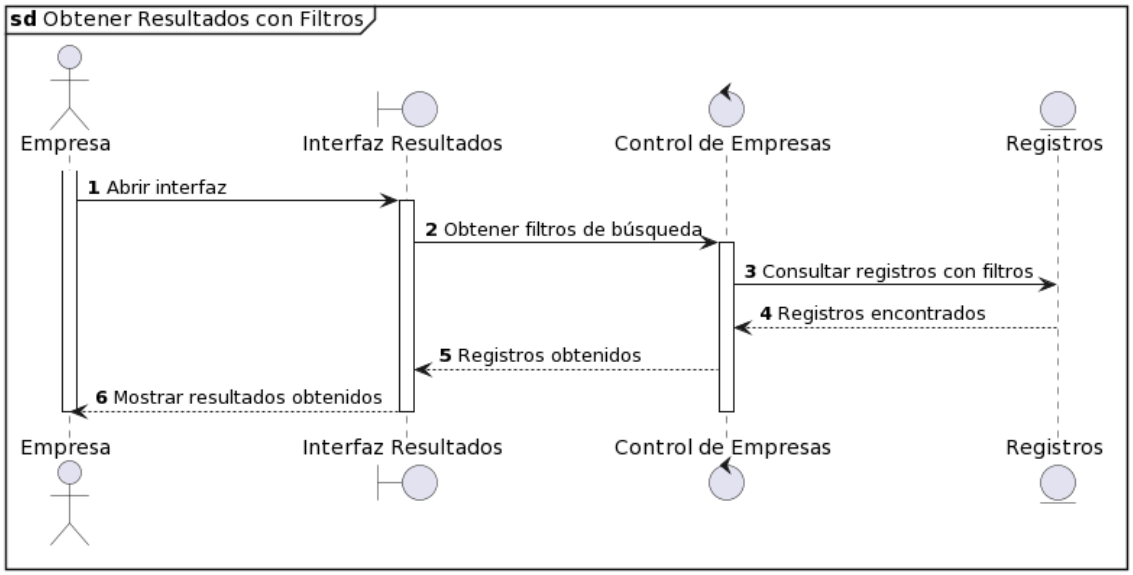


Ilustración 13 Análisis, SD Obtener Resultados con Filtros

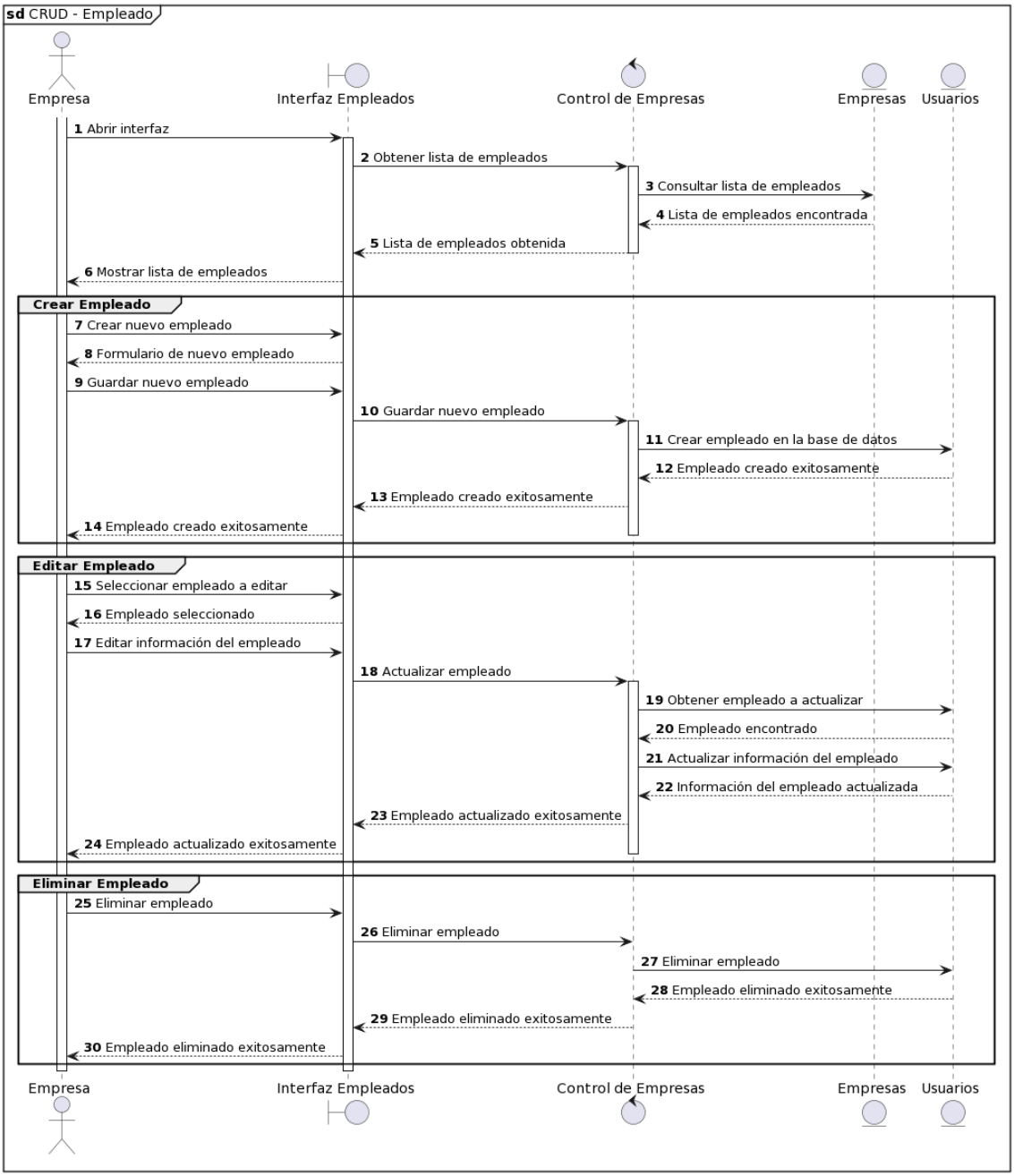


Ilustración 14 Análisis, SD CRUD - Empleados

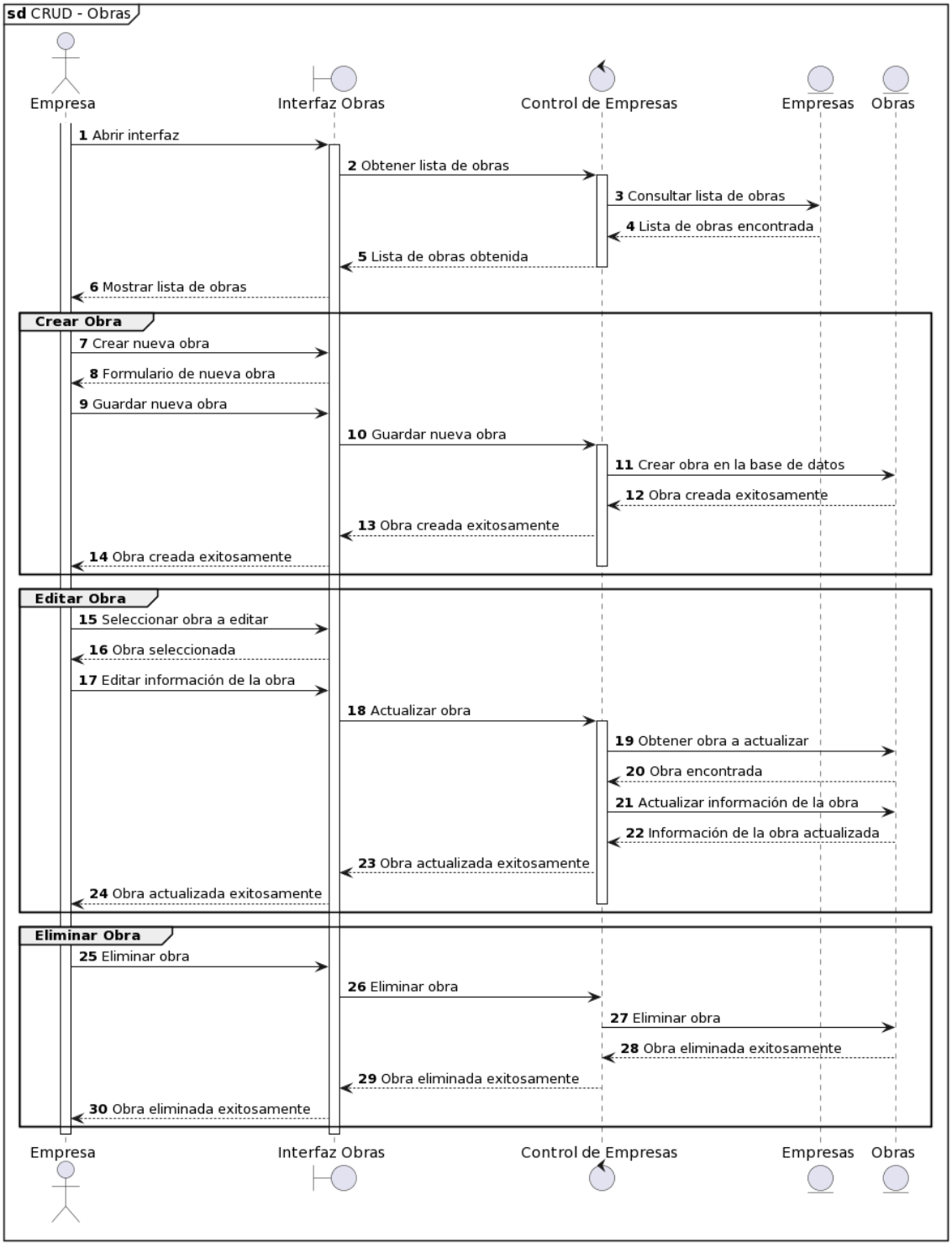


Ilustración 15 Análisis, SD CRUD - Obras

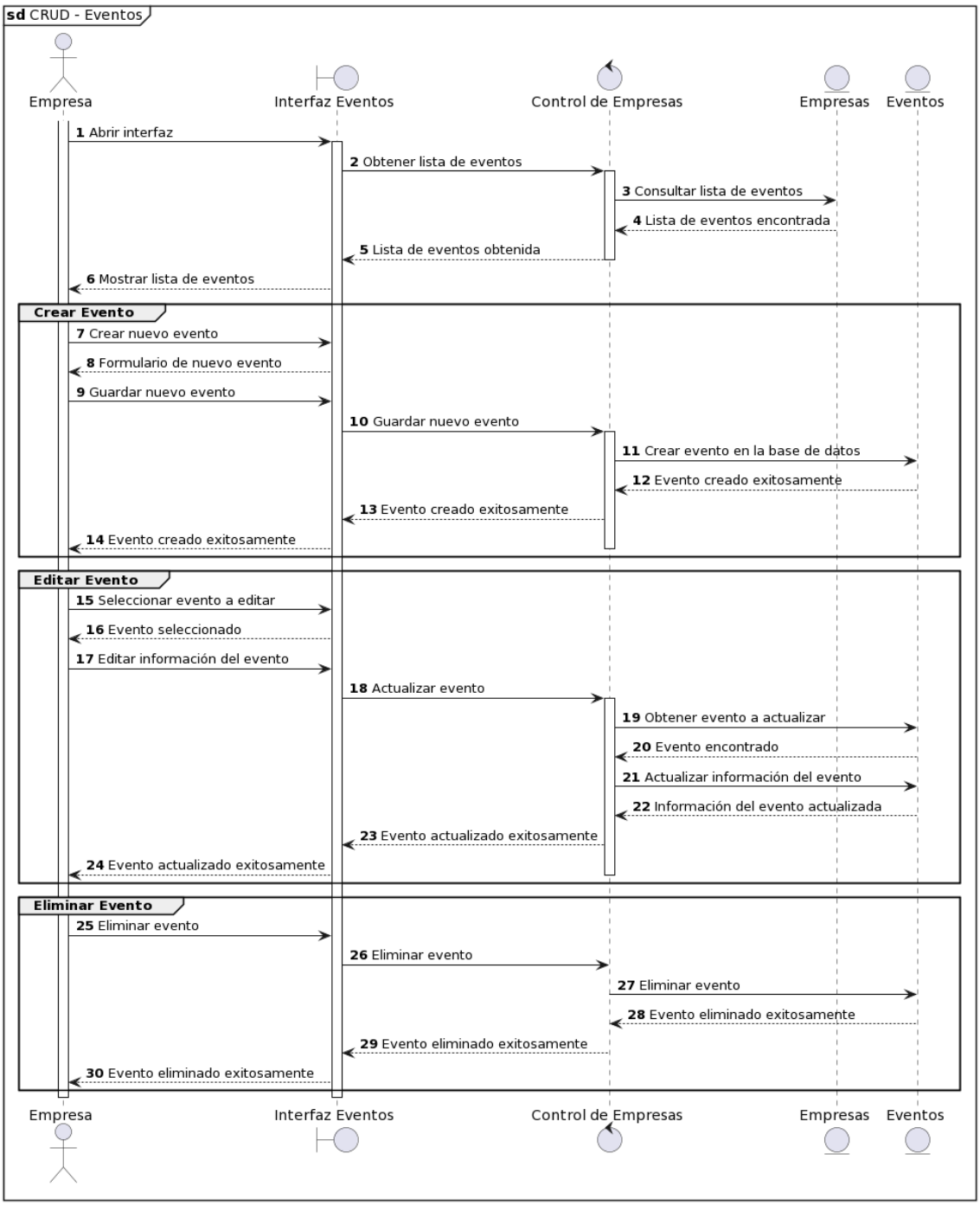


Ilustración 16 Análisis, SD CRUD - Eventos

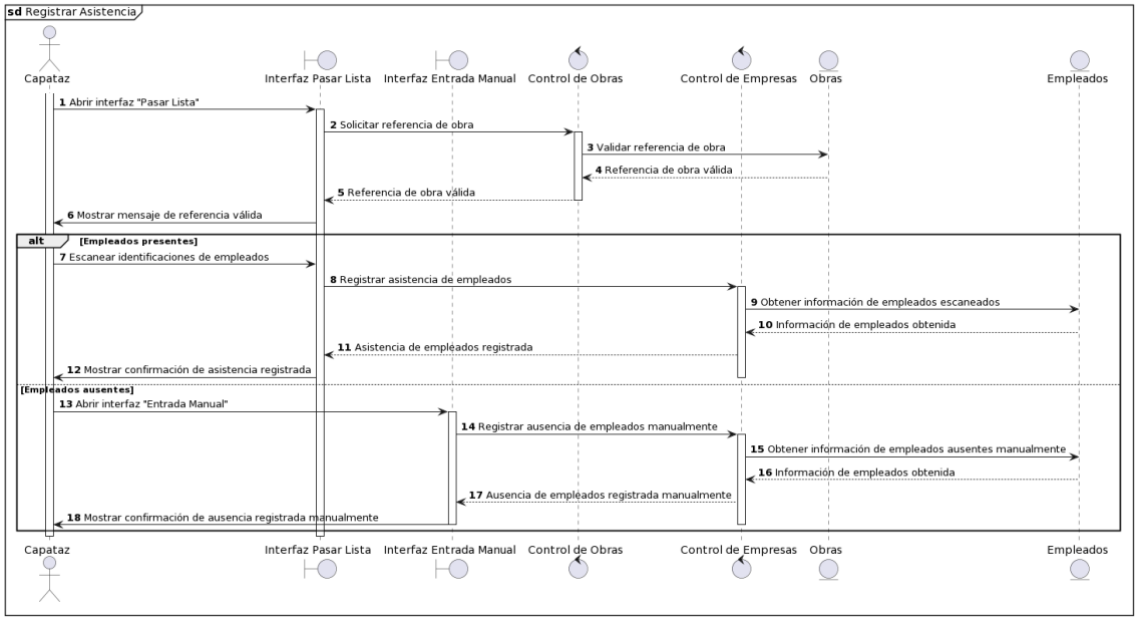


Ilustración 17 Análisis, SD Registrar Asistencia

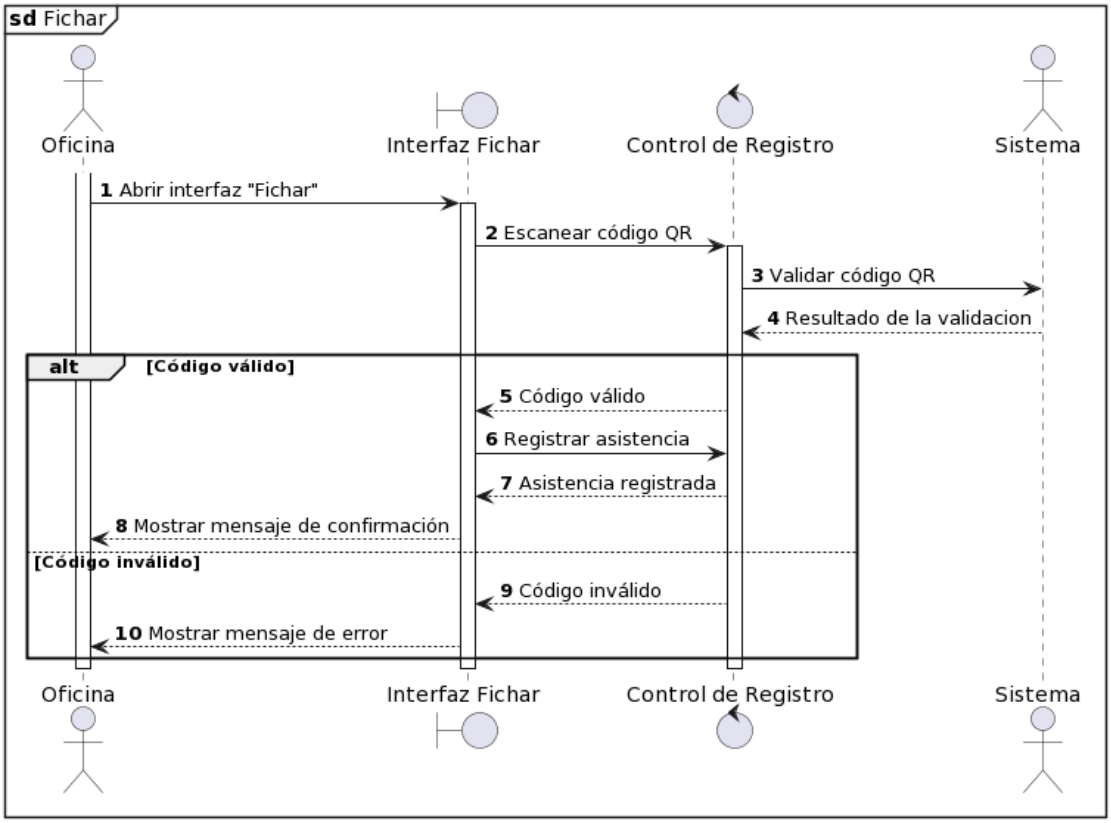


Ilustración 18 Análisis, SD Fichar

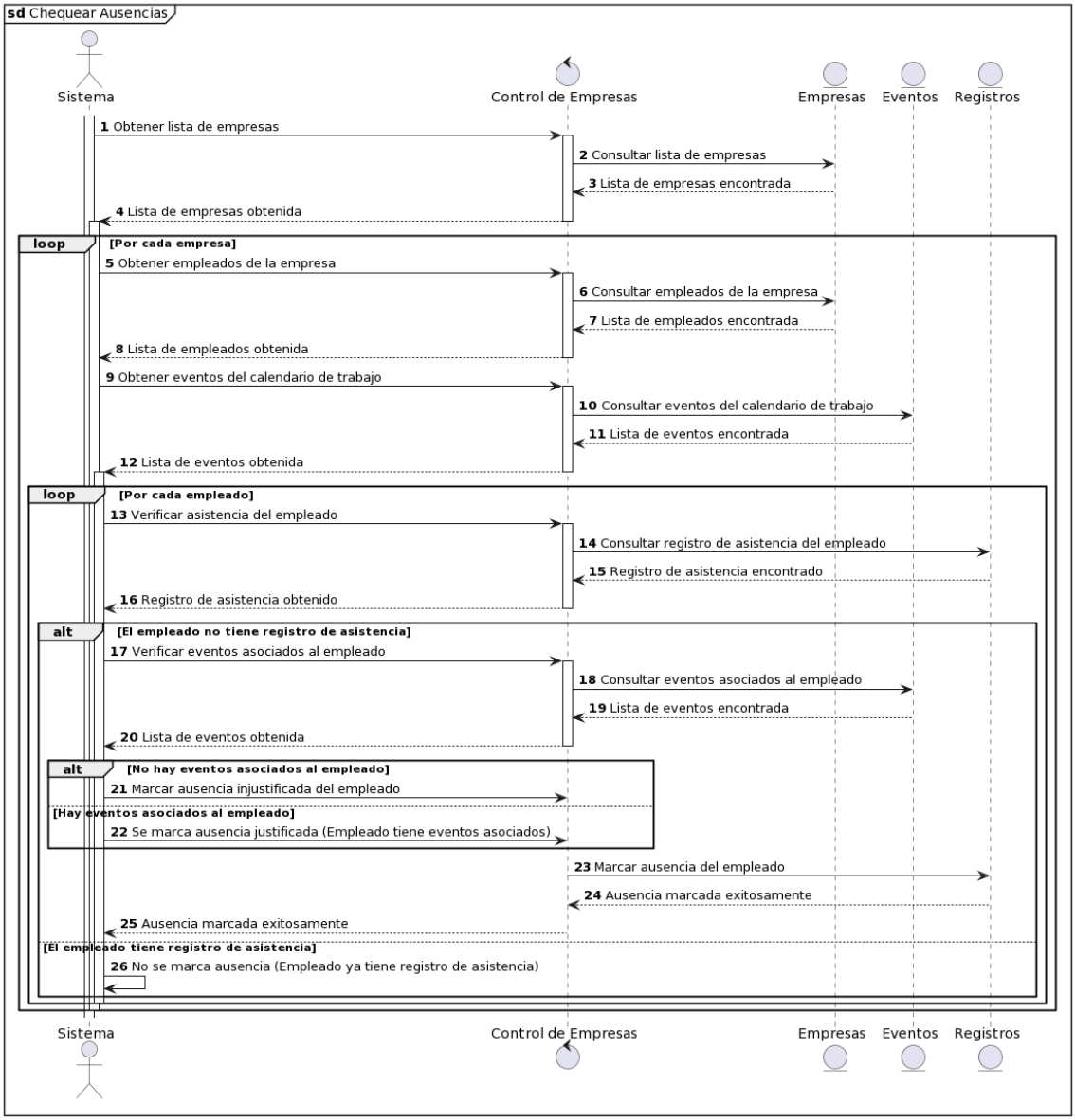


Ilustración 19 Análisis, SD Chequear Ausencias

2 Diseño

2.1 Introducción

En ese apartado se detallará el diseño final del sistema. Se definirá el modelo del dominio final, la arquitectura del diseño del sistema, así como los diagramas de secuencia que conforman la aplicación.

2.2 Patrones Arquitectónicos

En la parte de la aplicación del *front-end* se ha empleado el patrón *Model-View-ViewModel* (MVVM). La finalidad principal de este patrón es desacoplar lo máximo la interfaz y la lógica de negocio de la aplicación.

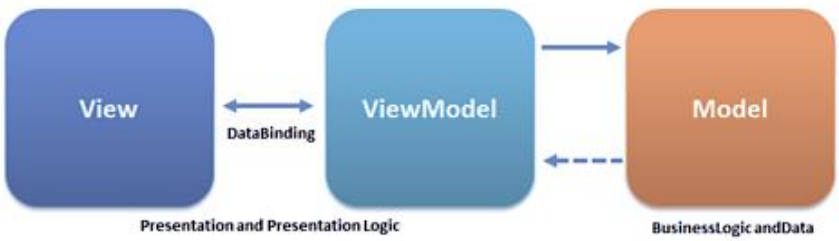


Ilustración 20 Patrón MVVM [3]

Las tres partes son:

- **El modelo (*Model*)** representa la capa de datos. Contiene la información, pero nunca las acciones o servicios que la manipulan. En ningún caso tiene dependencia alguna con la vista.
- **La vista (*View*)** representa la información a través de los elementos visuales que la componen. Contienen comportamientos, eventos y enlaces a datos que, en cierta manera, necesitan tener conocimiento del modelo subyacente.
- **El *ViewModel*** es un actor intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz. La comunicación entre la vista y el *ViewModel* se realiza por medio los enlaces de datos (*binders*).

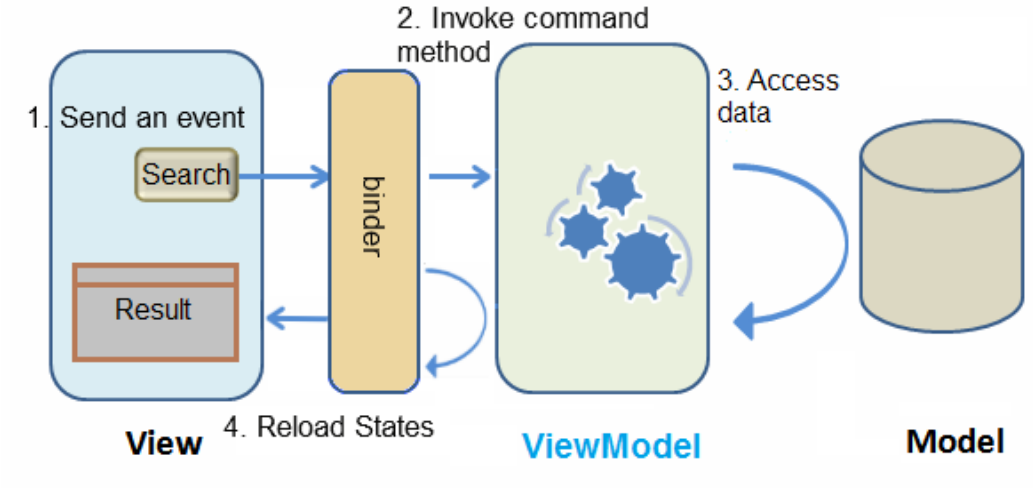


Ilustración 21 Patrón MVVM Detalle [4]

2.3 Subsistemas de Diseño

Se divide el sistema en dos partes, *front-end* que contendrá toda la lógica de la aplicación que interactúa con él cliente, y *back-end* que tratará la gestión de las faltas automáticas. Ambos subsistemas están basados en servicios de Firebase [5].

En todos los paquetes de la gestión del *front-end* se usa de Firestore [6] como base de datos. El paquete de gestión de usuarios también utiliza Firebase Authentication [7], para la verificación de las credenciales de los usuarios, y Firebase Cloud Messaging [8] para enviar notificaciones *push* a usuarios.

La gestión del *back-end* usa Cloud Functions [9], tanto para disparadores de la base de datos como para peticiones http con las que se gestionan el chequeo de ausencias.

El paquete Dispositivos contiene la lógica de los dispositivos usados para fichar en las oficinas, estos utilizarán RealTime Database [9] para almacenar su información.

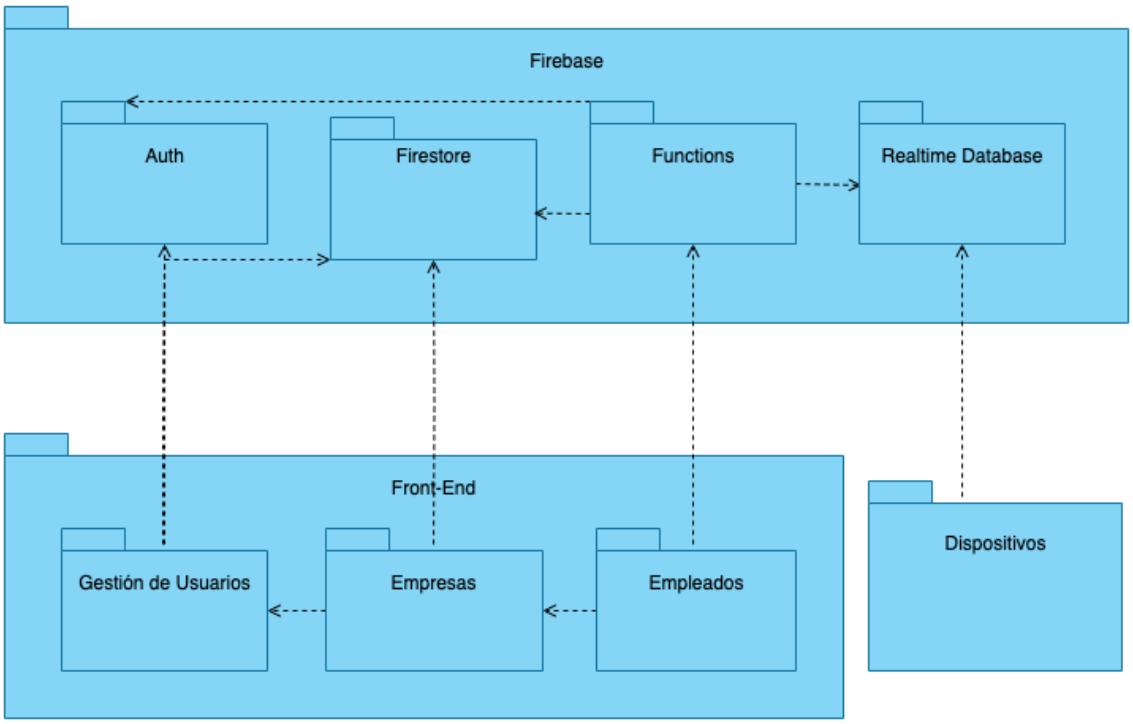


Ilustración 22 Subsistemas de diseño

2.3.1 Modelo del Dominio

En esta sección, se presenta el diagrama de clases con todos sus métodos en la Ilustración 23.

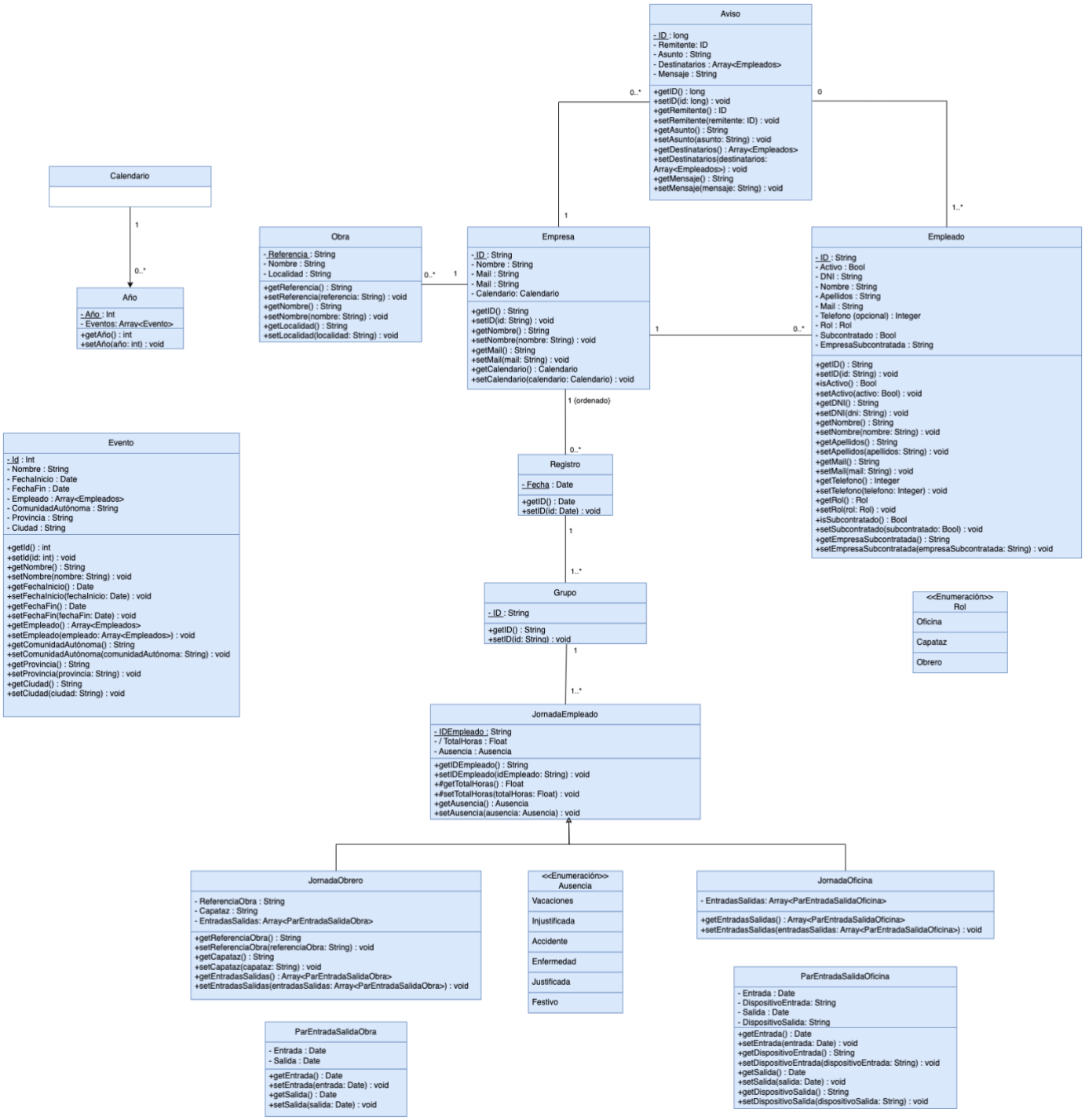


Ilustración 23 Diagrama de Clases Con Métodos

2.3.2 Vista de Arquitectura

Para evitar la dependencia entre el modelo de datos y la vista, se han utilizado *stores* de Pinia [10] donde se centralizan todos los datos y los métodos que enlazan estos datos con las Firestore Database.

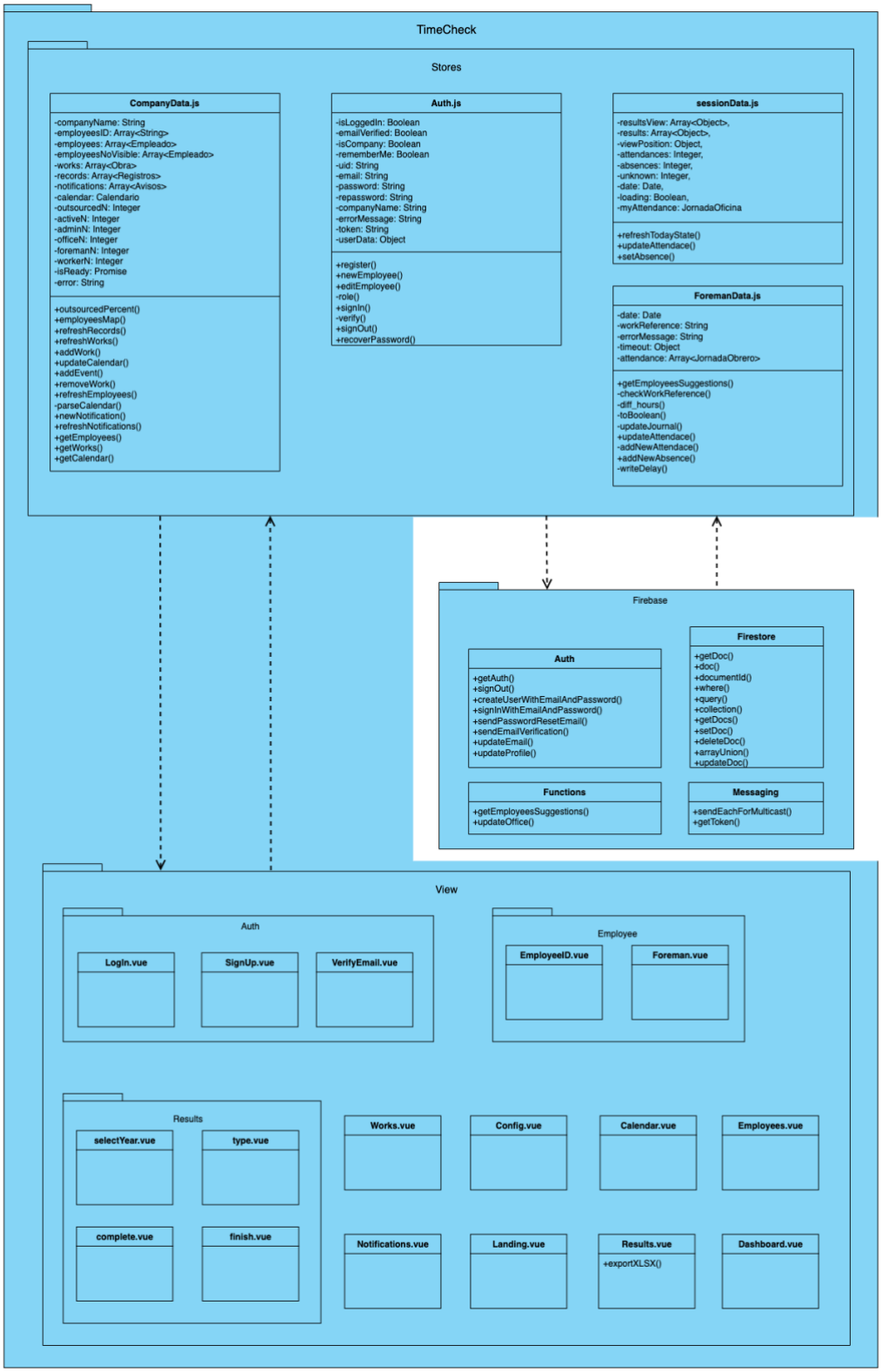


Ilustración 24 Diseño: Arquitectura

La arquitectura del subsistema de validación para oficinas se realiza mediante el dispositivo M5Stick-CPlus y se corresponde con el siguiente esquema (Ilustración 25):

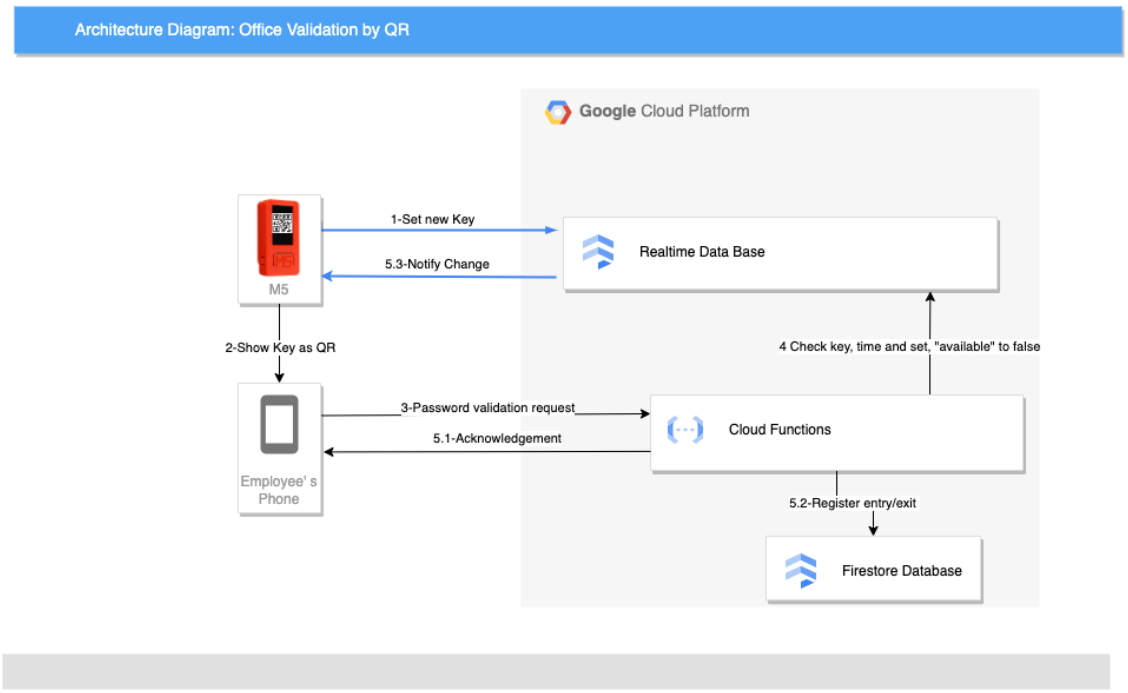


Ilustración 25 Arquitectura M5

Primero, el dispositivo establece una nueva clave en la base de datos y la muestra por pantalla en forma de código QR. Después, el empleado lo escanea con su aplicación móvil y esta llama a una Cloud Function para que sea validada. La Cloud Function, valida el código, lo marca a “No Disponible”, actualiza el estado del empleado en Firestore y lo devuelve al empleado. El dispositivo detecta que la clave ya no está disponible y genera una nueva.

2.3.3 Diagramas de Secuencia

En esta sección se encuentran los diagramas de secuencia usando todos los componentes definitivos, descritos anteriormente.

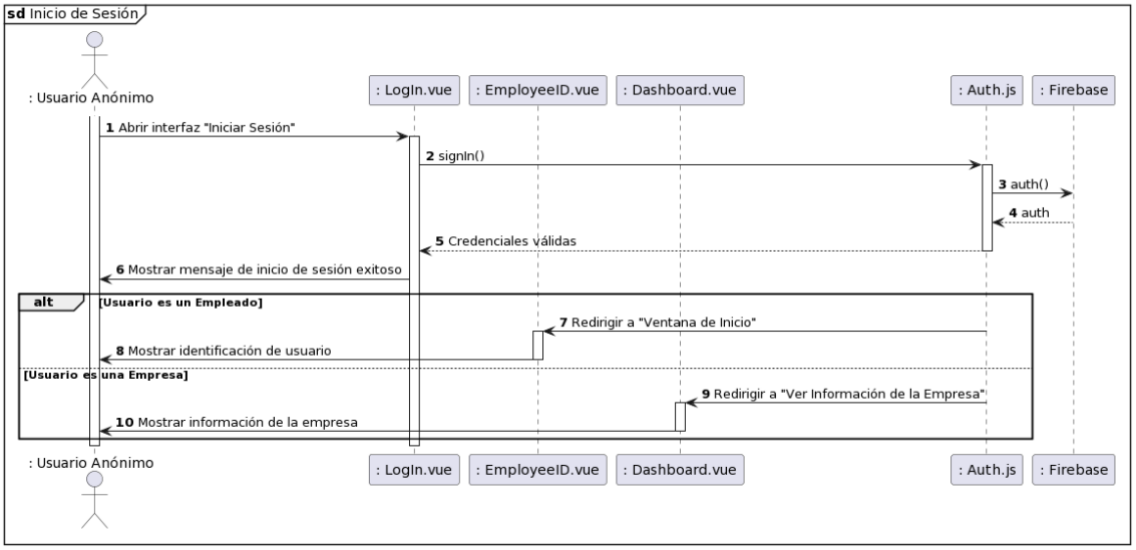


Ilustración 26 Diseño, SD Iniciar Sesión

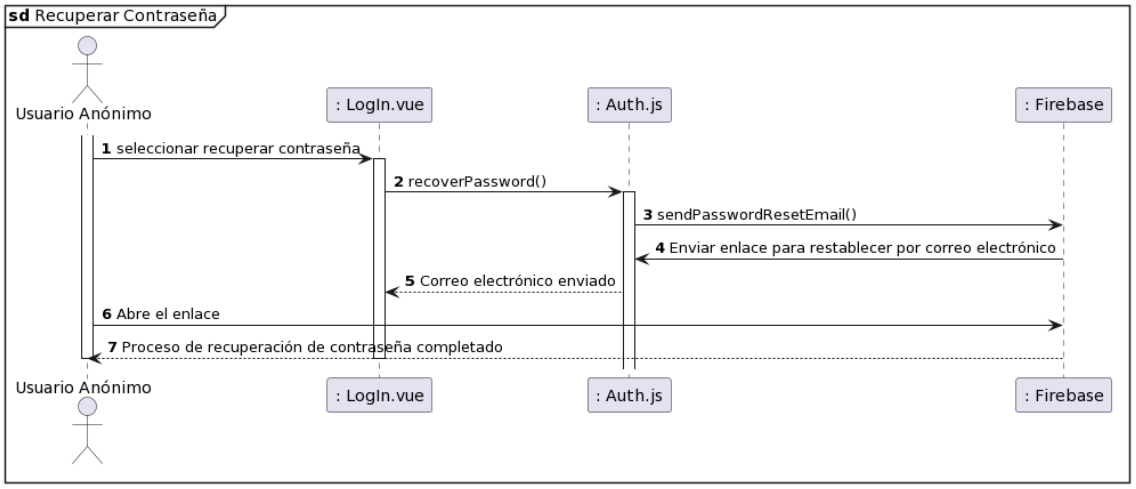


Ilustración 27 Diseño, SD Recuperar Contraseña

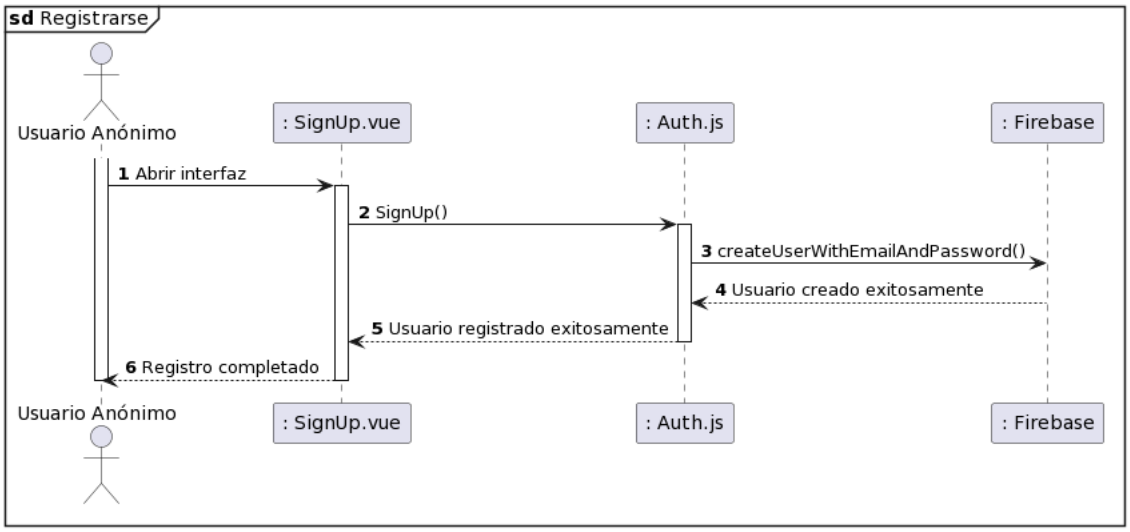


Ilustración 28 Diseño, SD Registrarse

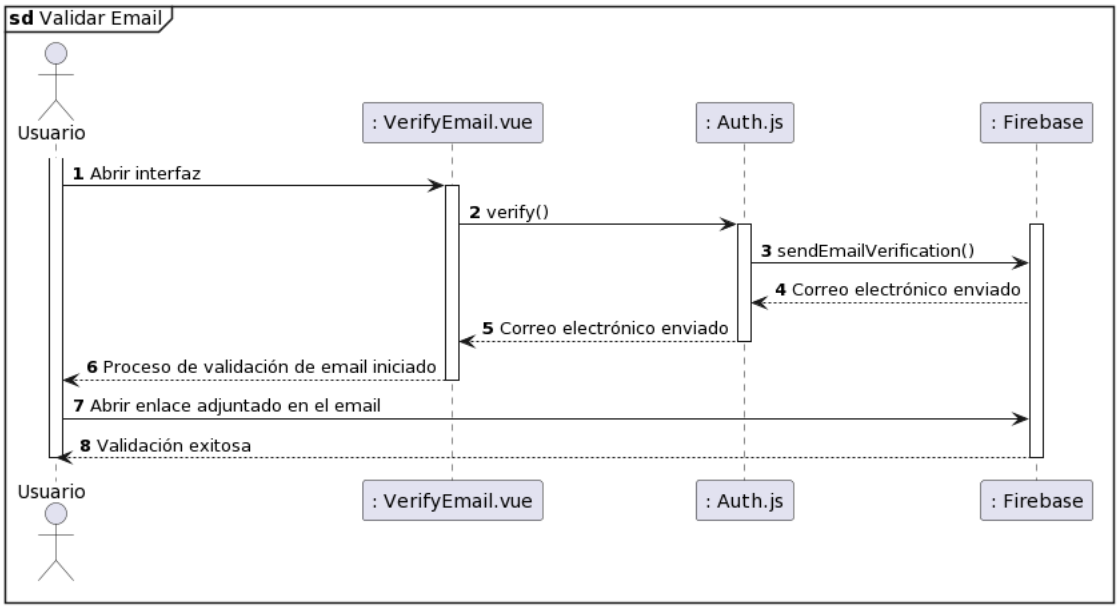


Ilustración 29 Validar Email

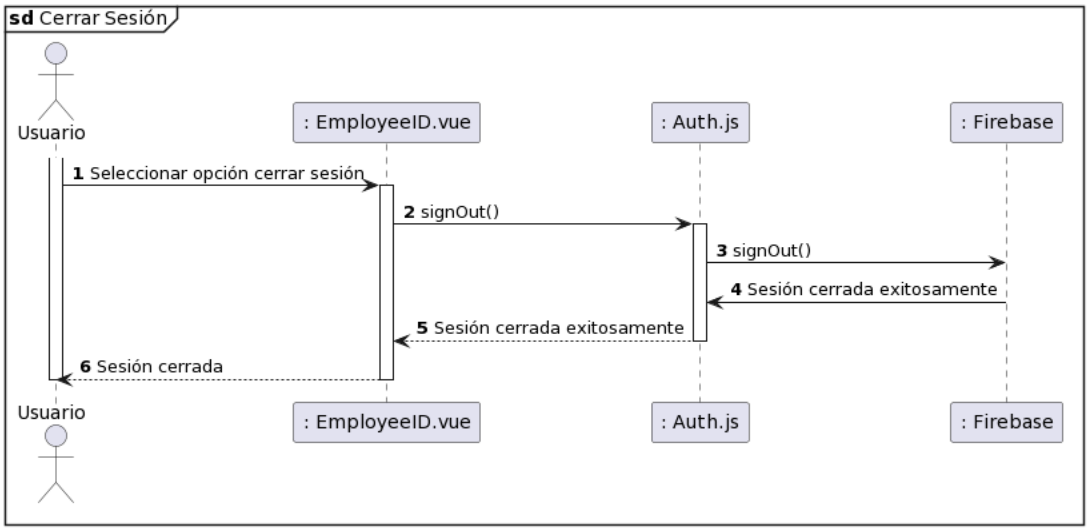


Ilustración 30 Diseño, SD Cerrar Sesión

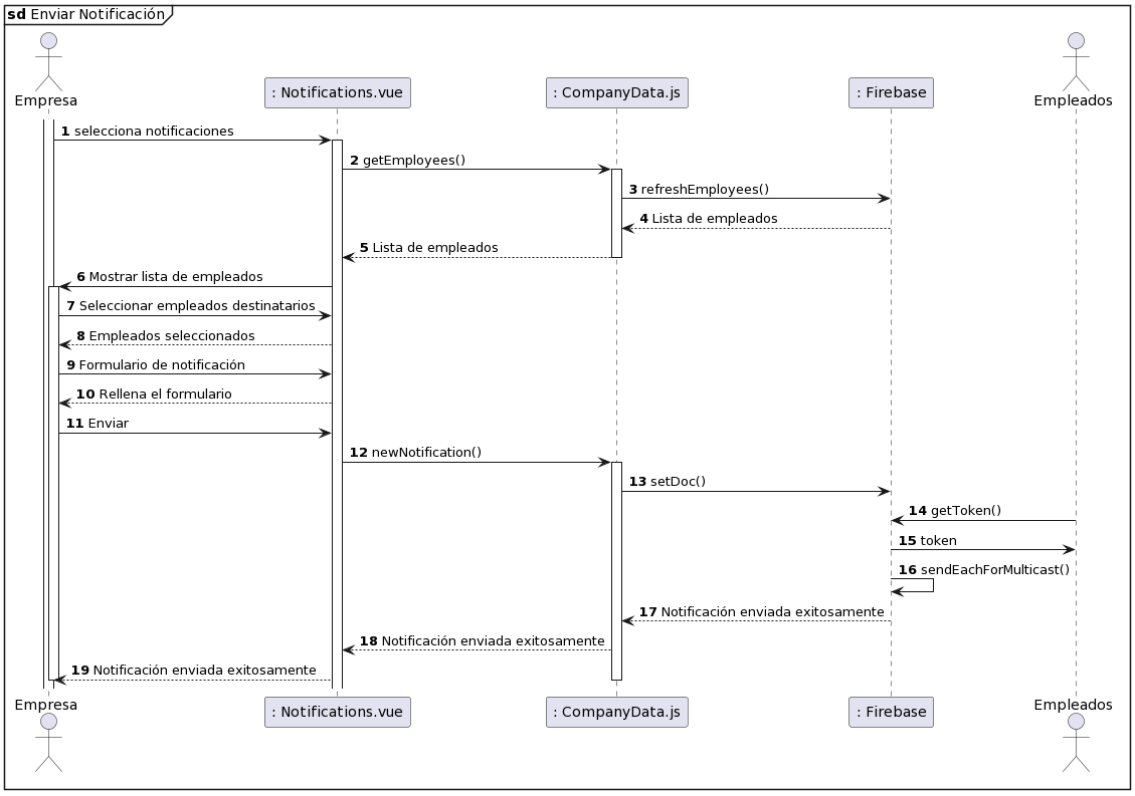


Ilustración 31 Diseño, SD Enviar Notificación

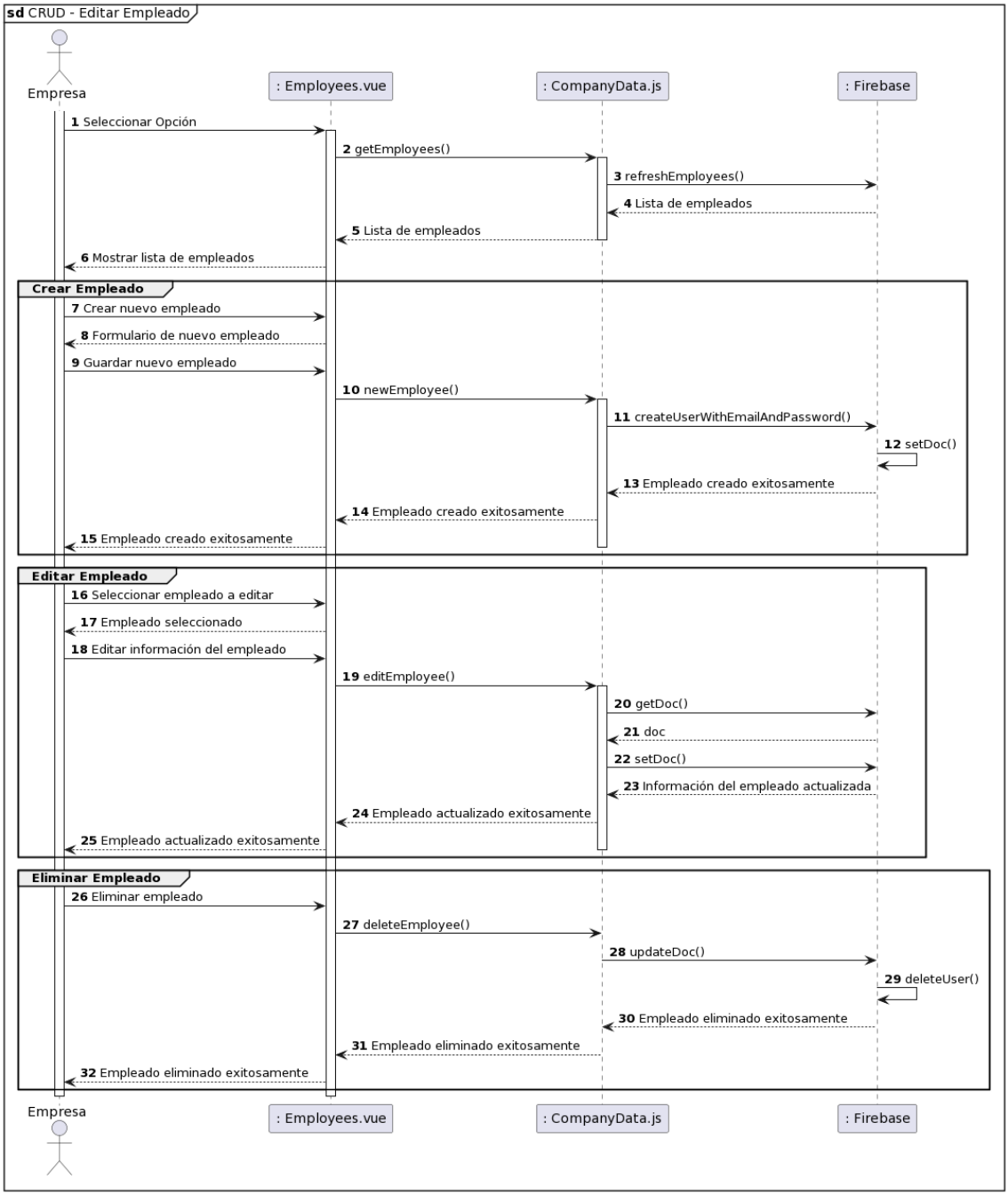


Ilustración 32 Diseño, SD CRUD - Empleado

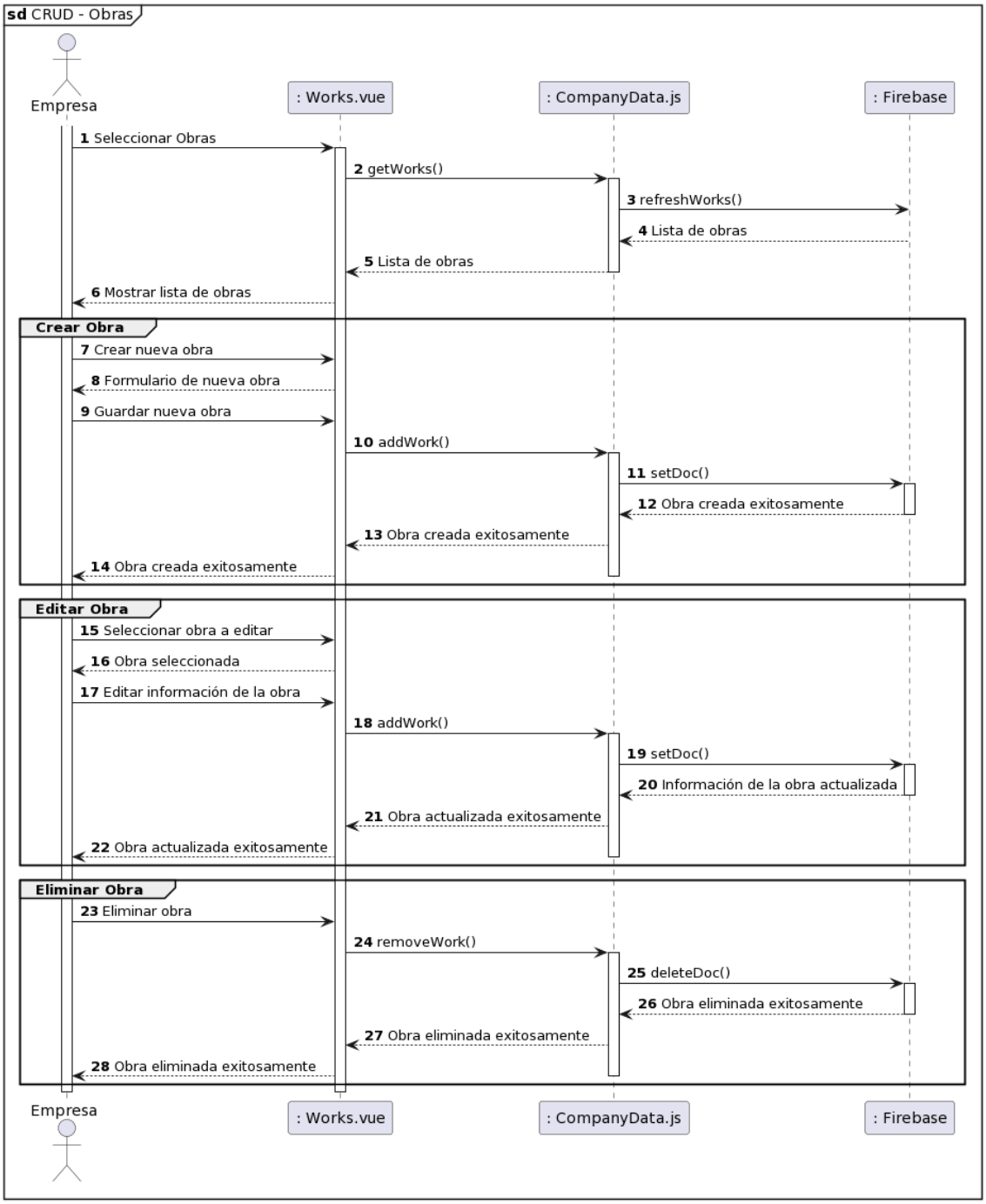


Ilustración 33 Diseño, SD CRUD - Obra

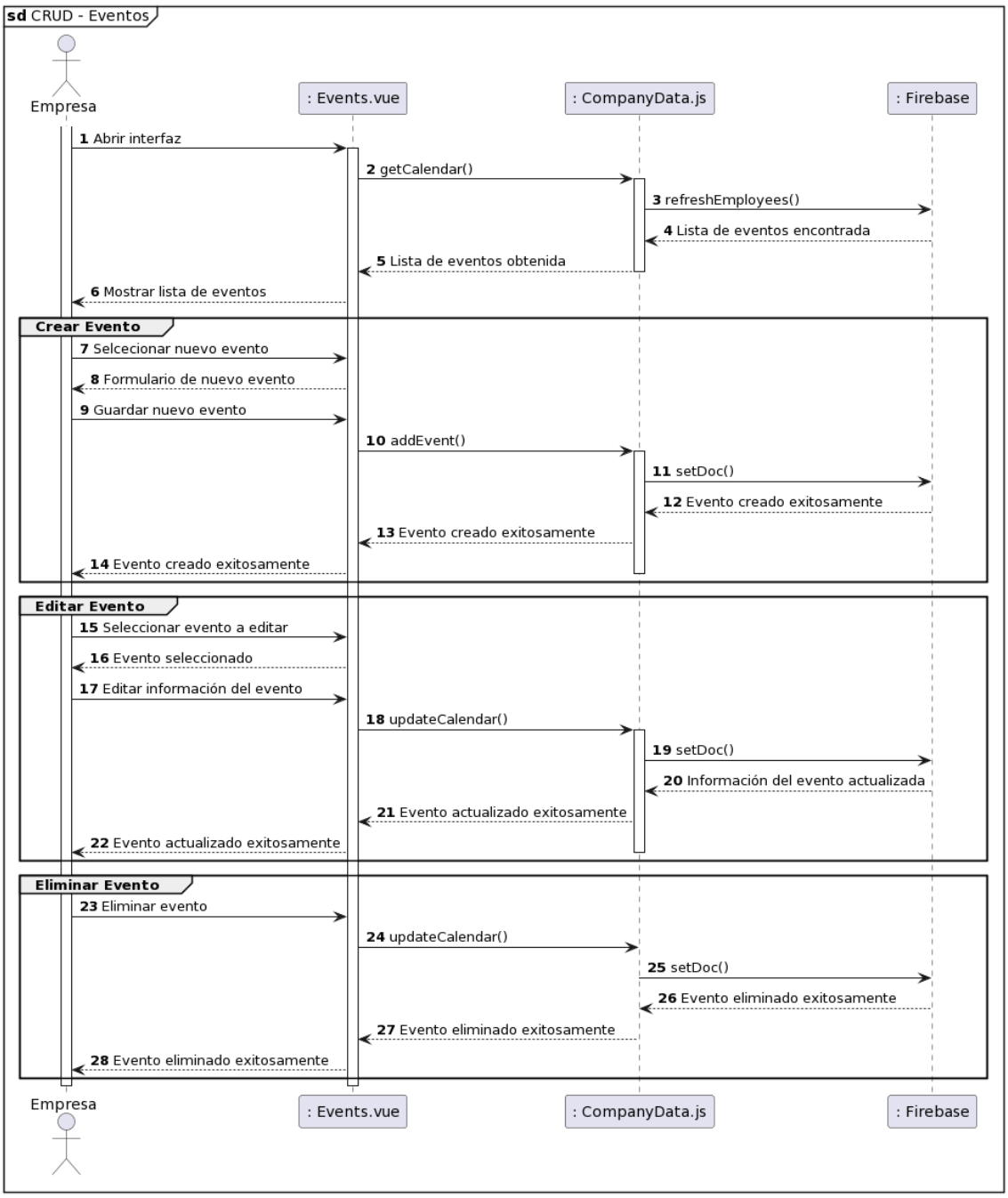


Ilustración 34 Diseño, SD CRUD - Evento

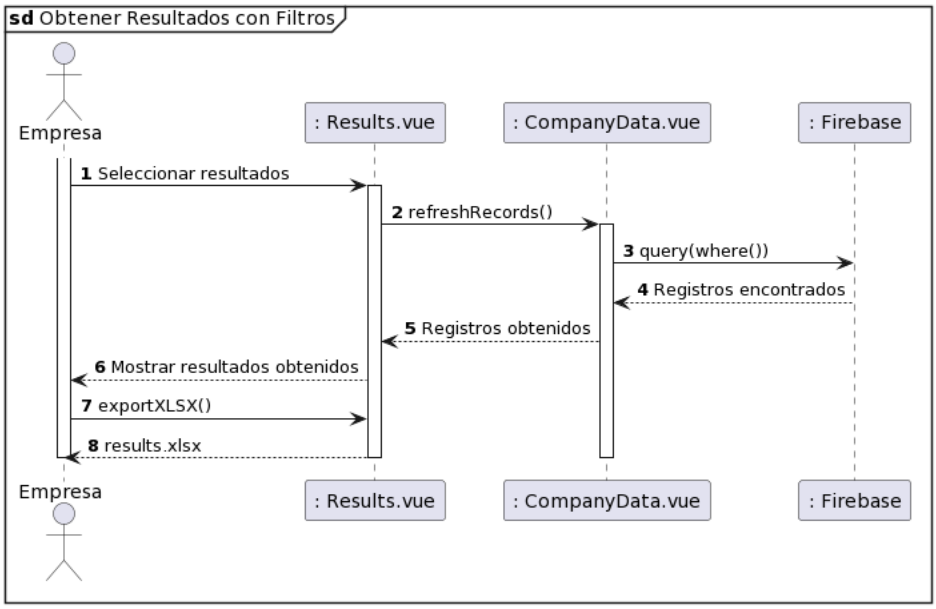


Ilustración 35 Diseño, SD Obtener Resultados

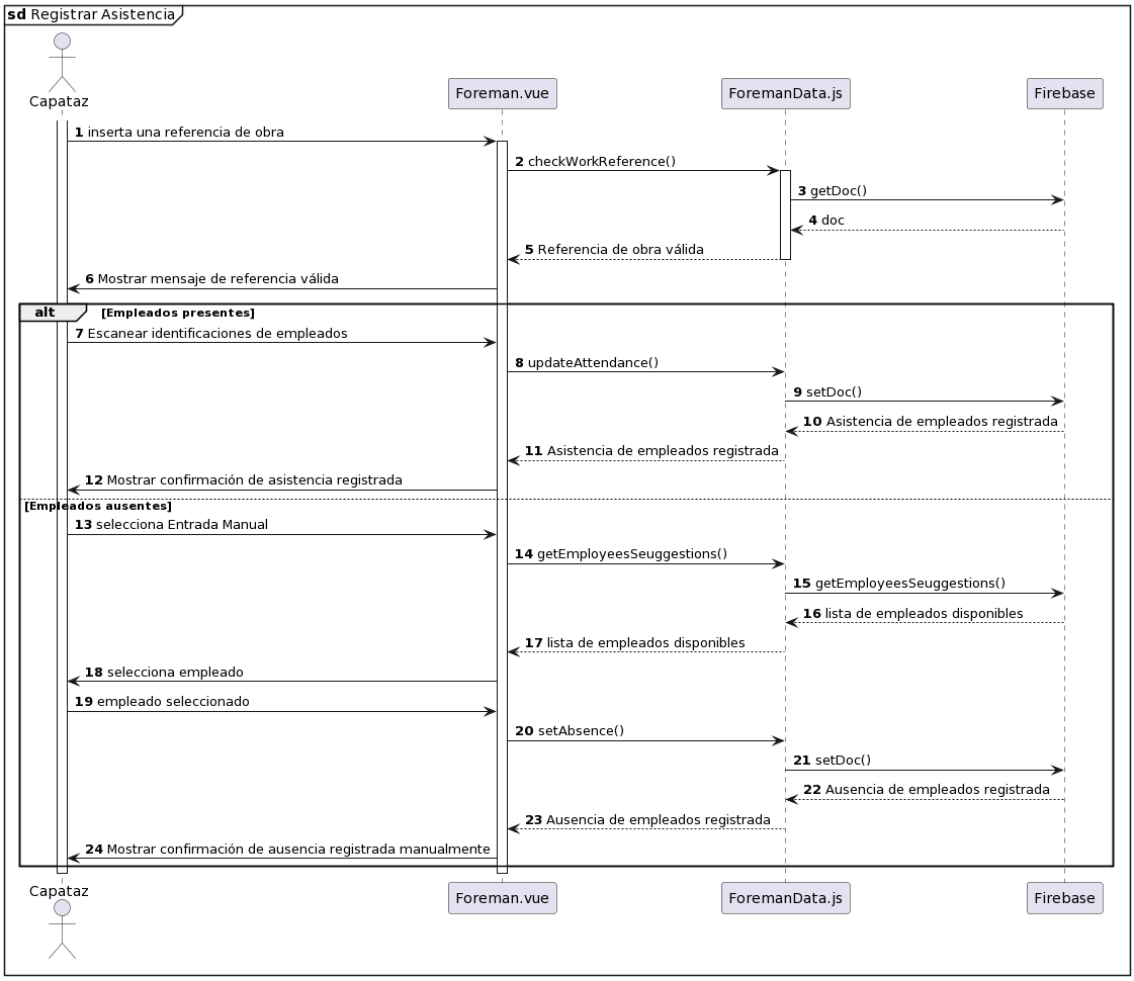


Ilustración 36 Diseño, SD Registrar Asistencia

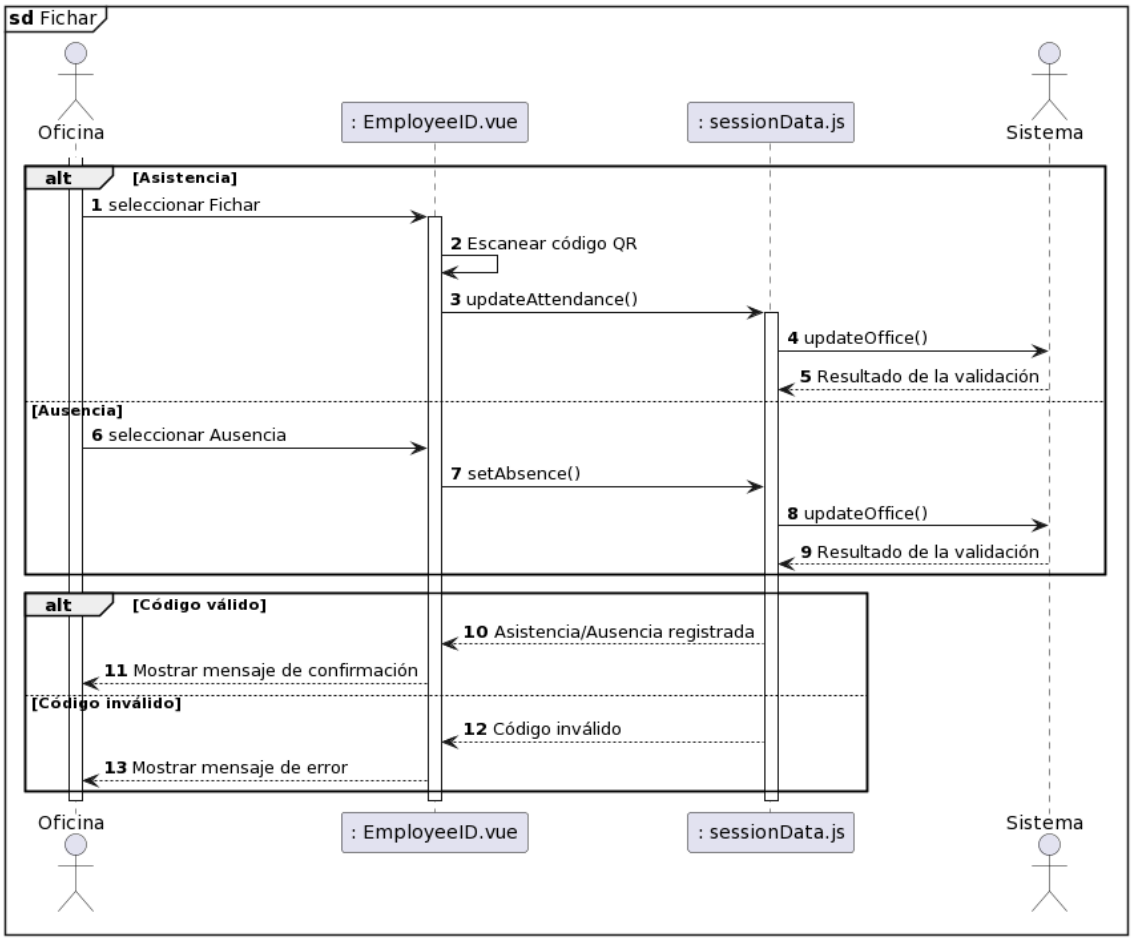


Ilustración 37 Diseño, SD Fichar Oficina

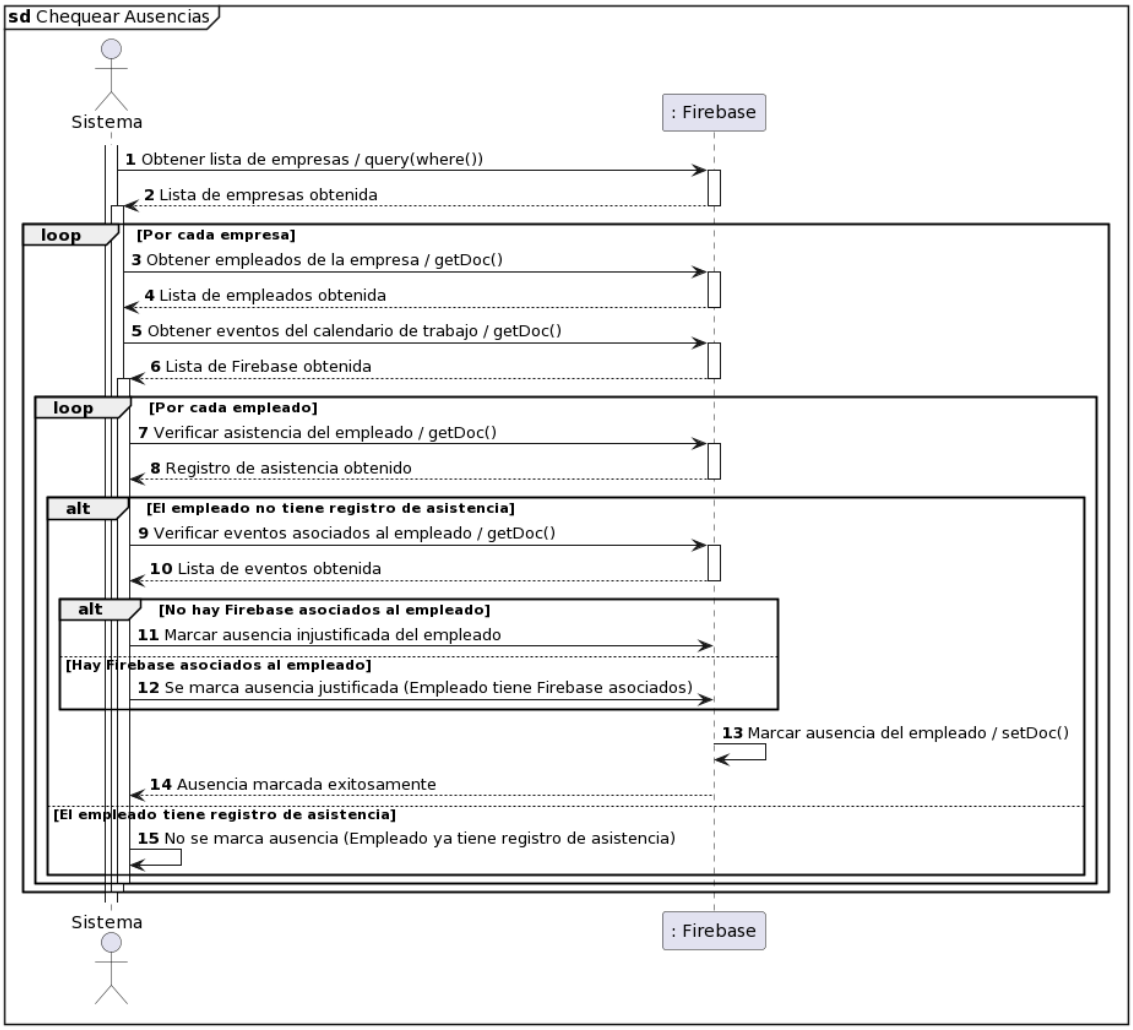


Ilustración 38 Diseño, SD Chequear Ausencias

2.3.4 Diseño de la Interfaz

En este apartado se pueden ver los primeros bocetos de las interfaces de usuario de la aplicación, se muestran modelos para el inició de sesión, registro de empleados, obras, y la interfaz para fichar. Estos bocetos se han dibujado a mano a con una plantilla propia que distingue entre dispositivo móvil u ordenador. En algunas ilustraciones como la Ilustración 41, Ilustración 42 e Ilustración 47, se ha añadido además una captura del resultado final de la interfaz.

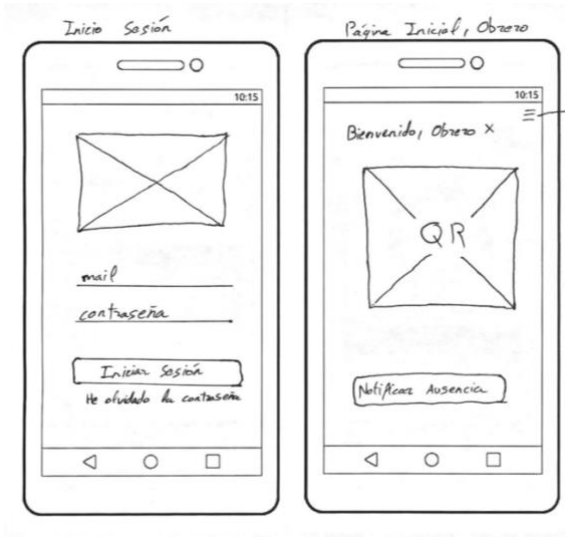


Ilustración 39 Wireframe Empleado Inciar Sesión

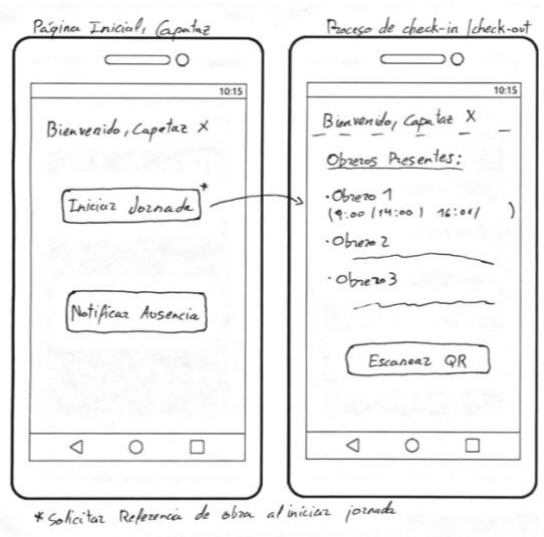
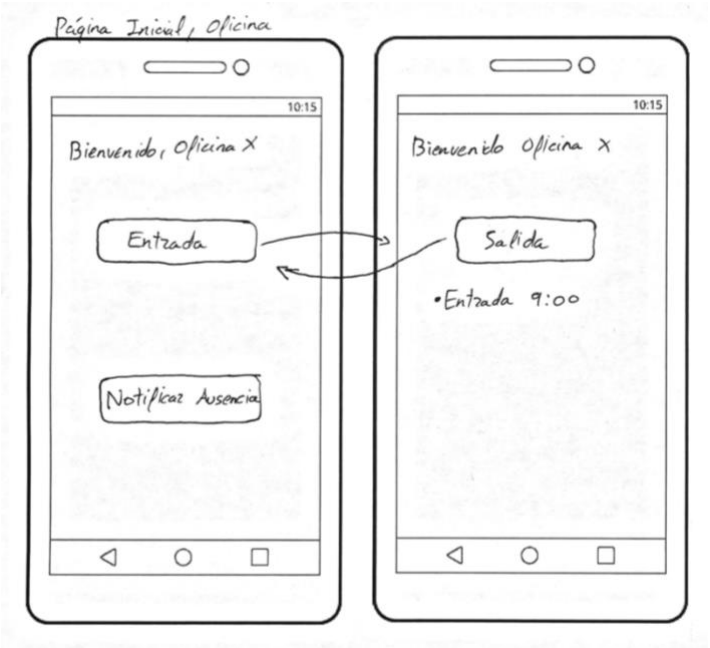


Ilustración 40 Wireframe Empleado Capataz



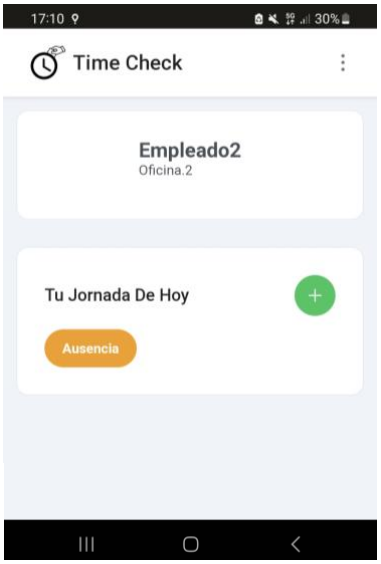


Ilustración 41 Wireframe Empleado Oficina

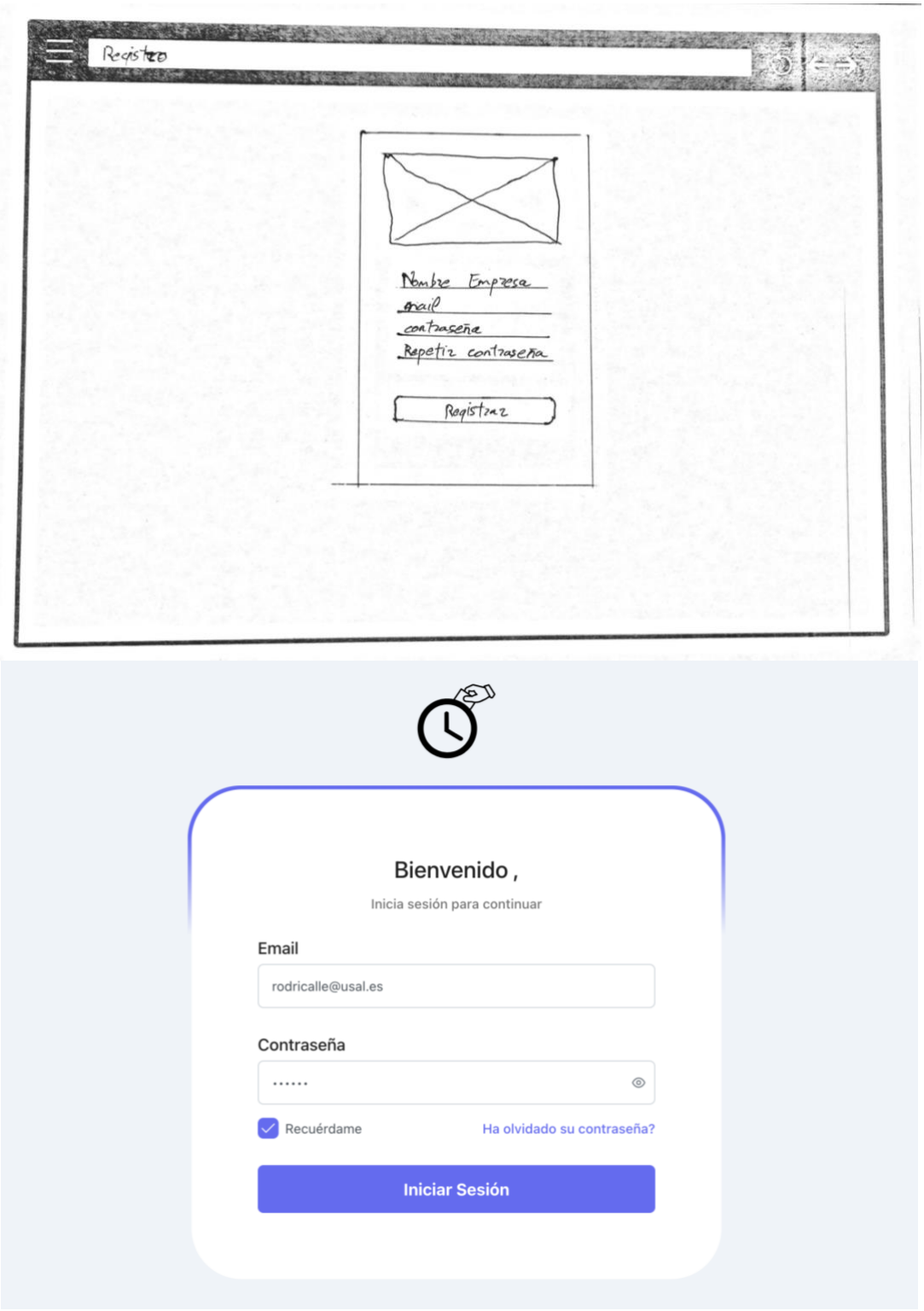


Ilustración 42 Wireframe Administrador Inciar Sesión

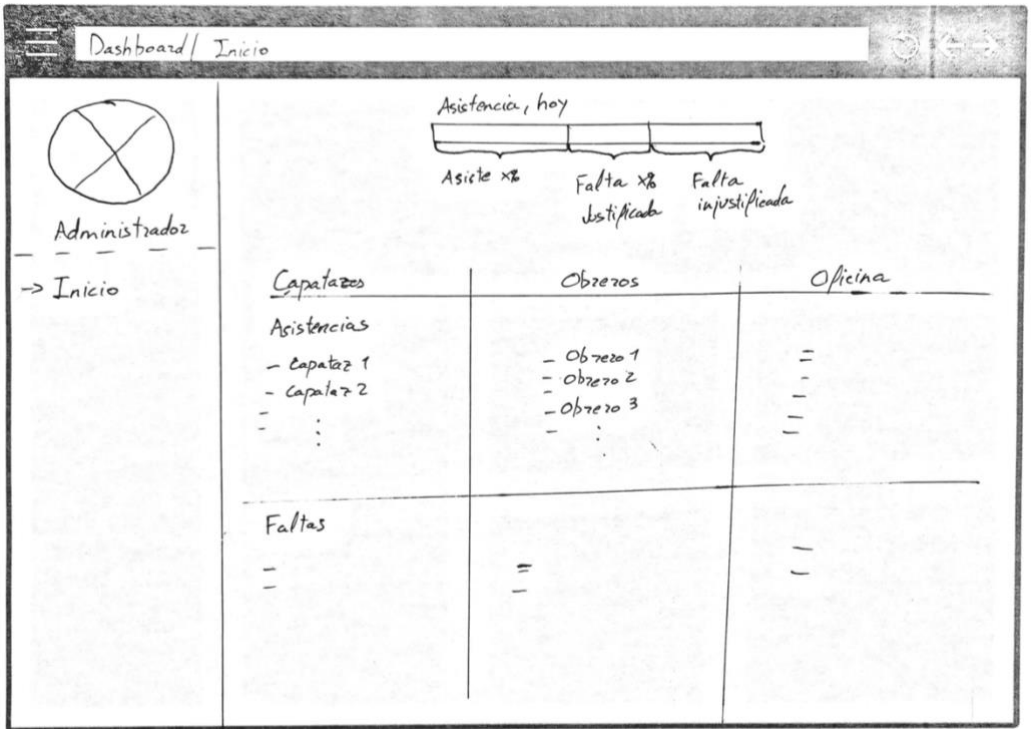


Ilustración 43 Wireframe Administrador Incio

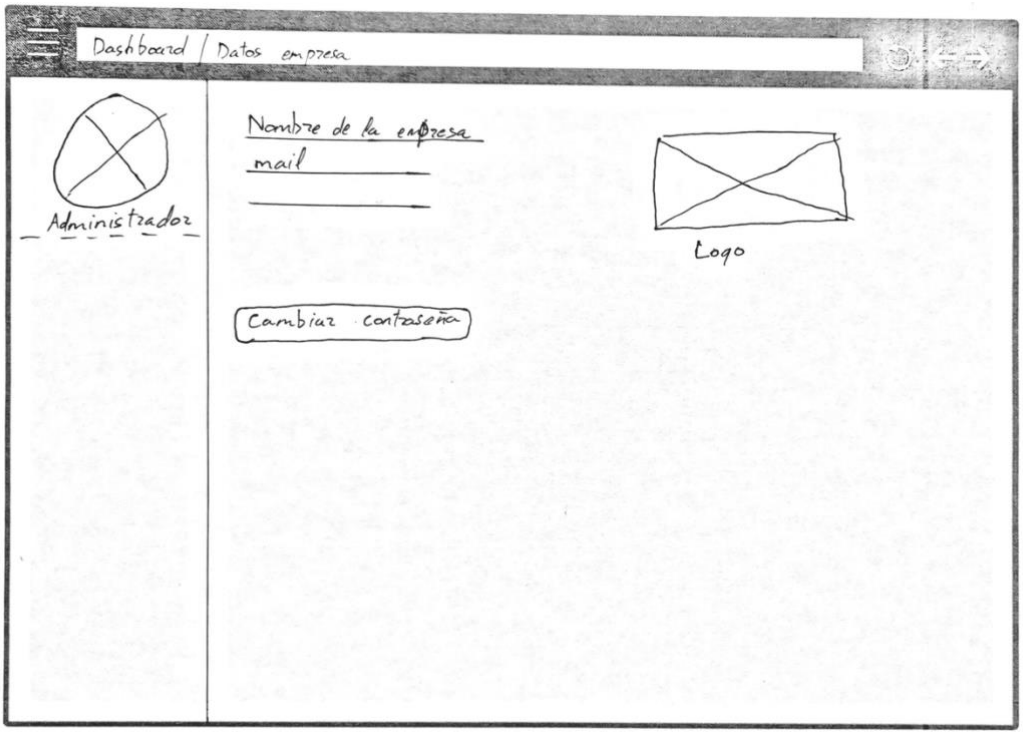


Ilustración 44 Wireframe Administrador Configuración

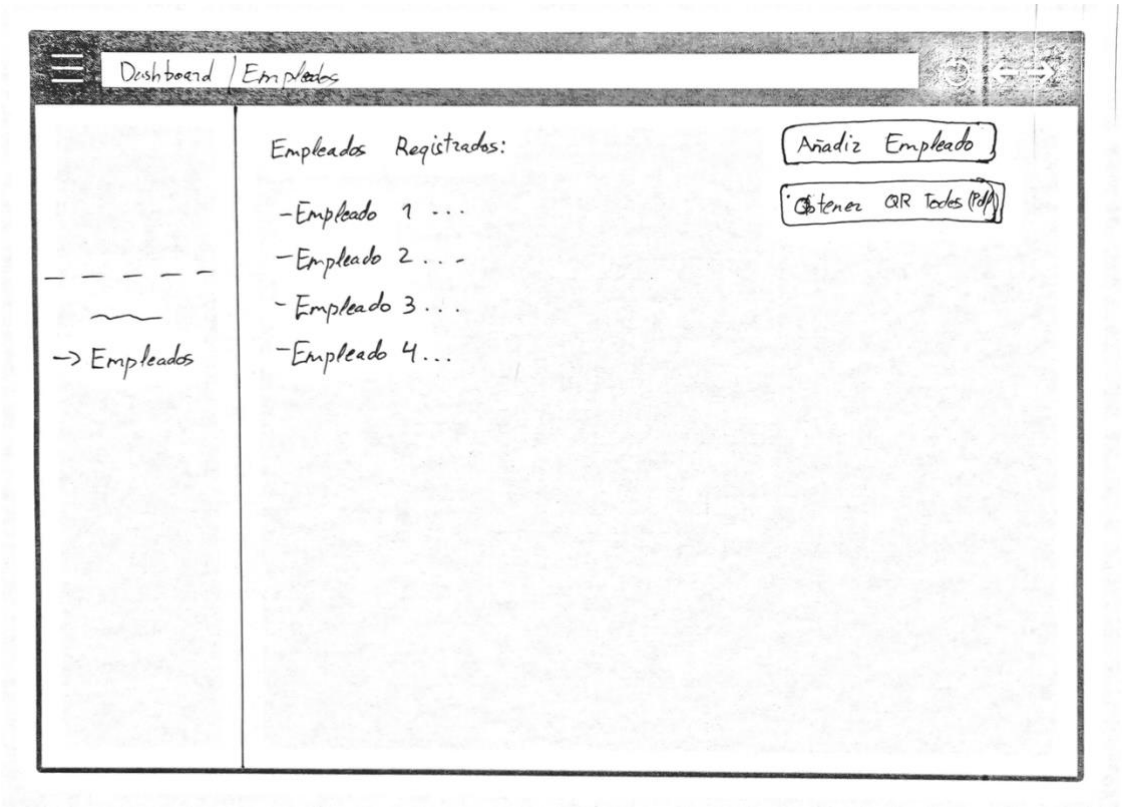


Ilustración 45 Wireframe Administrador Empleados

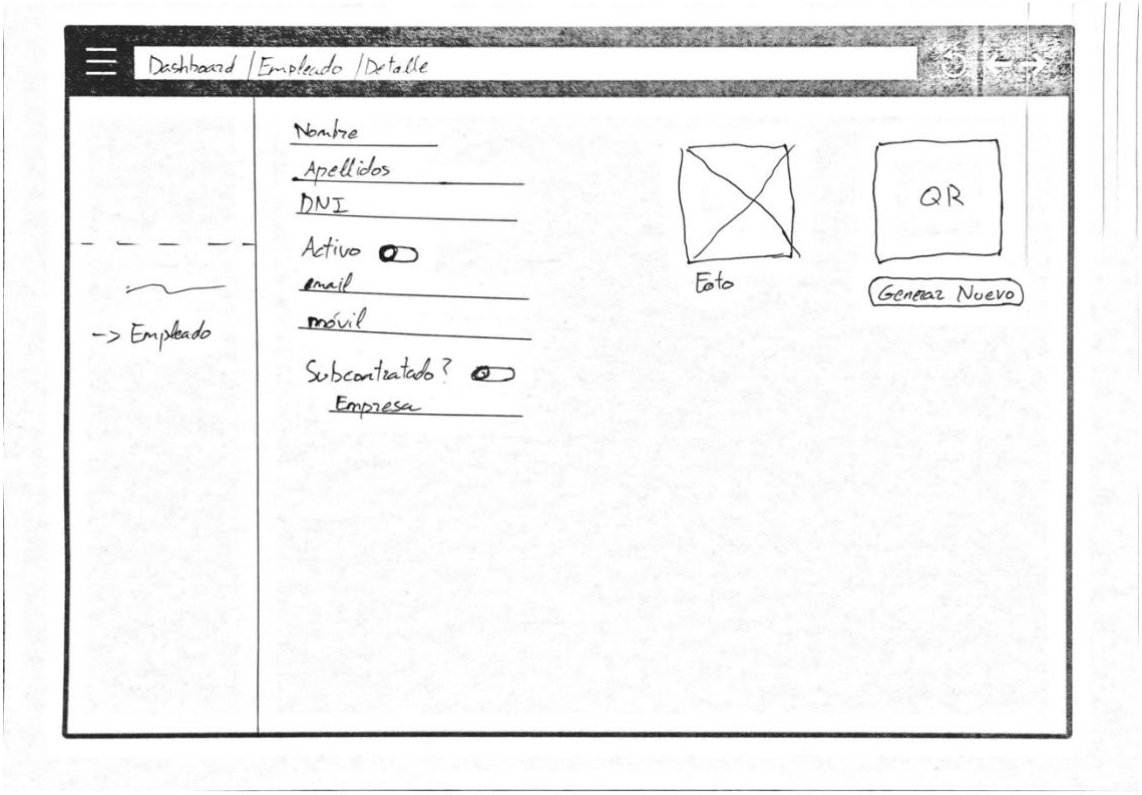


Ilustración 46 Wireframe Administrador Empleado Detalle

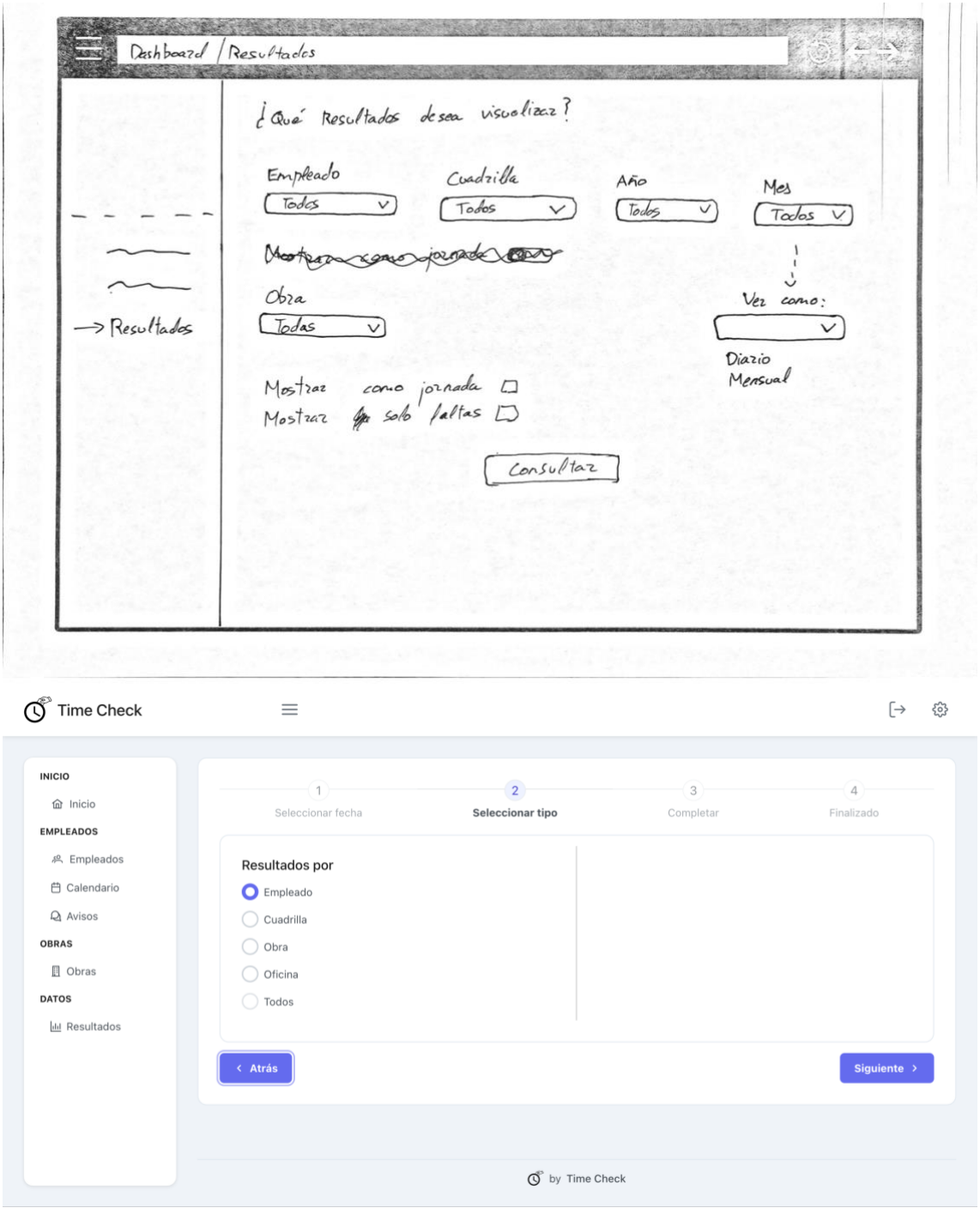


Ilustración 47 Wireframe Administrador Resultados

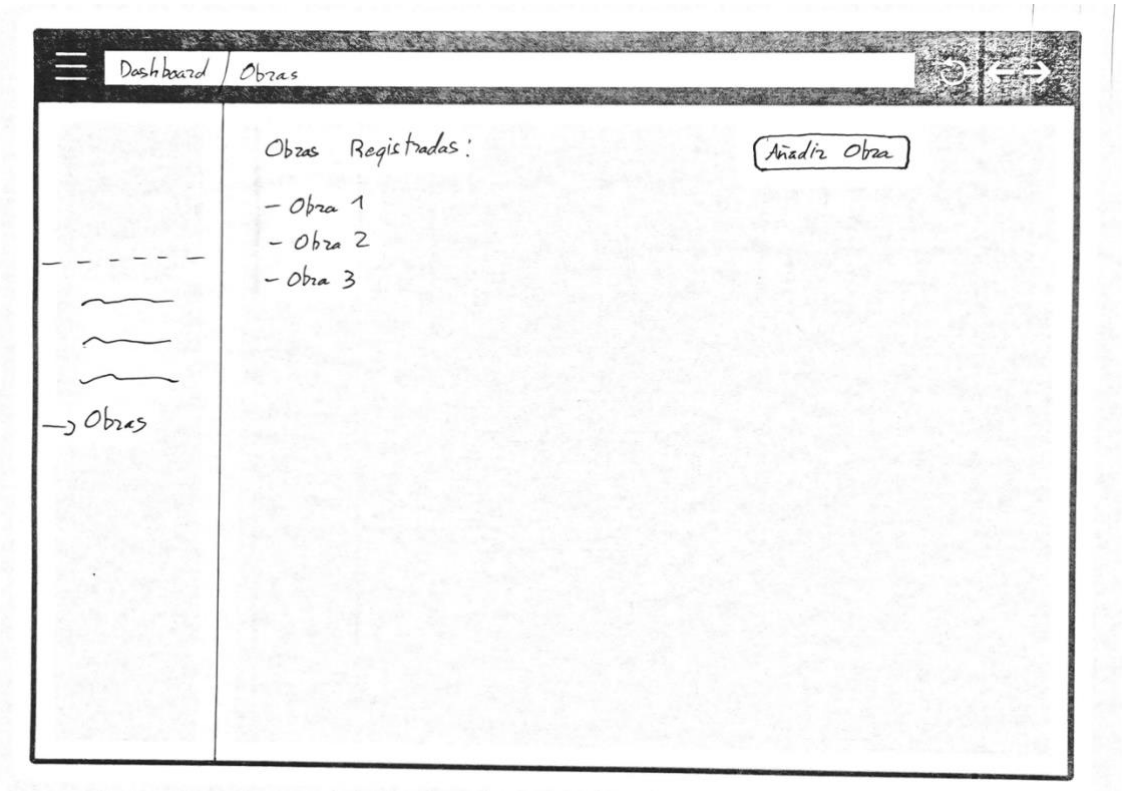


Ilustración 48 Wireframe Administrador Obras

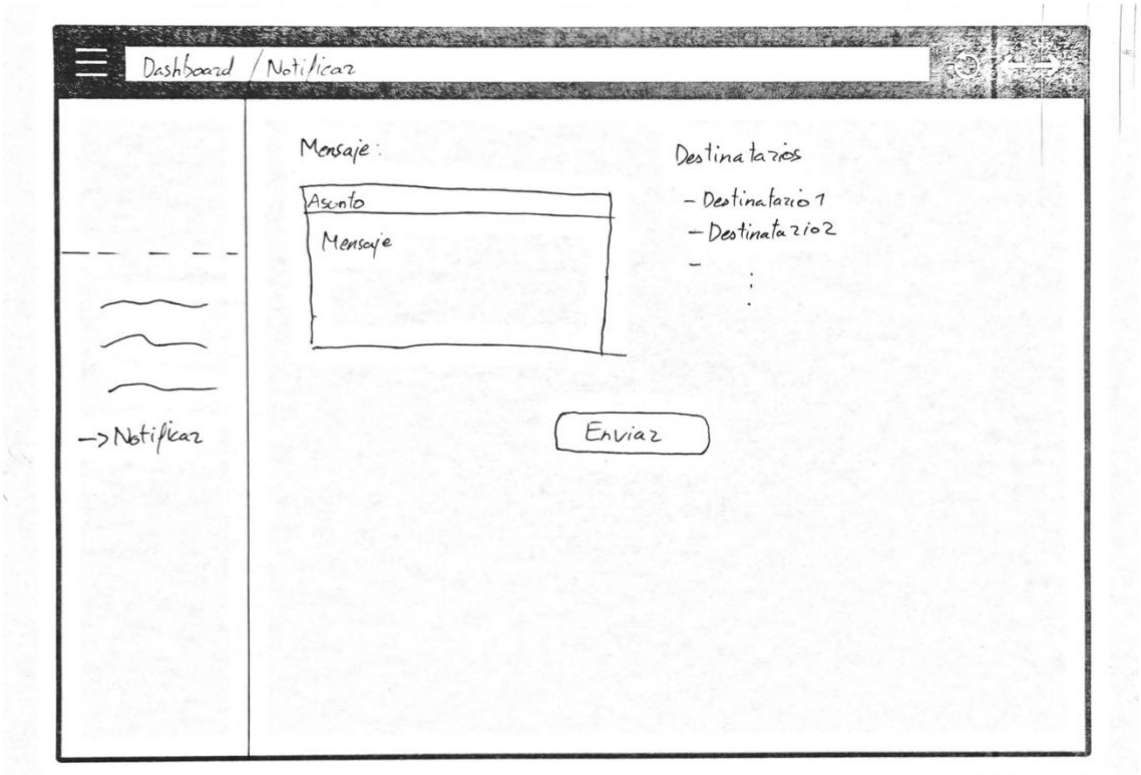


Ilustración 49 Wireframe Administrador Notificaciones

Evidentemente desde los bocetos hasta el resultado final, el aspecto de la interfaz ha sufrido cambios de diseño para mejorar su interacción con respecto a la evolución de la aplicación.

Para los colores de la aplicación se han optado por tonos sencillos y con alto contraste ya que recordemos, los obreros la utilizarán en el exterior y durante el día lo que puede dificultar su visión. Para la elección de estos colores se ha utilizado la herramienta Adobe Color [11] que permite no solo obtener paletas de distintas combinaciones si no comprobar la accesibilidad por contraste entre distintos colores como se puede ver en la Ilustración 50.

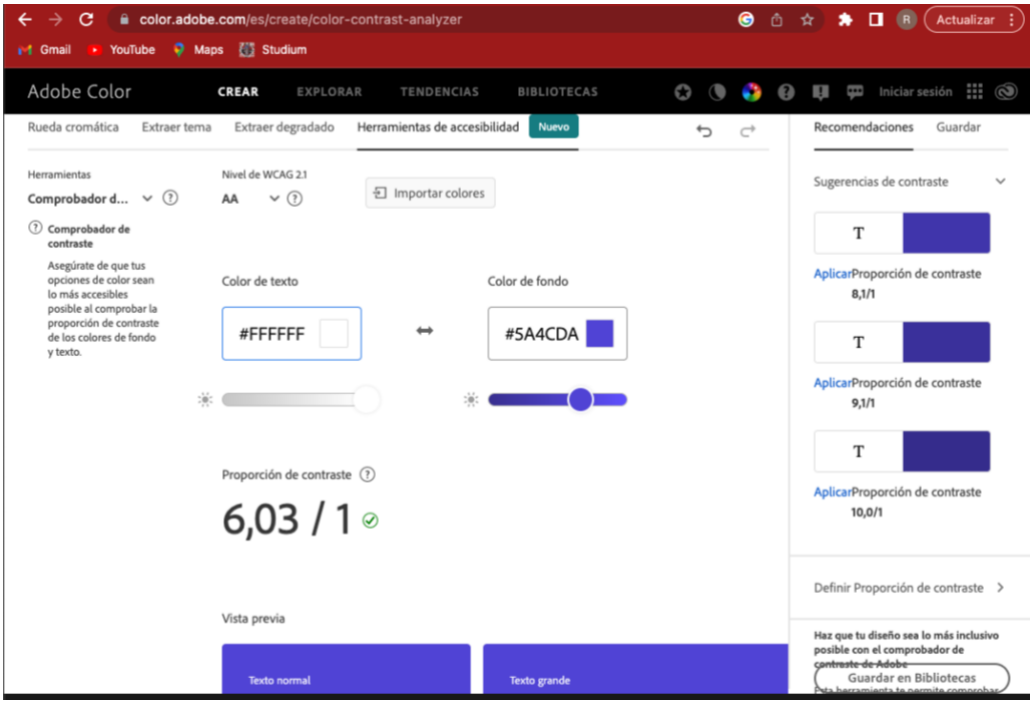


Ilustración 50 Adobe Color [11]

El logo de la aplicación se ha dibujado con la herramienta Canva [12], representa un reloj y la mano de un empleado fichando manualmente como se hacía tradicionalmente. Se encuentra en dos versiones para poder adaptarse al modo oscuro y claro.

Ilustración 51 Logo Oscuro

Ilustración 52 Logo Claro

2.4 Implementación

2.4.1 Framework VueJS

El *front-end* se ha desarrollado utilizando VueJS [13], un framework progresivo que se compone de múltiples componentes. Cada componente es un archivo con extensión ".vue" que contiene de manera organizada el HTML, CSS y JavaScript necesarios. Estos componentes siguen el patrón MVVM, lo que permite el desarrollo de proyectos escalables y modularizados.

VueJS es ampliamente reconocido por su enfoque intuitivo y su capacidad para facilitar la construcción de interfaces de usuario interactivas. Al utilizar VueJS, los desarrolladores pueden aprovechar las características del framework, como la reactividad de los datos y la capacidad de componer componentes complejos a partir de componentes más pequeños y reutilizables.

Además, VueJS cuenta con una amplia comunidad [13] de desarrolladores que contribuyen con bibliotecas y plugins para mejorar su funcionalidad. Esto brinda a los desarrolladores acceso a una gran cantidad de recursos y herramientas que pueden utilizar para agilizar el proceso de desarrollo y mejorar la calidad de sus proyectos.

En resumen, VueJS es una opción sólida para el desarrollo del *front-end*, ya que proporciona una estructura organizada, modularidad y escalabilidad, lo que resulta en un código más mantenible y una mejor experiencia de usuario.

2.4.2 Base de Datos.

Como base de datos se ha utilizado Firebase Cloud Firestore[6], una base de datos NoSQL [14] orientada a documentos y en tiempo real, se muestra en la Ilustración 54.

La estructura de datos se basa en una colección "Users" que contiene los datos de cada usuario del sistema. En el caso de que el usuario sea de tipo empresa (no empleado), tendrá además una estructura para el calendario de trabajo y tres subcolecciones. Una subcolección que almacenará cada obra en un documento, otra para almacenar las notificaciones y una última que contendrá un documento diario con el registro de asistencia para ese día (Ilustración 53).

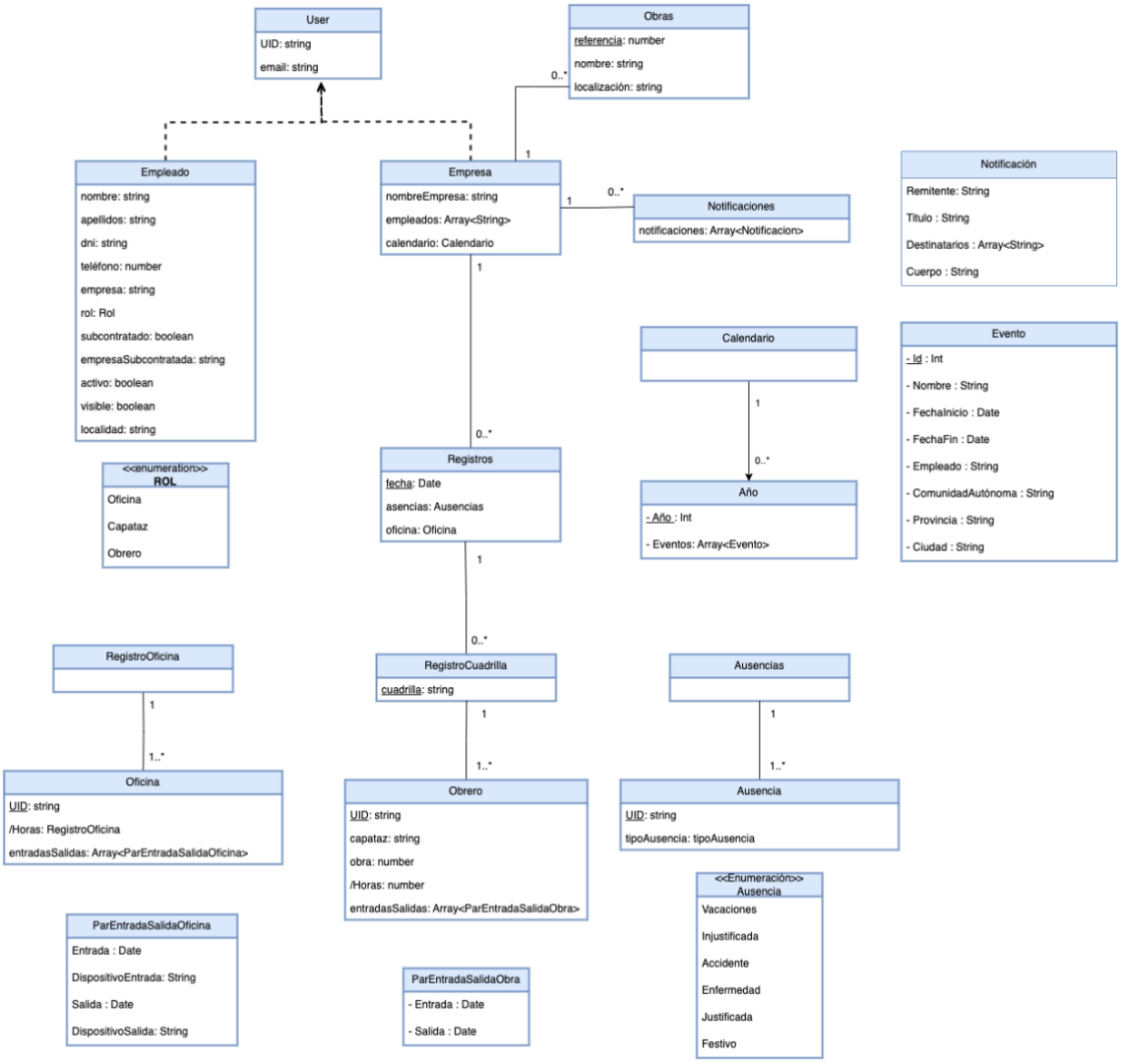


Ilustración 53 Diseño Base de Datos

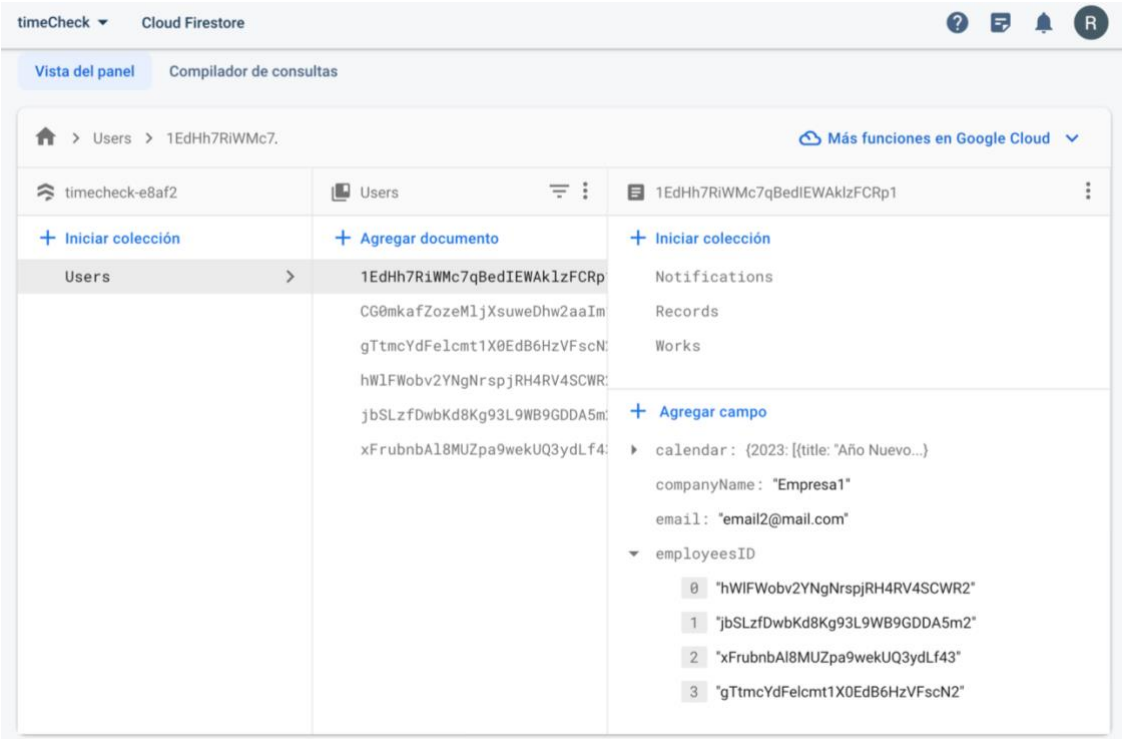


Ilustración 54 Base de datos desde la consola de Firebase

Adicionalmente se utiliza también el servicio RealTime Database [15] para almacenar las claves generadas por los dispositivos de fichar y para almacenar un caché con las ubicaciones y tokens de Firebase Cloud Messaging [8] de los empleados para que sea más rápido el proceso de chequear las ausencias como se muestra en la Ilustración 55.

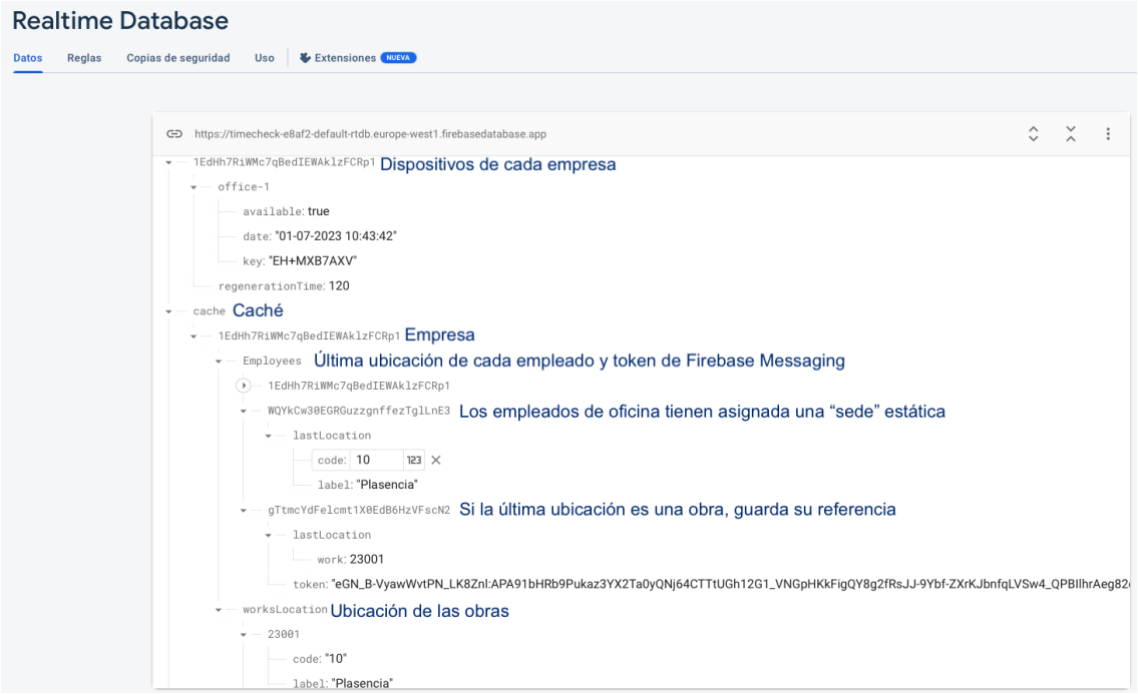


Ilustración 55 RealTime Database

2.4.3 Modelo de Despliegue

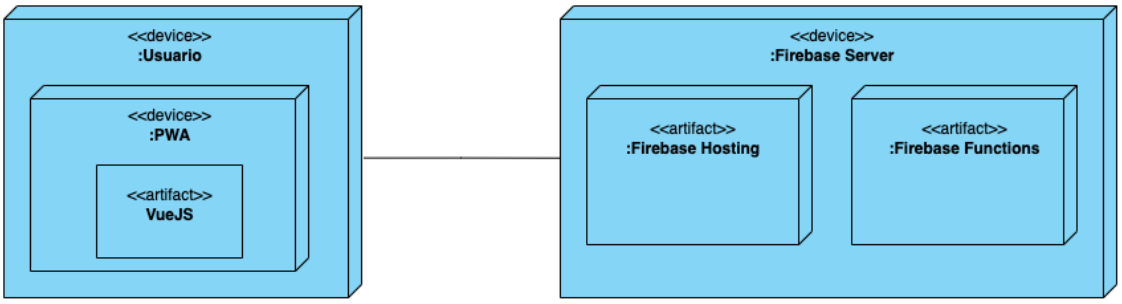


Ilustración 56 Diagrama de Despliegue

Para el alojamiento del *front-end* se ha utilizado Firebase Hosting [16], que proporciona alojamiento seguro, rápido y estable para aplicaciones web.

Para el *back-end* se ha empleado Firebase Cloud Functions [9], que permite ejecutar de forma automática funciones en respuesta a una solicitud HTTP o por disparadores de eventos de cambios en Firestore. El código de las funciones se almacena en la nube y se ejecuta en un entorno administrado.

Para el despliegue local y realizar las pruebas se realiza de la siguiente manera:

1. Instalar Node.js y Npm:

   `sudo apt install nodejs` / **`sudo apt install npm`**

2. **Intalación**: `npm install`

   a. **`npm run dev`** ejecución de la aplicación en modo de desarrollo de forma local.
   b. **`npm run build`** construcción del proyecto para producción.
   c. **`npm run preview`** se utiliza para ejecutar una aplicación ya construida como una vista previa de la compilación de producción.

Para poder usar Firebase y sus utilidades se tiene que instalar la utilidad de consola denominada Firebase CLI [5], usando npm, y asociándose con un proyecto de Firebase creado. Pasos:

- **`npm install -g firebase-tools`** Instalación**.**
- **`firebase login`** Inicio sesión en Firebase**.**
- **Hosting**:
  - **`firebase init`** **(Seleccionando Hosting)** Inicio proyecto de Firebase Hosting.
  - **`firebase deploy --only hosting`** Despliegue de únicamente del hosting
- **Functions:**
  - **`firebase init`** **(Seleccionando Functions)** Inicio proyecto de Firebase Functions.
  - **`firebase deploy --only functions`** Despliegue de únicamente las funciones.

2.5 Pruebas

Para asegurar la correcta evolución del proyecto, después de cada iteración del ciclo de vida (Anexo I) se han ido realizando únicamente pruebas manuales [17] en las cuales se navega por los diferentes componentes del sistema, depurando y revisando errores que puedan surgir para así refinar el producto.

Las pruebas que se han realizado son las siguientes:

- **Pruebas unitarias**: pruebas de bajo nivel, cercanas al código fuente de la aplicación. Se enfocan en probar de forma individual las funciones y/o métodos utilizados por el software. Estas pruebas son específicas y generalmente automatizadas, lo que las hace de menor costo y permite su rápida ejecución mediante un servidor de integración continua. Idealmente, se aísla la funcionalidad para escribir pruebas independientes, verificando la adecuación de nombres, parámetros y resultados. Para evitar dependencias, se reemplazan los servicios externos por funciones locales similares. En algunos casos, también se reemplazan consultas a bases de datos para centrarse en los valores de entrada. Si no es posible aislar el uso de bases de datos, es importante optimizar las consultas para evitar pruebas unitarias de larga duración que puedan ralentizar el desarrollo.
- **Pruebas de integración**: verifican que los diferentes módulos y/o servicios usados por nuestra aplicación funcionen en armonía cuando trabajan en conjunto. Por ejemplo, la interacción con una base de datos,
- **Pruebas funcionales**: se centran en los requerimientos de negocio de una aplicación. Estas pruebas verifican la salida (resultado) de una acción, sin prestar atención a los estados intermedios del sistema mientras se lleva a cabo la ejecución.
- **Pruebas de punta a punta**: replican el comportamiento de los usuarios con el software, en un entorno de aplicación completo. Estas pruebas verifican que los flujos que sigue un usuario trabajen como se espera, y pueden ser tan simples como iniciar sesión.
- **Pruebas de aceptación**: son pruebas formales, ejecutadas para verificar si un sistema satisface sus requerimientos de negocio. Es una mezcla entre las pruebas de integración, funcionales y de punta a punta.

En las últimas fases del proyecto, se han realizado pruebas reales de gran valor, gracias a la empresa voluntaria que han ayudado a reportar errores y deficiencias del sistema.

3 Bibliografía

[1] Visual Paradigm, “Visual Paradigm.” Accessed: Jun. 29, 2023. [Online]. Available: https://www.visual-paradigm.com/

[2] “draw.io.” Accessed: Jun. 29, 2023. [Online]. Available: https://app.diagrams.net/

[3] Anton Zuev, “Patrón MVVM en SwiftUI,” 2020. https://www.adictosaltrabajo.com/2020/06/05/patron-mvvm-en-swiftui/ (accessed Jun. 16, 2023).

[4] Anónimo, “MVC-MVVM (3-5),” Oct. 07, 2012. https://www.adictosaltrabajo.com/2012/10/07/zk-mvc-mvvm/ (accessed Jun. 16, 2023).

[5] Google for Developers, “Referencia de Firebase CLI.” Accessed: May 27, 2023. [Online]. Available: https://firebase.google.com/docs/cli?hl=es-419

[6] Google, “Cloud Firestore.” Accessed: Jun. 21, 2023. [Online]. Available: https://firebase.google.com/docs/firestore?hl=es-419

[7] Google, “Firebase Auth.” Accessed: Jun. 21, 2023. [Online]. Available: https://firebase.google.com/docs/auth?hl=es-419

[8] Google, “Firebase Cloud Messaging.” Accessed: Jun. 21, 2023. [Online]. Available: https://firebase.google.com/docs/cloud-messaging?hl=es-419

[9] Google, “Firebase Functions”, Accessed: Jun. 21, 2023. [Online]. Available: https://firebase.google.com/docs/functions?hl=es-419

[10] Eduardo San Martin Morote, “Pinia.” Accessed: Jul. 03, 2023. [Online]. Available: https://pinia.vuejs.org

[11] “Adobe Color.” https://color.adobe.com/es/create/color-wheel (accessed Jun. 21, 2023).

[12] “Canva.” Accessed: Jun. 21, 2023. [Online]. Available: https://www.canva.com/es_es/

[13] R. de Arriba Garcia, “Estudio de la popularidad del framework VueJS”.

[14] A. C. Romero, J. S. G. Sanabria, and M. C. Cuervo, “Utilidad y funcionamiento de las bases de datos NoSQL,” *Facultad de Ingeniería*, vol. 21, no. 33, pp. 21–32, 2012.

[15] Google, “RealTime Database.” Accessed: Jun. 21, 2023. [Online]. Available: https://firebase.google.com/docs/firestore?hl=es-419

[16] Google, “Firebase Hosting”, Accessed: Jun. 21, 2023. [Online]. Available: https://firebase.google.com/docs/hosting?hl=es-419

[17] “Los diferentes tipos de testing en el desarrollo de software.” Accessed: Jul. 03, 2023. [Online]. Available: https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software