Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de una aplicación web para la administración de empresas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Josep Miquel Manzanera Roma

Tutor: Ana Pont Sanjuan

2019-2020

Resumen

En este trabajo se desarrolla una aplicación web adaptativa para la administración de empresas. Esta aplicación se ha realizado utilizando el framework Quasar para el frontend y Express para la API REST que publica los servicios web necesarios para dotar de funcionalidad a la aplicación web, la cual implementa un sistema de seguridad basado en tokens.

El resultado final es una aplicación capaz de adaptarse a ordenadores y dispositivos móviles que ofrece las funcionalidades de la plataforma de un modo sencillo e intuitivo para el usuario.

Palabras clave: aplicación web, Quasar, Express, API REST, token, framework.

Resum

En aquest treball es desenvolupa una aplicació web adaptativa per a l'administració d'empreses. Aquesta aplicació s'ha realitzat utilitzant el framework Quasar per al frontend i Express per a l'API REST que publica els serveis web necessaris per a dotar de funcionalitat a l'aplicació web, la qual implementa un sistema de seguretat basat en tokens.

El resultat final és una aplicació capaç d'adaptar-se a ordinadors i dispositius mòbils que ofereix les funcionalitats de la plataforma d'una manera senzilla i intuïtiva per a l'usuari.

Paraules clau: aplicació web, Quasar, Express, API REST, token, framework.

Abstract

This project consists of developing a responsive web application for business administration. The application has been developed using Quasar framework as frontend and Express as REST API which provides the web services that makes the platform functional. This API integrates a token-based security system.

As a result, a responsive web application able to work on desktop computers and mobile devices is developed, and it offers all features of the platform in a user-friendly way.

KeyWords: web application, Quasar, Express, REST API, token, framework.

Índice de contenidos

Índice de figuras

1 Introducción

En este primer capítulo de la memoria se realiza una introducción de la situación actual de las herramientas para administrar un negocio, seguido de la motivación que ha llevado al desarrollo de este proyecto, los objetivos y el impacto esperado que se pretende conseguir.

Cuando una persona decide emprender, debe tomar una serie de decisiones muy importantes para asegurar el futuro del proyecto, una de estas decisiones es qué herramientas se van a emplear para administrar y gestionar el emprendimiento, esto, a menudo, nos suele llevar a la conclusión de que no existe ninguna herramienta que se adapte a nuestras necesidades.

Para cada tarea específica debemos buscar una herramienta diferente o innovar en la forma de utilizarla porque no está implementada para usarse como deseamos. Además, las funciones que ofrecen estos administradores son limitadas o hay que pagar una cuantiosa suma para poder hacer uso y disfrute de estas.

Este proyecto surge de la necesidad de centralizar todas o, al menos la mayoría, de las tareas necesarias para llevar a cabo la administración de un negocio, con la posibilidad de que se pueda escalar y adaptarse a cualquier función.

Uno de los problemas más importantes de estas herramientas es que no son libres, es decir, no permiten cambiar el código para adaptar el funcionamiento a las necesidades determinadas de un cliente por lo que solo cubren parcialmente éstas y obligan al cliente a innovar en su uso o usar parcialmente sus funcionalidades.

Por estas razones se desarrolla una aplicación web que permite administrar servicios básicos automatizando tareas recurrentes como la creación de facturas y presupuestos. El código fuente proporcionado puede utilizarse sin modificarse para cubrir unas funciones básicas o modificarse para que se use como una base de un sistema adaptado a las necesidades de cada cliente.

1.1 Motivación

La aplicación web está desarrollada con la última tecnología basada en el diseño MobileFirst[1] y responsive[2], esto proporciona unos conocimientos valorados en el ámbito del desarrollo web y aplicaciones móviles y permiten al programador obtener un buen empleo.

La principal motivación de realizar este trabajo es para el uso propio de dicha aplicación en futuros emprendimientos. El autor de este trabajo se encontraba con la necesidad de disponer de una herramienta que le permitiese crear facturas y presupuestos de una forma rápida y sencilla, automatizando al máximo el proceso para ser lo más eficiente posible. Además de la funcionalidad de administrar los gastos e ingresos de cada proyecto.

Otro detonante de su interés es que necesitaba una herramienta que le permitiese tener el 100% de poder de administración e ir mejorándola según las necesidades que se presentan y llegar a escalar en algo más que una herramienta que solo sirve para administrar.

[1] Se crea una versión optimizada para versiones móviles y luego se amplía
[2] Técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado es desarrollar una aplicación, que use los servicios de una API REST la cual ofrece los servicios necesarios para realizar una administración adecuada de un negocio. Para lograr este objetivo se proponen los siguientes subobjetivos:

1. Desarrollar un *frontend* que se sirva de los servicios de la API REST.
2. Desarrollar un *backend* que proporciones los servicios para administrar un negocio.
3. Permitir trabajar con facturas y presupuestos.
4. Administrar los gastos e ingresos de los negocios.
5. Controlar el fichaje de los trabajadores.
6. Visualizar los datos de la aplicación desde una vista general.
7. Desarrollar una aplicación que se puede utilizar en cualquier dispositivo.

1.3 Planificación temporal

Para cumplir con los objetivos que se han propuesto en el subapartado anterior, se ha realizado una planificación temporal con el objetivo de disponer una idea a priori del tiempo que se necesita para desarrollar la aplicación web.

En primer lugar, para cumplir el primer objetivo habría que programar toda la infraestructura del *frontend* que permitiese comunicarse con el *backend*, por lo que se estima una duración de 1-2 días.

A continuación, para desarrollar un *backend* que facilite los servicios necesarios, hay que desarrollar toda la estructura e implementar todas las funciones que se desean servir. Se estima una duración de 1-2 semanas.

Para permitir trabajar con facturas y presupuestos se deben poder realizar operaciones CRUD con estos, enviarse por correo a los clientes correspondientes, crear pdfs con los datos, etc. Por lo que el tiempo esperado para este objetivo es de 2 semanas.

Con el objetivo de administrar los gastos y los ingresos de los negocios, se debe de realizar un panel de administración que permita realizar operaciones CRUD con los gastos de los negocios y gestionar los ingresos recibidos. Se estima una duración de 1 semana.

En quinto lugar, para controlar los fichajes, se debe permitir a todos los empleados acceder a la página de realizar un fichaje y a los administradores acceder a la página de listado. Para desarrollar estas páginas se espera un tiempo de 1 semana.

Para visualizar los datos de la aplicación a un nivel general hay que desarrollar una página de inicio que muestra los datos más destacables de la aplicación, con gráficas y todo lo necesario. Por lo que se necesitaría de 2 a 5 días para programarse.

Para finalizar, con el objetivo de permitir utilizar la aplicación en cualquier dispositivo hay que programar las páginas de forma que escalen desde la versión móvil a una versión desktop. Este trabajo puede suponer una duración de 1 o 2 días.

Como se puede observar, el tiempo estimado para desarrollar una aplicación web que cumpla con todos los objetivos propuestos tiene una estimación de 5-7 semanas, sin tener en cuenta las posibles variaciones o retrasos que puedan darse.

1.4 Impacto esperado

Los principales beneficiarios de este trabajo son las personas que necesitan una herramienta para administrar un negocio. Estos podrán editar el código fuente para adaptarlo a sus necesidades o usarlo de acuerdo a la versión original.

Con esto se pretende dotar al usuario de un control total del administrador para que, si lo desea, pueda centralizar todas sus herramientas administrativas en un único sitio web, con el mínimo de costes y siendo lo más efectivo posible.

1.5 Estructura

Los capítulos que componen esta memoria reflejan los diferentes pasos que se han seguido para el desarrollo del proyecto. A continuación, se realiza una breve descripción de cada uno de ellos.

En el primer capítulo, se realiza una introducción del proyecto, estableciendo los diferentes objetivos que se persiguen con la realización de éste y el impacto que se espera causar.

A continuación, en el segundo capítulo se explica la situación actual de las herramientas de administración de empresas y qué se pretende cambiar con el desarrollo de este proyecto.

Seguidamente, en el tercer capítulo se recopilan las tecnologías más importantes, explicando con detalle cada una de ellas para que el lector entienda el porqué del uso de estas.

En el cuarto capítulo se detalla un análisis realizado previamente al desarrollo de la aplicación. En él hay una descripción de los diferentes tipos de usuarios y los requisitos funcionales del proyecto.

El quinto capítulo muestra el diseño de la solución, compuesto por la explicación de la arquitectura del sistema, una representación de la capa de persistencia y un conjunto de bocetos que componen la capa de usuario.

En el sexto capítulo se describe el desarrollo que se ha seguido para completar la aplicación. Comienza con la creación de un plan para realizar las tareas, y se detalla cómo se han implementado y el resultado final de cada una de ellas.

El séptimo capítulo contiene las pruebas realizadas para verificar el correcto funcionamiento de la aplicación. Para esto se ha desarrollado un plan de pruebas, permitiendo organizar el proceso de verificación.

Para finalizar, el octavo capítulo contiene las conclusiones obtenidas tras desarrollar el proyecto. Se detalla el cumplimento de los objetivos propuestos y la importancia que han tenido los estudios con la realización del proyecto desarrollado. Y como último apartado se realiza una lista de mejoras y correcciones que se pueden realizar en un futuro.

2 Estado del arte

En la actualidad, se recomienda informatizar en la medida de lo posible todos los procesos de gestión y administración empresarial, para poder así automatizar al máximo los procesos y prevenir la pérdida de datos mediante el uso de copias de seguridad. Para facilitar la realización de estos procesos se crean las herramientas de gestión empresarial.

Las herramientas de gestión empresarial son todos los sistemas, aplicaciones, soluciones de cálculo, etc… que ayudan en las tareas de gestión y administración de una empresa (Leal, 2018).

Actualmente, se pueden encontrar infinidad de herramientas tanto gratuitas como de pago. La mayoría de estas herramientas suelen centrarse en una única área administrativa de la empresa.

En el área de facturación, existen aplicaciones como *invoiceto.me* que permiten crear facturas en formato pdf de manera gratuita.

En cuanto a la gestión de clientes, existe la empresa Zoho CRM, la cual cuenta con múltiples herramientas para administrar y tratar con los clientes.

En el área de contabilidad y finanzas podemos encontrar la herramienta Alegra. Esta es una herramienta que permite crear facturas, registrar pagos y llevar el registro de clientes, entre otras funciones. Ofrece un servicio gratuito para un mes y más adelante se escoge el plan que se ajuste más a las necesidades de cada cliente (Díaz, 2013).

Para cada área en que se puede dividir una empresa se pueden encontrar herramientas que faciliten su administración. Pero también existen herramientas que ofrecen soluciones a diferentes áreas administrativas como es el caso de Gespymes.

Gespymes es un programa de contabilidad, facturación y gestión de empresas. Dispone de múltiples soluciones para diferentes sectores convirtiéndolo en uno de los softwares para la administración de empresas más completos.

2.1 Situación actual de la tecnología

La principal diferencia entre una página web estática y una aplicación web es que la página web estática es un elemento para la mera difusión de información y que la aplicación web es una herramienta para facilitar la realización de una tarea específica.

Las aplicaciones web han ido evolucionando con el tiempo, desde servir páginas con contenido estático a proporcionar usos interactivos y escalables.

Uno de los primeros lenguajes de programación para el desarrollo de aplicaciones web es Perl, el cual fue inventado por Larry Wall en 1987. Pero no fue hasta 1995 cuando Rasmus Lerdorf publicó el lenguaje PHP gracias al cual comenzó a despegar el desarrollo de aplicaciones web.

La verdadera revolución se dio cuando Netscape, uno de los navegadores más antiguos, permitió la ejecución de scripts por parte del navegador, desarrollando más adelante el lenguaje JavaScript (Barzanallana, 2012).

Actualmente el desarrollo de aplicaciones web se hace mediante *frameworks*. Un framework de aplicaciones web es un entorno software que facilita la tarea de escribir, mantener y escalar las aplicaciones. Estos entornos proporcionan herramientas y librerías que simplifican operaciones comunes en el desarrollo web (Developer mozilla, 2019).

Según las estadísticas de Stack Overflow del 2019, el framework web más utilizado es jQuery(48,7%), seguido de React.js(31,3%) y por último Angular.js(30.7%).

Hoy en día podemos encontrar diferentes formas de servir las aplicaciones web, por una parte, encontramos las SPA (*Single Page Application*). Estas aplicaciones web interactúan con el usuario reescribiendo dinámicamente la página actual en lugar de cargar páginas nuevas, por lo que la página no se recarga en ningún momento del proceso y, requiere de una carga inicial de recursos más elevada. Las SPA devuelven un archivo HTML incompleto o vacío y lo completan con JavaScript.

Uno de los principales problemas de las SPA, es que son difíciles de indexar por los motores de búsqueda, por esta razón se crearon las SSR (*Server Side Rendering*). Estas aplicaciones web se ejecutan tanto en el cliente como en el servidor, por lo que devuelven un archivo HTML que sí se puede indexar por los buscadores.

Por último, podemos destacar las PWA (*Progressive Web Application*), las cuales ofrecen una experiencia al usuario similar a la de una aplicación móvil. Por lo que tienen que cumplir ciertos requisitos (Quasar framework, 2020).

2.2 Crítica al estado del arte

Las herramientas de gestión empresarial más valoradas se centran en el modelo SaaS, ofreciendo acceso a la plataforma mediante pago. Una desventaja de este modelo es que se depende completamente del proveedor del software, dotándolo de total poder sobre la seguridad de los datos.

También cabe destacar el hecho de que estas plataformas no son adaptables a las necesidades propias ni personalizables para cada usuario, por lo que solo complacerá un cierto número de estas y se tendrá que adaptar al funcionamiento de dicha plataforma.

Por finalizar, una gran cantidad de herramientas gratuitas se centran en resolver una única tarea, por ejemplo crear facturas. Pero si lo que se pretende es cubrir un número elevado de necesidades se necesita disponer de una gran cantidad de herramientas, dificultando la compatibilidad entre ellas y su automatización, obligando al usuario a contratar una herramienta de gran coste.

2.3 Propuesta

Como se especifica en la introducción, en este trabajo se desarrolla una aplicación web para la administración de empresas ofreciendo a sus usuarios control total del funcionamiento y aspecto de la herramienta.

De esta forma los usuarios pueden hacer uso y disfrute del servicio sin modificación alguna o, editando el código fuente para que se adapte a alguna función especial. Se ofrecerá, pues, al usuario la posibilidad de mejorarla y decidir sobre aspectos tan importantes como la seguridad de los datos.

Esta aplicación web, se desarrolla pensando en las necesidades esenciales de las empresas por esta razón se ofrece una solución a las tareas de:

- Crear facturas y presupuestos.
- Administrar clientes.
- Gestionar los trabajadores y los fichajes.
- Administrar gastos e ingresos.

3 Tecnología utilizada

Para el desarrollo de la aplicación se hace uso de un conjunto de herramientas y entornos de desarrollo (*frameworks*) que facilitan la realización de las tareas. A continuación, se explican con detalle las tecnologías más utilizadas.

3.1 Vue.js

Vue.js es un entorno de desarrollo de JavaScript de código abierto para construir interfaces de usuario y aplicaciones SPA que fue desarrollado por Evan You en 2014. Vue.js se autodenomina como un *framework* progresivo.

La principal diferencia entre Vue.js, React.js y Angular es que Vue tiene las partes muy bien desacopladas, organizando el código en pequeños componentes formados por HTML, CSS y JavaScript. Estos componentes organizan un árbol jerárquico que forma la aplicación final.

3.2 Quasar framework

Quasar es un entorno de desarrollo basado en Vue.js que permite a los desarrolladores crear sitios web y aplicaciones (SPA, SSR, WPA, extensiones de navegador, aplicaciones móvil y aplicaciones de escritorio).

Quasar se centra en facilitar todas las herramientas para crear aplicaciones muy visuales y reducir el coste del desarrollo permitiendo, con un mismo código, crear diferentes tipos de aplicaciones.

3.3 Quasar CLI

Quasar CLI es una herramienta proporcionada por Quasar que facilita el desarrollo de aplicaciones. Cuando se crea un nuevo proyecto, proporciona la estructura necesaria para que funcione.

También se puede utilizar para la creación de partes de la aplicación, ya sean componentes, páginas, layouts o stores.

3.4 Nodejs

Node.js es un entorno de programación multiplataforma, de código abierto, creado para la capa de servidor (aunque no se limita solo a está) basado en JavaScript. Se desarrolló para la creación de programas de red altamente escalables.

3.5 NPM

Es un sistema de gestión de paquetes por defecto para Node.js. Entre los cuales se incluyen Quasar framework y todos los paquetes empleados en el proyecto.

3.6 Express.js

Es un entorno de desarrollo de aplicaciones web para Node.js diseñado para crear aplicaciones web y API. Es un *framework* muy minimizado con una gran cantidad de extensiones y módulos.

3.7 TypeScript

TypeScript es un lenguaje de código abierto basado en JavaScript desarrollado por Microsoft. Es un superconjunto de JavaScript que añade tipos y objetos basados en clases.

3.8 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft. Entre sus funcionalidades se incluyen:

- Depuración.
- Integración con Git.
- Resaltado de sintaxis.
- Refactorización de código.

3.9 Git

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia del mantenimiento de versiones cuando éstas tienen un gran número de archivos de código fuente. Permite llevar un registro de los cambios y coordinar el trabajo en equipo que se realiza sobre los archivos compartidos.

3.10 Firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google.

Firebase usa un conjunto de herramientas para la creación y sincronización de proyectos haciendo posible el crecimiento del número de usuarios.

3.11 TypeORM

TypeORM es un ORM que puede ejecutarse en plataformas NodeJS. Puede utilizarse con TypeScript y JavaScript. Su principal objetivo es permitir las últimas funciones de JavaScript para poder desarrollar cualquier tipo de aplicaciones que utilice bases de datos.

4 Análisis

Una vez se han descrito las tecnologías más importantes empleadas en el desarrollo del proyecto, se va a realizar un análisis del problema. En la primera parte se describen los diferentes usuarios identificados en la aplicación. A continuación, se detallarán los requisitos funcionales que se implementarán para cubrir las necesidades de los usuarios.

4.1 Características del usuario

Para el desarrollo de este proyecto se han tenido en cuenta 3 tipos de usuarios: Usuario empleado, usuario administrador y usuario no autentificado.

El usuario empleado, es aquel que tiene el rol de empleado. Este usuario solo puede registrar horas en el sistema, por lo que no tiene acceso al resto de vistas de la aplicación.

El usuario administrador es el que tiene el rol de administrador. Tiene los permisos para poder realizar cualquier acción en el sistema.

Por último, el usuario no autentificado solo puede realizar la acción de inicio de sesión.

4.2 Requisitos funcionales

Los requisitos funcionales definen las funciones esenciales que debe implementar una aplicación. Cada función está compuesta por una entrada y una salida. Los datos que hay que introducir en la entrada de cada requisito son los datos mínimos y no todos los que se pueden llegar a introducir.

Para este proyecto se han definido los siguientes requisitos funcionales:

Iniciar sesión

Descripción: Un usuario registrado debe poder acceder a la aplicación mediante sus credenciales.

Referencia: RF01.

Entrada: Correo electrónico y contraseña.

Salida:

- Si las credenciales son correctas y el usuario es un trabajador accede a la pantalla de fichajes.
- Si las credenciales son correctas y el usuario es un administrador accede a la pantalla de inicio.
- Si las credenciales son incorrectas se notifica el error.

Cerrar sesión

Descripción: Un usuario que ha iniciado sesión debe poder cerrar la sesión.

Referencia: RF02.

Entrada: -

Salida: Se cierra la sesión actual en la aplicación.

CRUD usuarios

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar usuarios.

Referencia: RF03.

Entrada:

- **Crear:** Email, contraseña y rol.
- **Editar:** uid, email, contraseña y rol.

Salida: Se muestran los datos de los usuarios.

CRUD negocios

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar negocios.

Referencia: RF04.

Entrada:

- **Crear:** Nombre, NIF, configuración email, contactos, direcciones y cuentas.
- **Editar:** Nombre, NIF, configuración email, contactos, direcciones y cuentas.

Salida: Se muestran los datos de los negocios.

CRUD costes

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar costes.

Referencia: RF05.

Entrada:

- **Crear:** Negocio, tipo de coste, vendedor, fecha, IVA y cantidad.
- **Editar:** Negocio, tipo de coste, vendedor, fecha, IVA y cantidad.

Salida: Se muestran los datos de los costes.

CRUD clientes

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar clientes.

Referencia: RF06.

Entrada:

- **Crear:** Nombre, NIF, email, contactos y direcciones.
- **Editar:** Nombre, NIF, email, contactos y direcciones.

Salida: Se muestran los datos de los clientes.

CRUD empleados

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar empleados.

Referencia: RF07.

Entrada:

- **Crear:** Negocio y usuario.
- **Editar:** Negocio y usuario.

Salida: Se muestran los datos de los empleados.

CRUD facturas

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar facturas.

Referencia: RF08.

Entrada:

- **Crear:** Nombre, cliente, negocio, fecha, IVA, elementos de la factura, y pago.
- **Editar:** Nombre, cliente, negocio, fecha, IVA, elementos de la factura, y pago.

Salida: Se muestran los datos de las facturas.

CRUD presupuestos

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar presupuestos.

Referencia: RF09.

Entrada:

- **Crear:** Fecha, fecha de entrega, cliente, negocio, nombre, estado, IVA, descuento, desarrollos y pagos.

- **Editar:** Fecha, fecha de entrega, cliente, negocio, nombre, estado, IVA, descuento, desarrollos y pagos.

Salida: Se muestran los datos de los presupuestos.

CRUD métodos de pago

Descripción: Un usuario administrador debe poder añadir, editar, listar, filtrar y eliminar métodos de pago.

Referencia: RF10.

Entrada:

- **Crear:** Nombre.
- **Editar:** Nombre.

Salida: Se muestran los datos de los métodos de pago.

Enviar factura al cliente

Descripción: Un usuario administrador debe poder enviar un correo electrónico con una factura a un cliente.

Referencia: RF11.

Entrada: Título, texto y factura.

Salida:

- Si hay un error se muestra por pantalla.
- Si el mensaje se envía con éxito se muestra por pantalla.

Enviar presupuesto al cliente

Descripción: Un usuario administrador debe poder enviar un correo electrónico con un presupuesto a un cliente.

Referencia: RF12.

Entrada: Título, texto y presupuesto.

Salida:

- Si hay un error se muestra por pantalla.
- Si el mensaje se envía con éxito se muestra por pantalla.

Registrar horas

Descripción: Un usuario que ha iniciado sesión debe poder registrar horas.

Referencia: RF13.

Entrada: Mensaje.

Salida:

- Si hay un error se muestra por pantalla.
- Si se registra con éxito se muestra por pantalla.

Crear factura desde presupuesto

Descripción: Un usuario administrador debe poder crear una factura desde un pago asociado con un presupuesto.

Referencia: RF14.

Entrada: Presupuesto.

Salida:

- Si hay un error se muestra por pantalla.
- Si se crea la factura con éxito, se redirige a la página para editar dicha factura.

5 Diseño de la solución

Una vez se ha realizado el análisis del problema, se procede a diseñar una solución viable para llevar a cabo la implementación. En esta sección se profundizan los diferentes elementos que contiene la aplicación.

5.1 Arquitectura del sistema

La solución propuesta se basa en la arquitectura cliente/servidor donde se pueden distinguir principalmente 2 componentes:

- Cliente: Es el encargado de realizar la petición AJAX al servidor para que este realice una función.
- Servidor: Ofrece servicios como una API REST y atiende peticiones.

En el esquema mostrado en la Figura 1 se ilustra cómo se relacionan estos 2 componentes entre sí:

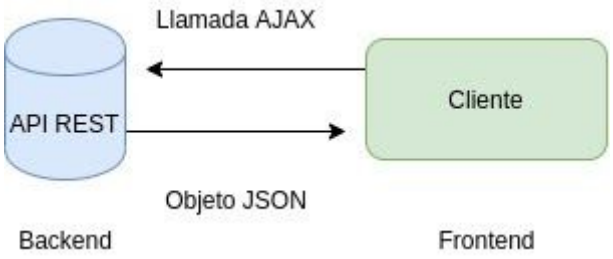

Figura 1: Esquema comunicación frontend-backend

Para el desarrollo del *frontend* se emplea Vue.js, el cual utiliza el patrón de diseño Modelo-Vista-VistaModelo más conocido como MVVM, Figura 2. La principal diferencia entre el patrón Modelo-Vista-Controlador y MVVM es que este último modelo dispone de la llamada unión en 2 sentidos gracias a la cual un cambio producido en la vista se refleja en el modelo y viceversa.

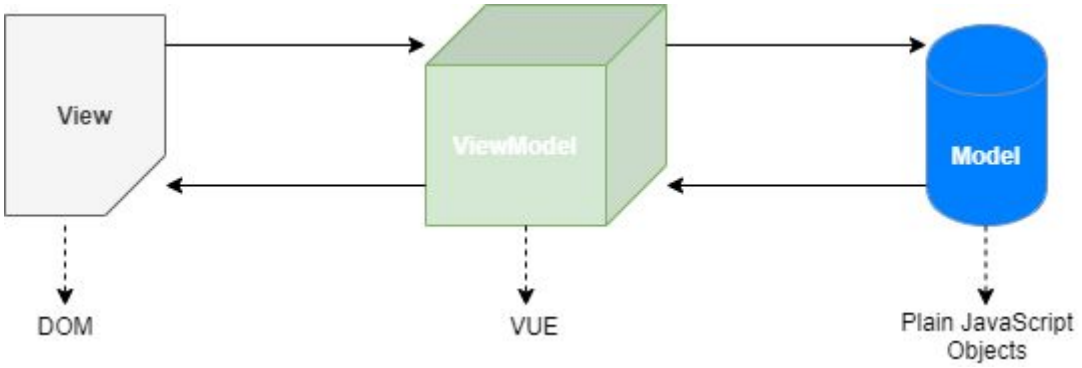

Figura 2: Diseño MVVM

El *backend* se ha diseñado mediante el entorno de desarrollo Express.js. Este entorno o *framework* no tiene una estructura definida por lo que permite organizar el proyecto como el desarrollador prefiera. En la Figura 3 se explica el funcionamiento que se sigue para atender las peticiones que realiza el *frontend*:

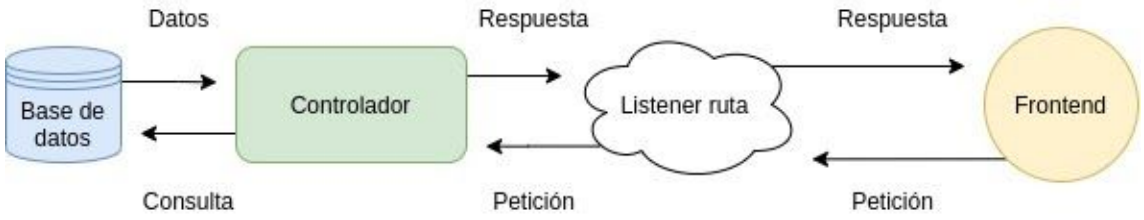

Figura 3: Diseño funcionamiento backend

5.2 Diseño detallado

Este apartado se detalla, en primer lugar la capa de persistencia de la aplicación y, en segundo, se muestra la capa de presentación mediante unos bocetos.

5.2.1 Capa de persistencia

A continuación se describen las diferentes entidades que forman parte del proyecto, junto con sus relaciones entre ellas mediante un diagrama de clases como se muestra en la Figura 4.

- **User:** Representa un usuario registrado en el sistema. Mediante un usuario se puede acceder al sistema y, según el rol que tenga, se podrán realizar diferentes tipos de acciones.
- **UserFirebase:** Es el usuario que se encuentra en la base de datos de *firebase* que permite la autentificación con el sistema. Se relaciona con el usuario de la propia base de datos mediante el atributo uid.
- **Address:** Es la representación de una dirección. Las direcciones pueden pertenecer a un negocio, cliente o trabajador.
- **Business:** Representa un negocio dentro del sistema.
- **BusinessAccount:** Representa una cuenta de un negocio. Se utilizan para indicar el método de pago y la cuenta del beneficiario en las facturas y, los presupuestos que realiza el negocio.
- **ClockIn:** Representa un fichaje dentro de un negocio. Cada empleado, independientemente de que el rol sea empleado o administrador, puede fichar en la aplicación indicando si es necesario un mensaje.
- **Contact:** Representación de un contacto que puede pertenecer a un negocio o cliente.
- **Cost:** Es la representación de un coste de un negocio, por lo que pueden ser compras o gastos que se tienen para ofrecer un servicio. No se tienen en cuenta las nóminas de los empleados puesto que estos tienen una entidad aparte.
- **CostType:** Indica el tipo de coste.
- **Customer:** Representa a un cliente dentro del sistema. Los datos del cliente son muy parecidos a los de un negocio.

- **DevelopmentLine:** Representa los datos de una tarea a realizar dentro de un presupuesto. Se compone de las horas de trabajo que cuesta de realizar, un precio y una descripción.
- **Employee:** Es la representación de un empleado de un negocio. Un usuario puede ser un empleado diferente en cada negocio.
- **EmployeeCost:** Representa a un gasto de empleado, es decir, la nómina.
- **Invoice:** Es la representación de una factura. Se da la opción de crear una a partir de un presupuesto.
- **InvoiceLine:** Es la representación de cada elemento a facturar.
- **Payment:** Representa un pago dentro del sistema.
- **PaymentMethod:** Indica el método de pago.
- **Quote:** Es la representación de un presupuesto en el sistema.

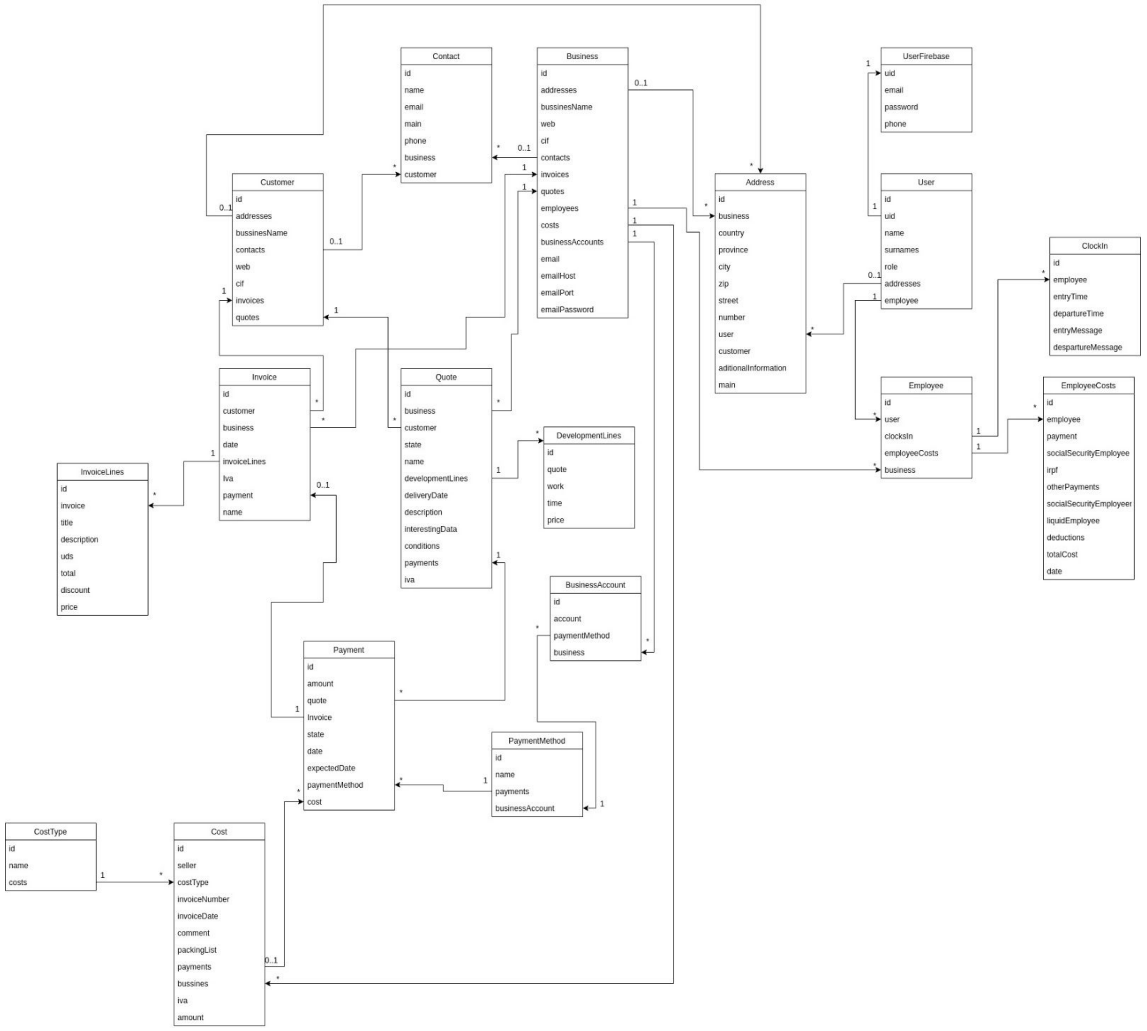

Figura 4: Diagrama de clases UML

5.2.2 Capa de presentación

Uno de los aspectos más importante en una aplicación web es la interfaz de usuario, puesto que sin esta no se podría interactuar con la aplicación. Para facilitar el posterior desarrollo de la aplicación se han diseñado una serie de prototipos con el objetivo de conseguir un diseño minimalista.

5.2.2.1 Inicio de sesión

Como se aprecia en la Figura 5, en la parte de arriba hay una barra con el título de la aplicación. A continuación, aparece un título que indica que la página actual es la de inicio de sesión, junto a dos inputs esenciales para la identificación: correo electrónico y contraseña. Por último, se incluye un enlace para recuperar la contraseña y un botón para validar las credenciales y acceder a la cuenta.

Figura 5: Boceto inicio de sesión

5.2.2.2 Listado entidad

A lo largo de la aplicación, la estructura seleccionada para listar las diferentes instancias de una entidad se repite en casi todas las entidades, por esta razón se hace un modelo general, como se muestra en la Figura 6.

En primer lugar, destaca la barra con el menú a la izquierda mediante el cual se podrá navegar por la aplicación. A continuación, aparece el nombre de la entidad, un buscador para poder filtrar los resultados por alguna palabra concreta y un botón para añadir un nuevo objeto de dicha entidad.

Para finalizar, destaca un listado compuesto por cada objeto que pertenece a dicha entidad. El cuerpo de cada objeto se puede dividir en dos partes, la primera es un botón a la derecha que despliega un menú con las diferentes acciones que permite realizar dicha entidad. Estas acciones pueden ser: ver, editar, eliminar, crear factura, enviar por correo y descargar. La segunda parte incluye un listado con los atributos del objeto que se consideran importantes para listarse.

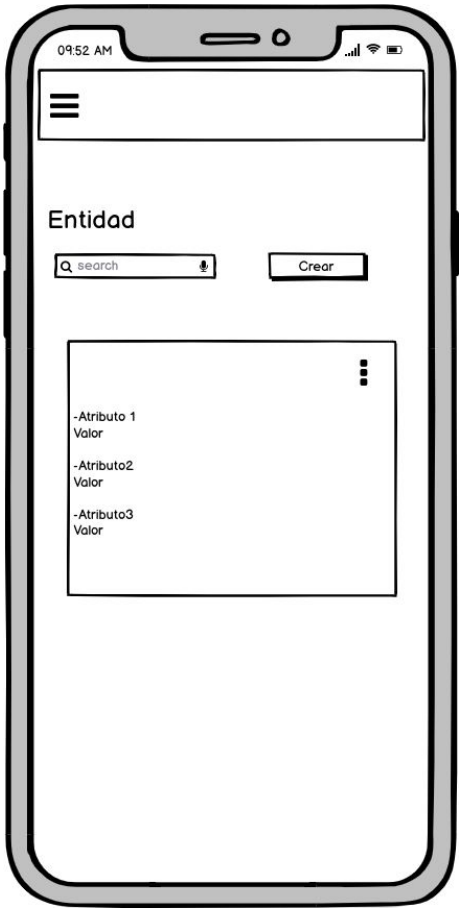

Figura 6: Boceto listado de entidad

5.2.2.3 Creación y edición de entidad

También en estas vistas se repite la misma estructura, por esta razón se divide en dos partes fundamentales: los atributos estáticos y atributos dinámicos.

Como se muestra en la Figura 7, en primer lugar, aparece un título seguido de una serie de atributos. El número de estos atributos no cambian. A continuación, se incluye una sección con una lista dinámica de atributos. En este tipo de sección se pueden añadir todos los atributos que sean necesarios, por ejemplo, en el caso de los clientes, se pueden añadir tantos contactos como sean necesarios.

Para finalizar, siempre se añade un botón para confirmar los cambios o la creación de la entidad.

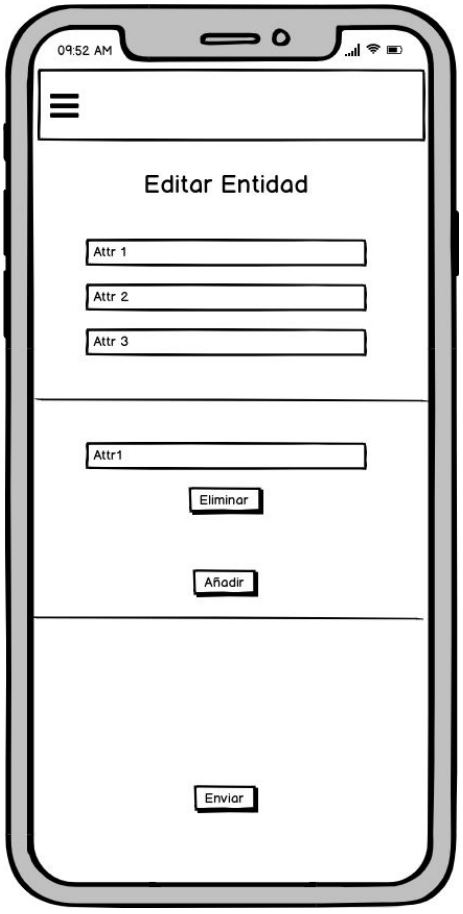

Figura 7: Boceto creación y edición de entidad

5.2.2.4 Inicio

En la página de inicio se muestra un resumen de los datos generales del sistema, por ejemplo: los beneficios, el total facturado, los gastos de los empleados, etc.

Como se muestra en la Figura 8, en primer lugar se incluye la opción de filtrar por año y negocio, dando la opción de mostrar unos datos generales o más específicos.

A continuación, se muestra un listado compuesto por el título de los datos que se muestran y un gráfico descriptivo.

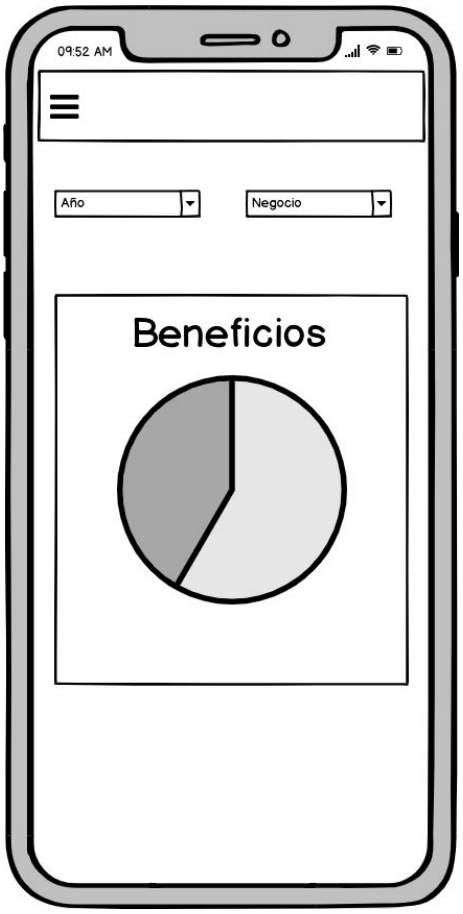

Figura 8: Boceto inicio

5.2.2.5 Fichar

Tanto los administradores como los empleados pueden acceder a esta página. Como muestra la Figura 9, en la primera parte se incluye una barra con el título de la aplicación y un botón para cerrar la sesión.

Es importante destacar que un usuario del sistema puede tener diferentes empleados, por ejemplo, en el caso de que un usuario trabaje en más de un negocio puede ser interesante crear un objeto de empleado para cada negocio.

A continuación, aparece el título de la página, junto a un botón para volver a la pantalla de administración el cual solo aparece si el rol del usuario es administrador. Y un botón para fichar que despliega un diálogo para añadir un mensaje si es necesario y, seleccionar uno de los empleados que están relacionados con el usuario que ha iniciado sesión para añadir el fichaje.

Para finalizar, se muestra un listado que permite seleccionar el empleado cuyos fichajes se desean visualizar.

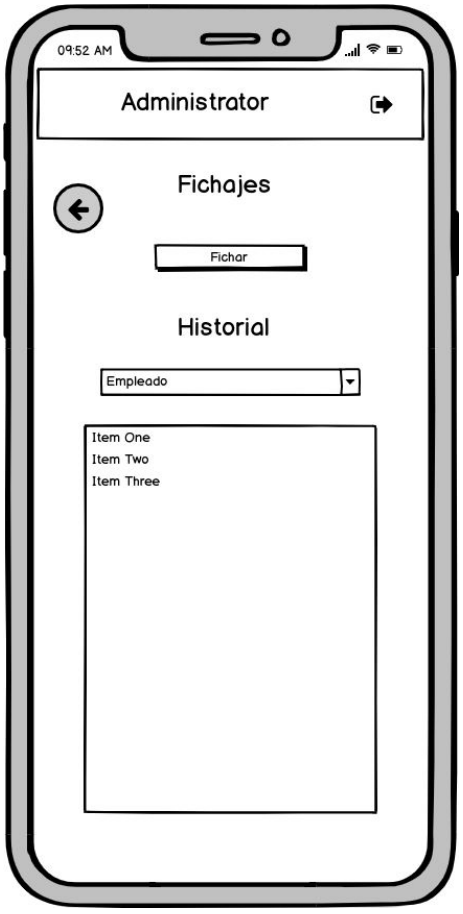

Figura 9: Boceto fichar

6 Desarrollo de la solución

Tras haber realizado el análisis del problema y diseñado una solución, a continuación se detalla el desarrollo de la misma, explicando cada una de las partes en las que se compone la aplicación.

6.1 Planificación

Con el propósito de organizar y estructurar el desarrollo se ha utilizado una metodología en cascada. Como en el periodo de desarrollo hay un número de tareas que se repiten, se han agrupado en 6 partes principales:

- **Inicio del proyecto:** Creación de las estructuras básicas del *frontend* y *backend*.
- **Inicio de sesión y securización:** Implementación del inicio de sesión y la securización básica que deniega el acceso a los usuarios no autentificados.
- **Listado de las entidades:** Creación de las páginas de listado para cada entidad.
- **Creación y edición de las instancias de las entidades:** Desarrollo de las páginas de creación y edición para cada entidad.
- **Fichajes e inicio:** Implementación de las páginas de fichajes de empleados y de la página de inicio.
- **Pruebas:** Desarrollo de tests y pruebas.

Como se muestra en la Figura 10, el primer bloque es el de inicio del proyecto que se tiene que desarrollar al principio ya que en este se construye la estructura en la que se basa el proyecto. Una vez se ha creado dicha estructura se procede a desarrollar las páginas relacionadas con el inicio de sesión y la securización de las páginas del *frontend* y de las API Endpoint.

A continuación, se implementan las páginas del listado de las entidades. Estas páginas están formadas principalmente por un componente que, según los datos que le proporcione cada página, ofrece unas acciones y muestra los resultados con una estructura determinada.

En el cuarto bloque se implementan las páginas para crear y editar las entidades. Éstas se agrupan en este bloque puesto que la estructura que se ha seguido para crear cada página es muy parecida.

En quinto lugar, se desarrollan las páginas relacionadas con los fichajes de los empleados, tanto la página para mostrar los fichajes como la propia página para fichar. También se crea la página de inicio que muestra un balance general del sistema.

Finalmente se realizan una serie de tests y pruebas una vez finalizado el desarrollo del proyecto con el objetivo de verificar su correcto funcionamiento.

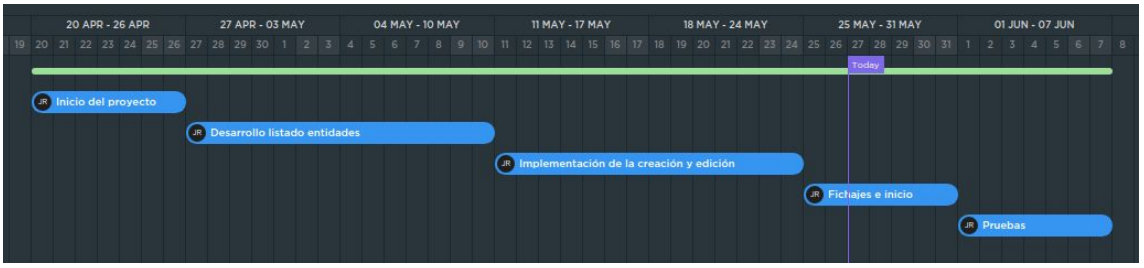

Figura 10: Diagrama de Gantt

6.2 Inicio del proyecto

En primer lugar, para comenzar el proyecto hay que tener las herramientas necesarias instaladas. Tanto para desarrollar el *frontend* como el *backend* se necesita el entorno de desarrollo Node.

6.2.1 Estructura *frontend*

Como se puede apreciar en la Figura 11, de todos los archivos que componen la estructura del *frontend* los más importantes son:

- **src:** Es la carpeta que contiene todo el cuerpo del proyecto, desde las páginas e imágenes hasta las traducciones.
- **src/assets:** En esta carpeta se almacenan imágenes y archivos js.
- **src/boot:** Contiene los archivos js que se ejecutan continuamente en la aplicación.
- **src/componentes:** Almacena los componentes que se utilizaran en las páginas y en los layouts.
- **src/css:** Guarda los archivos css que se utilizan en la aplicación.
- **src/i18n:** Se utiliza para almacenar las diferentes traducciones del sistema.
- **src/layouts:** Contiene todos los layouts que contendrán las páginas de la aplicación.
- **src/pages:** Almacena las páginas de la aplicación.
- **src/router:** Está compuesto por los archivos js que se emplean para manejar las rutas de la aplicación.
- **src/statics:** Se emplea para guardar imágenes estáticas en la aplicación.
- **src/store:** Se usa para almacenar la configuración para emplear Vuex.
- **src/App.vue:** Es el archivo principal de la aplicación.
- **.env:** Es el archivo que se utiliza para almacenar constantes que se pueden usar en toda la aplicación.
- **.eslintrc.js:** Es el archivo que se emplea para configurar ESlint.
- **package.json:** Contiene las dependencias del proyecto.
- **quasar.conf.js:** Es el archivo para la configuración de quasar.

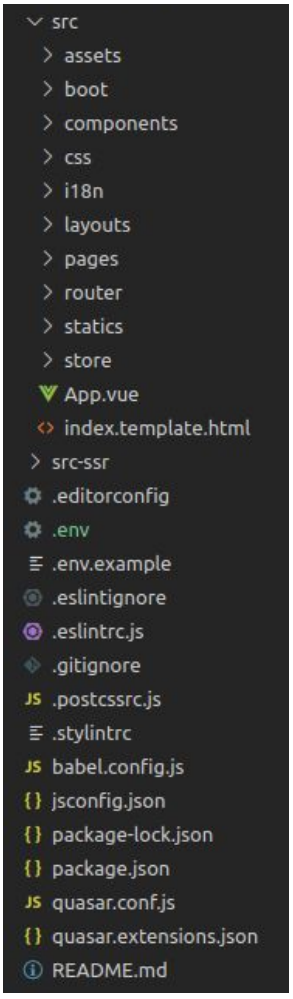

Figura 11: Estructura frontend

6.2.2 Estructura *backend*

Como se puede apreciar en la Figura 12, entre todas las carpetas y archivos que componen la estructura del *backend*, cabe destacar:

- **src:** Es la carpeta que contiene el cuerpo de la API.
- **src/assets:** En esta carpeta se almacenan los archivos que se consideran necesarios para la creación de archivos pdf, archivos html, css, imágenes, etc…
- **src/controllers:** Contiene los controladores que realizan funciones cuando se hace una petición a un API Endpoint.
- **src/locales:** Contiene las traducciones de la API.
- **src/middleware:** Contiene archivos ts que realizan funciones como por ejemplo las de autentificación para saber si quien realiza una petición está autentificado.
- **src/migrations:** Contiene las migraciones que se hacen a la base de datos.
- **src/models:** Está formado por los archivos que contienen a cada entidad y los cuales se transforman en las tablas de la BBDD.
- **App.ts:** Contiene la clase principal con la configuración necesaria para funcionar como API.
- **server.ts:** Es el archivo lanzadera, el cual importa los diferentes controladores, e inicia la aplicación con estos controladores.

- **swagger.json:** Este archivo contiene la configuración necesaria para crear una documentación y tests.
- **.env:** Es el archivo que contiene las diferentes constantes que se pueden emplear en toda la aplicación.
- **firebase.json:** Contiene la información necesaria para conectar la API con firebase.
- **ormconfig.json:** Es el archivo que contiene la configuración de TypeORM.
- **package.json:** Contiene las dependencias del proyecto.
- **tsconfig.json:** Contiene la configuración de TypeScript.

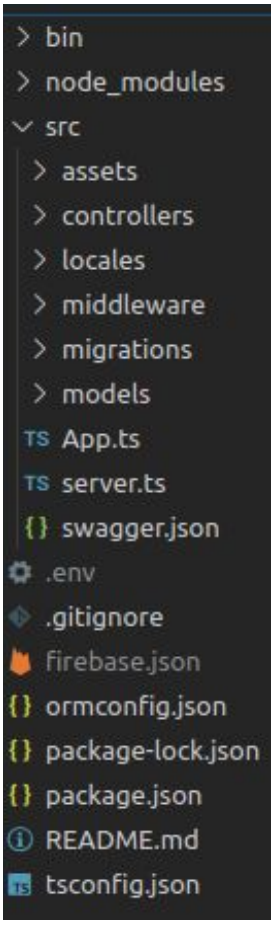

Figura 12: Estructura backend

6.3 Inicio de sesión y securización

Este apartado se va a dividir en 2 partes: En la primera parte se detallan las páginas relacionadas con el inicio de sesión y a continuación se realiza una descripción de la securización de la aplicación, tanto la que se ha aplicado en el *frontend* como en el *backend*.

6.3.1 Inicio de sesión y recuperación de contraseña

La primera página que se ha desarrollado es la de acceso a la aplicación puesto que es una página sencilla y fácil para comenzar.

La página de inicio de sesión se ha desarrollado con una barra superior con el título de la aplicación y un formulario en el centro que permite iniciar sesión de 2 formas:

- Mediante correo electrónico y contraseña.
- Mediante la cuenta de Google.

Cabe destacar que para realizar el acceso mediante la cuenta de Google se tuvo que reorganizar la base de datos y el *backend* para poder funcionar con firebase. Para poder utilizar firebase en el *frontend*, hay unas librerías pensadas con el propósito de ser utilizadas sólo desde éste, permitiendo comunicar el *frontend* directamente con firebase sin tener que pasar por el *backend*. En el caso del inicio de sesión, las dos formas de iniciar sesión realizan una petición directamente a firebase, esta petición, cuando es resuelta y las credenciales son correctas, devuelve una serie de datos entre los cuales hay un token que se utilizará posteriormente para identificar quién realiza la petición en el *backend*. En la Figura 13 se puede apreciar el resultado.

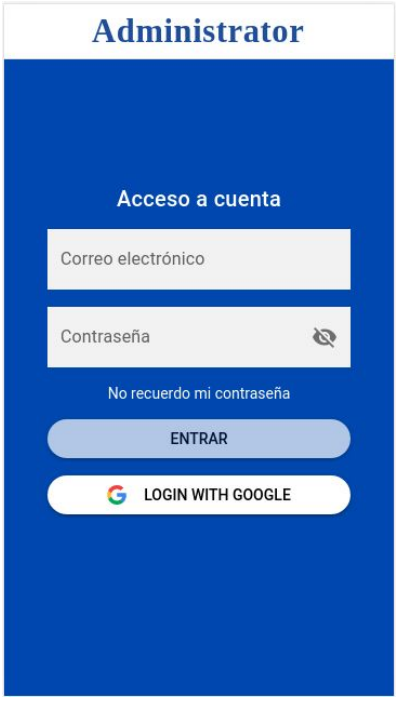

Figura 13: Inicio de sesión

En el caso de que no se recuerde la contraseña, si se realiza un click sobre el texto "No recuerdo mi contraseña", éste carga la página de recordar contraseña la cual sigue un diseño similar al de inicio de sesión.

Esta página se compone de un formulario para introducir el correo electrónico y el botón para enviar el formulario. Una vez se envíe el formulario, firebase enviará un mensaje con un enlace para poder cambiar la contraseña de la cuenta asociada a dicho correo electrónico.

A continuación, en la Figura 14 se puede apreciar el resultado.

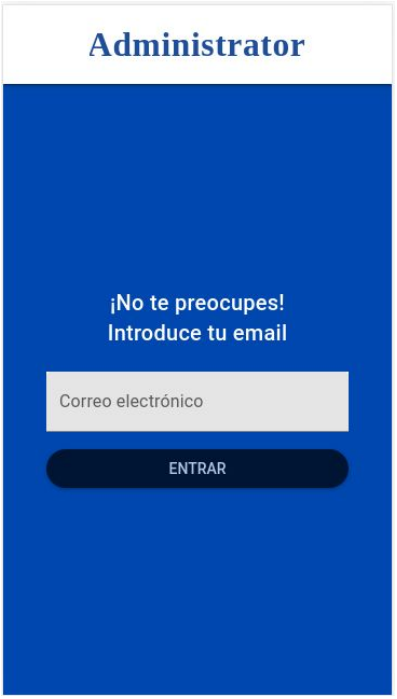

Figura 14: Recuperar contraseña

6.3.2 Securización

La securización se puede abordar de dos formas: en el *frontend* y en el *backend*.

6.3.2.1 Securización en el frontend

La securización en el *frontend* se realiza con el objetivo de no permitir el acceso a las páginas del administrador o de fichajes a un usuario no autenticado en la aplicación. Aunque un usuario halle la forma de saltarse esta seguridad, debido a que al realizar la petición al *backend* también se verifica el usuario, se dispone de una doble seguridad.

Para implementar esta seguridad, se ha añadido un código que cada vez que se quiere acceder a una página hace la siguiente comprobación:

- Si la ruta a la que se quiere acceder requiere estar autenticado y no se tiene un token de acceso, se redirige a la página de inicio de sesión.
- En caso contrario se deja acceder a cualquier ruta.

6.3.2.2 Securización en el backend

En el caso de que un usuario consiga saltarse la primera seguridad, la cual no comprueba si el token que dispone el usuario es válido, aún debería conseguir saltarse esta segunda barrera.

Cuando se hace una petición a un API Endpoint, este ejecuta una función en un controlador. Estas funciones se pueden securizar añadiendo una llamada a una función externa la cual se utiliza para realizar una verificación. En el caso de que no pase la verificación se devuelve el error correspondiente al origen de la petición. Por el contrario, si la verificación se pasa con éxito, se procede a continuar con la ejecución de la función. Este funcionamiento se comprende mejor observando la Figura 15:

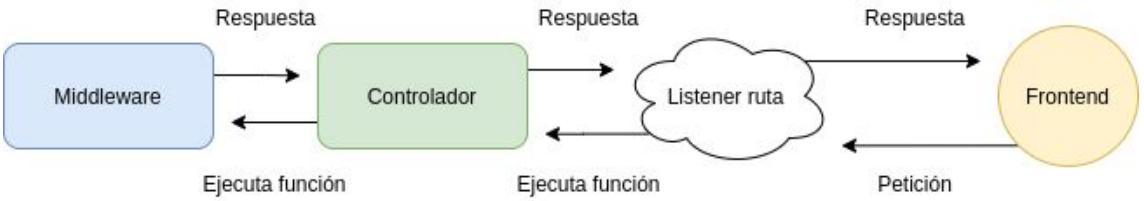

Figura 15: Flujo de una petición con verificación

En este caso, la verificación realiza una función que envía el token a firebase para que lo valide, esto puede dar lugar a 2 opciones:

- El token es correcto por lo que devuelve los datos del usuario, por lo que se procedería a continuar la ejecución de la función del controlador.
- El token es incorrecto, por lo que se devuelve el error.

6.3.3 Menú de navegación

En este subapartado se muestra el menú de navegación de la aplicación.

Como se aprecia en la Figura 16, la primera parte del menú de navegación se compone de un cuadro con el color principal de la aplicación. Dentro de este cuadro aparece el logo de la aplicación, el correo electrónico del usuario que ha iniciado la sesión y un botón para cerrar dicha sesión.

A continuación aparece un listado con las diferentes páginas a las que se puede acceder. Se han organizado diferentes páginas dentro de una sección para facilitar el uso y la simplicidad de la aplicación, haciendo que las páginas que se supone que se emplean menos, no se muestren.

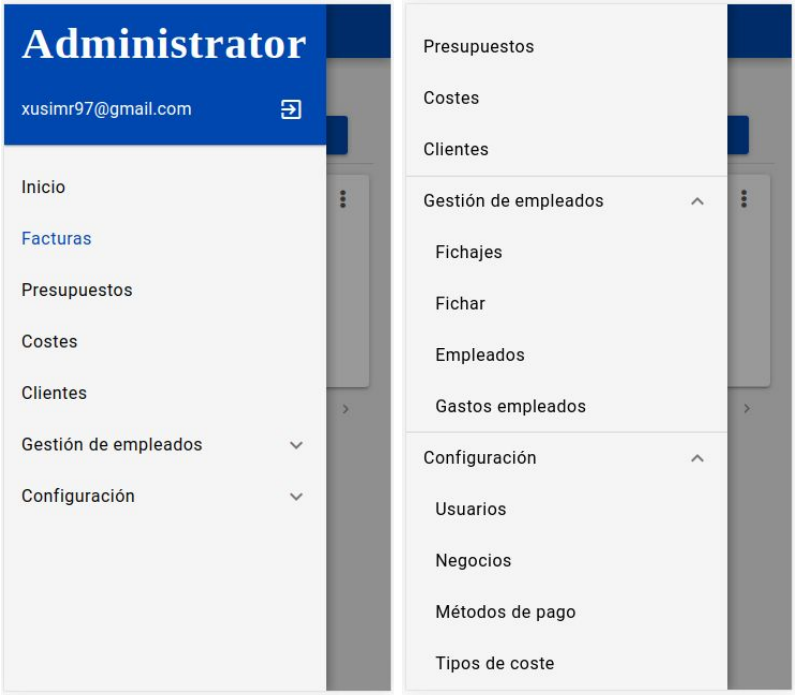

Figura 16: Menú de navegación

6.4 Listados de las entidades

En esta sección, se aborda la implementación que se ha seguido para desarrollar los listados de las entidades. Se debe tener en cuenta que cuando se lista una entidad, no solo se listan los atributos de ésta, sino que también puede darse el caso de mostrarse un atributo de una entidad relacionada con la primera. Por ejemplo, en el caso de las facturas, una factura se relaciona con un cliente, y en el listado se muestra el atributo businessName de la entidad cliente.

En primer lugar, se debe entender como organiza Quasar los elementos. Esto se consigue observando la Figura 17.

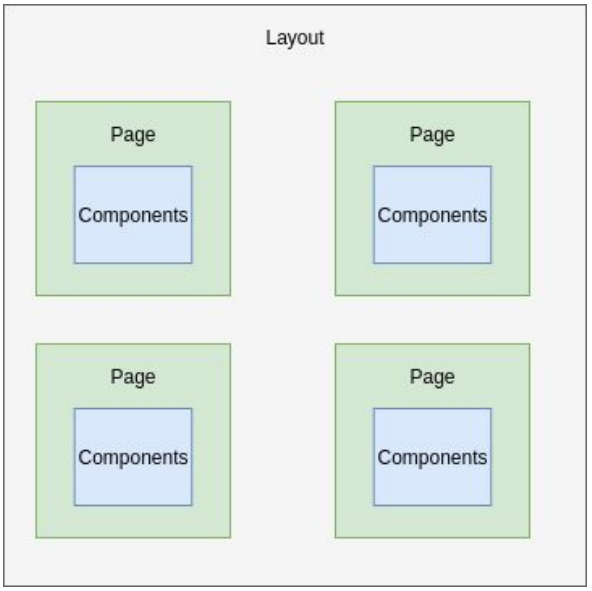

Figura 17: Jerarquía de elementos

Como se aprecia en la figura anterior, en primer lugar, se dispone de un *layout*. Este *layout* puede contener a su vez páginas o simplemente funcionar como si de una página se tratase. Por ejemplo, en este proyecto se han empleado 4 *layouts*:

- MyLayout.vue, que es el que contiene las páginas para la administración.
- ClockIn.vue, que es el que se utiliza para realizar los fichajes en la aplicación.
- Login.vue, que se emplea para el inicio de sesión.
- RememberPassword.vue, que se utiliza para obtener el email con el enlace para obtener una nueva contraseña.

Dentro de un *layout* se pueden mostrar diferentes páginas y, a su vez, dentro de cada página se pueden mostrar diferentes componentes. Estos componentes pueden ser los mismos que quasar ofrece por defecto o se pueden crear unos nuevos.

Como el desarrollo del listado es muy repetitivo para cada página y las variaciones suelen ser mínimas se ha desarrollado un componente llamado awesome-table que permite listar las instancias de una entidad ofreciendo un formato diferente para ordenadores y móviles.

La mayoría de las páginas que se emplean para listar las instancias de una entidad tienen la misma estructura, la diferencia radica en los datos que le pasan estas páginas al componente desarrollado. Los principales datos que se envían son:

- **Title:** Título que aparece en la tabla.
- **Url:** Url de la aplicación web donde tiene que redirigir la aplicación si se desea editar o crear una instancia de una entidad.
- **UrlApi:** Url de la API donde hacer las peticiones.
- **List:** Conjunto de atributos que se quieren mostrar en el listado de las instancias de una entidad.

También se pueden enviar los siguientes datos en entidades como las facturas o los presupuestos:

- **ToInvoice:** Valor booleano que indica si se quiere permitir la acción de convertir un pago de una instancia en una factura. Se emplea en los presupuestos.
- **Type:** Valor que indica el tipo de entidad.
- **Download:** Valor booleano que indica si se quiere permitir la acción de descargar un archivo pdf creado a partir de una instancia. Se emplea en las facturas y los presupuestos.
- **Mail:** Valor booleano que indica si se quiere permitir la acción de enviar un correo electrónico con un pdf creado a partir de una instancia. Se emplea en las facturas y los presupuestos.

Como se puede observar en la Figura 18, la parte de la izquierda es la versión de escritorio y la de la derecha es la versión de móvil, para el ejemplo de la entidad factura. En primer lugar, siempre se dispone del título del listado, un buscador para filtrar por palabras y el botón de añadir una nueva instancia.

A continuación, para cada instancia se muestra un botón con el icono de una elipsis vertical que despliega un menú para interactuar con dicha instancia y un listado de los atributos que se quieren mostrar en esta vista.

Para finalizar aparecen una serie de opciones que permiten mostrar el contenido, cambiando el número de instancias por página.

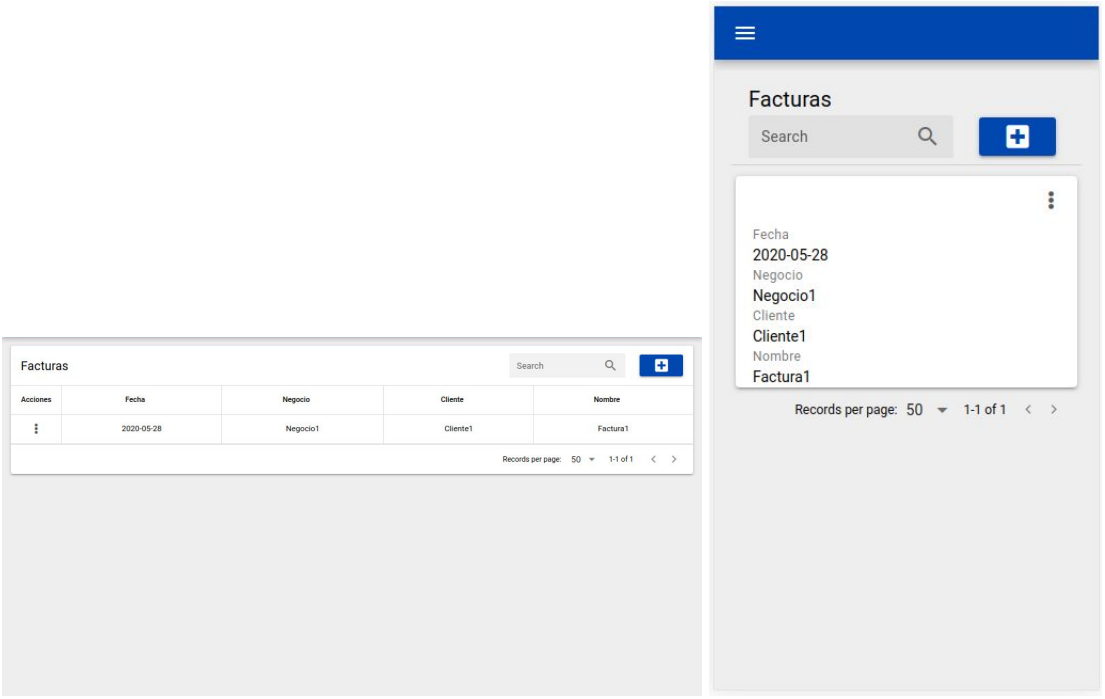

Figura 18: Listado de instancias de una entidad

6.5 Creación y edición de las instancias de las entidades

Para poder crear y editar una instancia de una entidad, se ha seguido una misma estructura en todas las páginas correspondientes, a continuación, se describen las diferentes partes que puede tener un formulario de creación/edición.

Pero antes cabe destacar que, cuando se crea o edita una instancia, también se editan las relaciones que tienen con las instancias de otra entidad, por lo que podremos añadir nuevas instancias de una entidad desde la página de edición de otra. Por ejemplo, en la creación de una factura, se pueden añadir instancias de la entidad InvoiceLine que representa el objeto que se factura.

6.5.1 Título y atributos propios de la instancia

En primer lugar, siempre se muestra un título indicando la acción que se está realizando y el nombre de la entidad con la que se va a interactuar. A continuación, se muestran un conjunto de atributos de la entidad. Se puede dar el caso de que aparezca un atributo que es una relación con otra entidad, pero solo permite seleccionar la instancia y no editarla.

En la Figura 19 se puede observar que el título es "Crear Factura”, a continuación se muestran un conjunto de atributos propios de la entidad factura:

- Nombre.
- Fecha, se puede añadir mediante un pop up que muestra un calendario.

- IVA, por defecto tiene el valor de 21.

También están los atributos Cliente y Negocio que representan una relación con otra instancia, estos se pueden añadir mediante una acción *select*.

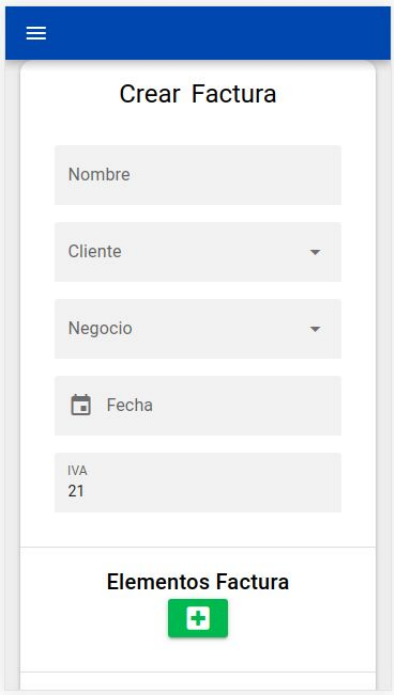

Figura 19: Creación de una instancia 1

6.5.2 Relación OneToMany

Este tipo de relaciones entre las instancias permite que se pueden añadir, editar y eliminar instancias de una entidad desde la creación/edición de otra.

Como se aprecia en la Figura 20, el título de la sección es “Elementos Factura” y contiene un botón que se utiliza para añadir las nuevas instancias.

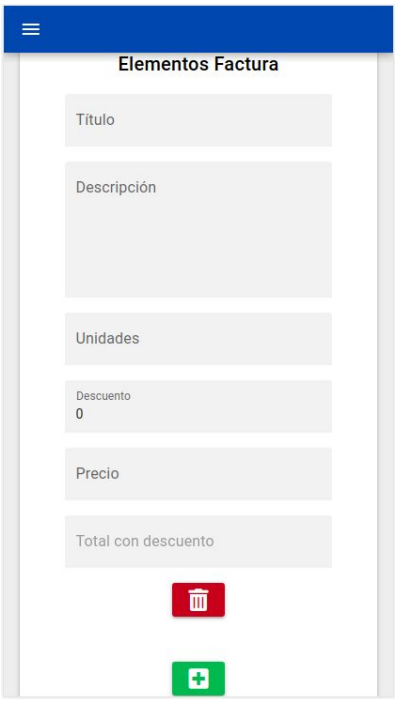

Figura 20: Creación de una instancia 2

Cada vez que se añade una nueva instancia, también se añade un botón por si se desea eliminar dicha instancia.

6.5.3 Relación OneToOne

Este tipo de relación entre las instancias de diferentes entidades se puede presentar en la aplicación dentro de una sección de la misma página como es en el caso de pago y factura. Dentro de la creación de una factura se pueden editar los datos del pago asociado.

Como se observa en la Figura 21, se ha añadido un watcher que calcula automáticamente el valor del pago total que hay que incluir, en el caso de que haya un error, muestra el valor de NaN.

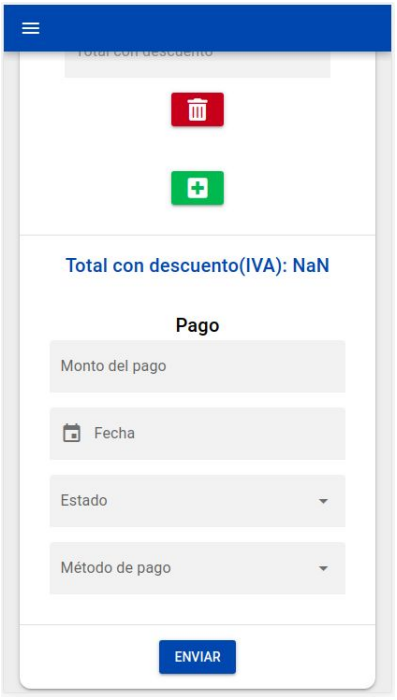

Figura 21: Creación de una instancia 3

6.6 Fichajes e inicio

Para finalizar el capítulo del desarrollo se han elaborado 2 secciones importantes de la aplicación, en primer lugar la parte correspondiente a los fichajes de los empleados y por último la página de inicio que muestra los balances que registra la aplicación.

6.6.1 Fichajes

Hay dos páginas relacionadas con los fichajes, la primera se emplea para poder registrar un fichaje y listar los relacionados con dicho usuario. La segunda página se emplea para poder listar los fichajes de todos los empleados.

6.6.1.1 Registrar fichaje

La página para registrar un fichaje está formada por una barra superior con el logo y un botón para cerrar la sesión. A continuación, muestra un botón que despliega un diálogo que permite realizar el fichaje enviando también, si se desea, un mensaje.

Por último, hay una lista que permite ver el registro de fichajes que se han realizado por parte del usuario. A continuación, en la Figura 22 se puede ver el resultado:

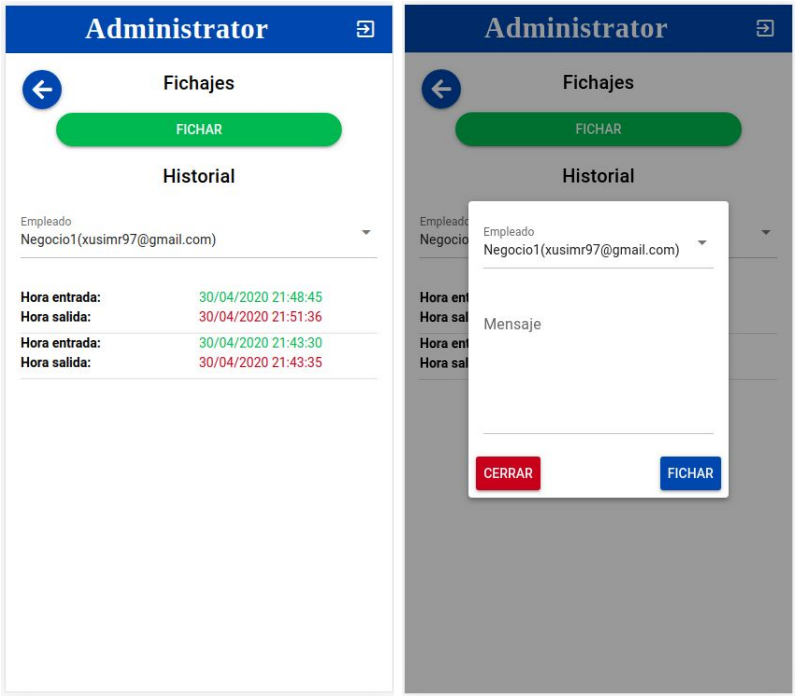

Figura 22: Registrar fichajes

Para realizar la página de fichajes se ha tenido en cuenta que esta página deben de poder ser accedida tanto por administradores como por empleados por lo que, en el caso de un administrador, se deberá poder volver a las páginas de la administración. Por esta razón, en la izquierda de la pantalla aparece un botón con una flecha que solo se ve si el usuario que ha iniciado sesión es un administrador.

6.6.1.2 Listar fichajes

Para poder listar los fichajes de los empleados se ha seguido una estructura diferente a la usada en las anteriores páginas. En primer lugar, se realiza un filtro indicando el negocio y el intervalo de fechas entre las que listar los fichajes.

Una vez obtenidos los resultados, se distribuyen en los días del intervalo filtrado y dentro de cada día se muestran los diferentes empleados. A su vez, dentro de cada empleado se pueden ver los registros que ha realizado ese día. En la Figura 23 se muestra el resultado:

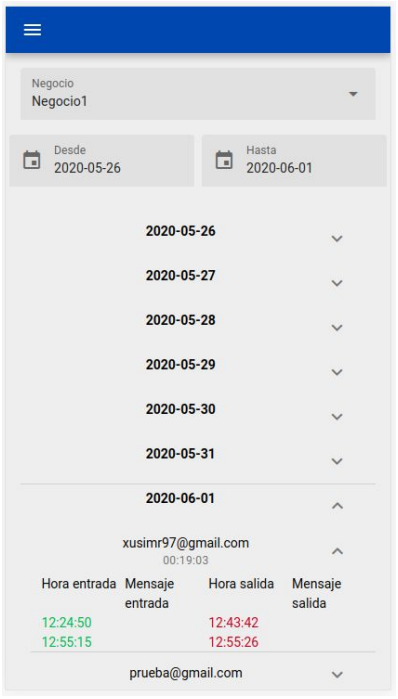

Figura 23: Listar fichajes

Como se observa en la figura anterior, debajo de cada correo electrónico en el caso de que el empleado haya registrado horas se muestra el tiempo total que ha estado trabajando ese día.

6.6.2 Inicio

La página de inicio está diseñada para emplearse como una vista general de la administración del sistema. Esta página solo sirve para mostrar datos y no se puede realizar ninguna acción que no tenga el objetivo de visualizar datos, en la Figura 24 se muestra el resultado.

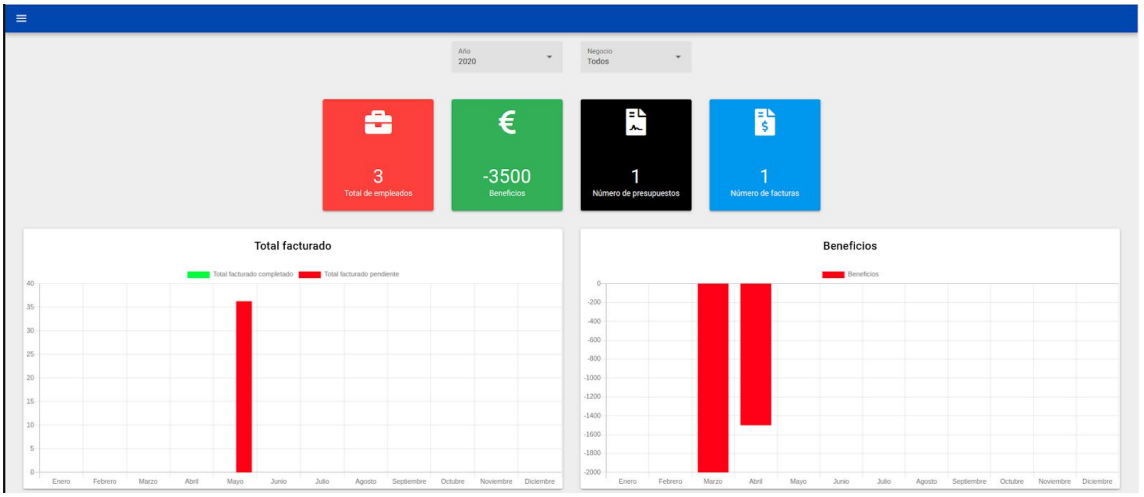

Figura 24: Inicio 1

En primer lugar se presenta un filtrado para poder ver los resultados clasificados por año y según el negocio seleccionado. A continuación, hay 4 apartados que representan:

- El número de empleados.

- Los beneficios.
- El número de presupuestos.
- El número de facturas.

La última sección se compone de gráficas que desglosan los datos por meses, de esta forma se puede saber por ejemplo el total de beneficios que se han obtenido cada mes.

La vista de la tabla cambia si se visualiza desde un móvil, en este caso los datos se mostrarán como indica la Figura 25:

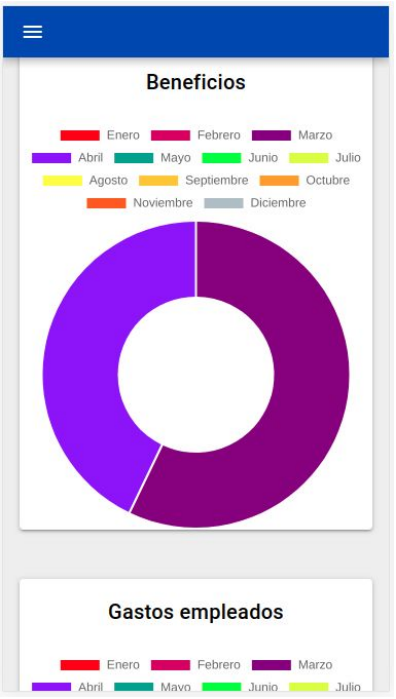

Figura 25: Inicio 2

7 Pruebas

Una de las etapas más importantes en el desarrollo de un proyecto es la realización de las pruebas para garantizar su corrección según las especificaciones dadas. En esta etapa se diseñan un conjunto de pruebas para comprobar si el funcionamiento es el esperado. Para la realización de esta sección, se han seguido estas 3 fases:

- Diseño de las pruebas.
- Ejecución de las pruebas.
- Corrección de errores.

El plan completo de pruebas se muestra en el Anexo de esta memoria.

7.1 Requisitos de instalación

Para poder instalar la aplicación web en el servidor, se debe de tener instalado y actualizado:

- Node.js.
- NPM.
- Git.

7.2 Diseño de las pruebas

Para realizar el diseño de las pruebas se ha elaborado un plan en el que se realiza una descripción minuciosa de cada prueba. Cada prueba tiene que estar pensada con el objetivo de evitar en la medida de lo posible los errores más comunes que pueden ocurrir.

Para diseñar cada prueba, se han identificado 4 partes principales:

- **Objetivo:** Determina cuál es el objetivo que se pretende alcanzar con la realización de la prueba.
- **Requisitos:** Indica qué condiciones se deben cumplir para la realización.
- **Pasos a seguir:** Se listan los pasos que se deben seguir.
- **Criterio de validez:** Se especifica cual es el resultado esperado para considerar que la prueba ha sido un éxito.

7.3 Ejecución de las pruebas

Tras haber desarrollado las pruebas se procede a ejecutarlas. Si no se obtiene el resultado esperado hay que realizar una búsqueda del error para resolverlo posteriormente.

La realización de estas pruebas se ha hecho con personas que no conocen el funcionamiento de la aplicación para verificar que un usuario sin conocimiento pueda hacer uso de la misma. También se ha tenido en cuenta la realización de las pruebas desde diferentes dispositivos para garantizar que no hay errores si se usa desde un ordenador o desde un teléfono móvil.

7.4 Corrección de errores

Una vez se han realizado las diferentes pruebas, se han recopilado los errores más comunes que ha habido a lo largo de la fase de pruebas y que se han tenido que solucionar. Los problemas que se han presentado han sido los siguientes:

- **Adaptabilidad:** En ciertas páginas ha habido componentes que no se han mostrado correctamente al visualizarse en la versión móvil, por lo que se ha editado el código para que se adapte correctamente.
- **Lógica de edición y creación:** Cuando se pretendía editar o añadir una instancia no se realizaban bien las comprobaciones y saltaba un error.
- **Listado:** En el momento de listar una entidad, cuando se pretendía listar los atributos de una relación daba errores por lo que había que ir editando el código para que se adaptara a los diferentes casos.

Todos los problemas presentados han sido resueltos correctamente y ya no se presentan en la versión actual de la aplicación.

8 Conclusiones

En este capítulo se detallan las conclusiones obtenidas una vez se ha completado el desarrollo de la aplicación. Estas conclusiones están basadas en los objetivos propuestos en el apartado 1.2, analizado si se han podido cumplir y los problemas que se han afrontado.

En primer lugar, hay que destacar que se ha conseguido el objetivo principal propuesto dando lugar a una aplicación web que permite administrar de una forma sencilla ciertos aspectos de una empresa.

El primer objetivo corresponde al **desarrollo de un *frontend* que se sirva de los servicios de la API REST**. Para poder llevar a cabo este objetivo se ha tenido que desarrollar una lógica que permite realizar peticiones a la API y controlar los errores ocurridos.

El siguiente objetivo alcanzado es **desarrollo de un *backend* que proporcione los servicios para administrar un negocio.** Crear la API ha sido un proceso complejo con muchas horas de desarrollo de código para ofrecer una buena lógica en el momento de administrar entidades como las facturas. El mayor reto ha sido controlar todos los errores que se pueden producir al interactuar con la base de datos.

El tercer objetivo consistía en **permitir trabajar con facturas y presupuestos.** Para poder cumplir con este objetivo se ha considerado que se debía poder realizar las siguientes acciones: crear, editar, ver, filtrar, eliminar, descargar, enviar al cliente y crear una factura a partir de un presupuesto. Sin lugar a dudas, conseguir este objetivo ha sido el más difícil de todos por la extensa cantidad de acciones que había que proveer.

Uno de los objetivos más sencillos ha sido el de **administrar los gastos e ingresos de los negocios.** Para realizar este objetivo solo había que emplear la estructura ya usada para el objetivo de las facturas, pero aun así ha requerido de un cierto trabajo para adaptarse a las situaciones propias.

El quinto objetivo marcado era **controlar el fichaje de los trabajadores.** Se ha tenido que desarrollar una lógica que permitiese fichar tanto a empleados como a administradores. El mayor reto de este objetivo ha sido reestructurar los datos para que se visualicen en el formato seleccionado.

Para cumplir con el objetivo de **visualizar los datos de la aplicación desde una vista general,** se ha desarrollado la página de inicio la cual recoge los datos más destacables del sistema como los beneficios, número de facturas, número de empleados, etc.

Para finalizar, con el objetivo de **desarrollo de una aplicación utilizable desde cualquier dispositivo** se ha diseñado la aplicación primero para el dispositivo móvil haciendo que se adapte al tamaño de la pantalla permitiendo mostrar desde móvil a ordenador.

En relación con la planificación temporal propuesta en la sección 1.3, hay que añadir que no ha habido ningún desvío y se ha podido completar el desarrollo con éxito dentro del plazo acordado, requiriendo 7 semanas para terminar la aplicación web.

8.1 Relación del trabajo desarrollado con los estudios cursados.

Con este trabajo el autor finaliza el estudio del grado de Ingeniería Informática. Este trabajo le ha permitido adquirir un buen número de conocimientos adicionales que se han aplicado directamente en el desarrollo del mismo. La aplicación web resultante ha sido programada desde cero y aunque los lenguajes no sean los mismos sí que lo son las bases empleadas.

Las asignaturas del plan de estudios que han sido especialmente de ayuda a la hora de desarrollar el presente trabajo han sido:

- Gestión de proyectos.
- Ingeniería del software.
- Sistemas y servicios en red.
- Desarrollo centrado en el usuario.
- Desarrollo web.

8.2 Trabajos futuros

A continuación, se presentan las posibles mejoras y correcciones que se podrían aplicar en futuras versiones:

- Una primera mejora que se plantea es sincronizar la aplicación con Jira y poder así administrar desde la misma los diferentes proyectos e incidencias de Jira, centralizando las diferentes herramientas.
- Otra mejora es modificar la página de inicio para mostrar más datos relevantes de la aplicación.
- También se podría añadir nuevas vistas para mostrar, por ejemplo:
    - La facturación total de cada cliente y sus pagos pendientes.
    - Facturación desglosada de cada cliente.
- Finalmente se podrían añadir funcionalidades como sincronizarse con Google Analytics para visualizar los detalles de las webs de los negocios, realización de email Marketing, …

9 Referencias

Adriana Carolina Leal (2018), ¿Qué es un software de gestión?.
Disponible:https://www.siigo.com/blog/empresario/software-de-gestion/

Javier Díaz (2013), 20 Aplicaciones web para administrar una empresa.
Disponible:https://www.negociosyemprendimiento.org/2013/01/aplicaciones-web-para-administrar-empresa.html

Rafael Barzanallana (2012). Historia del desarrollo de aplicaciones Web.
Disponible:https://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Historia-desarrollo-aplicaciones-web.html

Developer mozilla (2019), Frameworks Web de lado servidor.
Disponible:https://developer.mozilla.org/es/docs/Learn/Server-side/Primeros_pasos/Web_frameworks

Stack Overflow (2019),Most popular technologies: Web Frameworks.
Disponible:https://insights.stackoverflow.com/survey/2019#technology-_-web-frameworks

Quasar Framework (2020), What is SPA.
Disponible:https://quasar.dev/quasar-cli/developing-spa/introduction

Gonzalo Méndez (1998), Especificación de Requisitos según el estándar de IEEE 830.
Disponible:https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

Vue.js (2020), The Progressive JavaScript Framework.
Disponible:https://vuejs.org

10 Anexo

10.1 Plan de pruebas

Objetivo:	Iniciar sesión
Requisitos:	• Disponer de un usuario registrado en la aplicación
Pasos a seguir:	• Acceder a la página de inicio de sesión • Iniciar sesión con cualquiera de los 2 métodos
Criterio de validez:	El usuario accede a la aplicación y dependiendo del rol del usuario accede a la página para registrar horas o a la vista de administradores

Objetivo:	Fichar
Requisitos:	• Haber iniciado sesión
Pasos a seguir:	• Acceder a la página de fichar • Seleccionar el empleado • Introducir si se desea un mensaje antes de hacer click en el botón de fichar
Criterio de validez:	Si se realiza el fichaje correctamente y aparece por pantalla el nuevo registro

Objetivo:	Listar fichajes
Requisitos:	• Haber iniciado sesión como administrador
Pasos a seguir:	• Acceder a la página de fichajes • Seleccionar las fechas entre las que se desea listar los fichajes
Criterio de validez:	Si se muestran los fichajes existentes de cada día

Objetivo:	Mostrar los balances del sistema
Requisitos:	• Haber iniciado sesión como administrador
Pasos a seguir:	• Acceder a la página de inicio • Realizar un filtrado por año o por negocio
Criterio de validez:	Si se han mostrado los resúmenes de las cuentas y rellenado las tablas correctamente.

Objetivo:	Descargar presupuesto
Requisitos:	• Haber iniciado sesión como administrador • La entidad debe tener alguna instancia para poder descargar
Pasos a seguir:	• Acceder a la página de listado de presupuestos • Hacer click sobre el botón de acciones de una instancia • Seleccionar descargar
Criterio de validez:	Si se ha descargado un fichero pdf con los datos del presupuesto

Objetivo:	Descargar factura
Requisitos:	• Haber iniciado sesión como administrador • La entidad debe tener alguna instancia para poder descargar
Pasos a seguir:	• Acceder a la página de listado de facturas • Hacer click sobre el botón de acciones de una instancia • Seleccionar descargar
Criterio de validez:	Si se ha descargado un fichero pdf con los datos de la factura

Objetivo:	Enviar presupuesto al cliente
Requisitos:	• Haber iniciado sesión como administrador • La entidad debe tener alguna instancia para poder enviar al cliente • El negocio debe haber configurado los datos del correo electrónico
Pasos a seguir:	• Acceder a la página de listado de presupuestos • Hacer click sobre el botón de acciones de una instancia • Seleccionar enviar • Escoger un título y un texto para el correo
Criterio de validez:	Si aparece el mensaje de que el correo se ha enviado correctamente

Objetivo:	Enviar factura al cliente
Requisitos:	• Haber iniciado sesión como administrador • La entidad debe tener alguna instancia para poder enviar al cliente • El negocio debe haber configurado los datos del correo electrónico
Pasos a seguir:	• Acceder a la página de listado de facturas • Hacer click sobre el botón de acciones de una instancia • Seleccionar enviar • Escoger un título y un texto para el correo
Criterio de validez:	Si aparece el mensaje de que el correo se ha enviado correctamente

Para cada página que se sirve del componente awesome-table, sigue la misma estructura y objetivo, por lo que las siguientes pruebas se aplican a todas las páginas correspondientes:

Objetivo:	Listar
Requisitos:	• Haber iniciado sesión con un usuario administrador • La entidad debe tener alguna instancia para poder listar
Pasos a seguir:	• Acceder a la página de listado
Criterio de validez:	En el caso de que se rellena la tabla y no se muestre ningún error.

Objetivo:	Filtrar
Requisitos:	• Haber iniciado sesión con un usuario administrador • La entidad debe tener más de una instancia para poder filtrar
Pasos a seguir:	• Acceder a la página de listado • Ejecutar una acción ya sea filtrar por palabra, atributo o usando la paginación.
Criterio de validez:	Si el filtrado se realiza correctamente y no salta ningún error

Objetivo:	Eliminar
Requisitos:	• Haber iniciado sesión con un usuario administrador • La entidad debe tener al menos una instancia para poder eliminar
Pasos a seguir:	• Acceder a la página de listado • Hacer click sobre el botón de acciones de una instancia • Seleccionar eliminar • Aceptar el diálogo para proceder con la eliminación
Criterio de validez:	Se ha eliminado correctamente la instancia

Objetivo:	Añadir
Requisitos:	• Haber iniciado sesión con un usuario administrador
Pasos a seguir:	• Acceder a la página de listado • Hacer click sobre el botón de añadir instancia • Rellenar los campos de la instancia • Clickar sobre el botón de enviar
Criterio de validez:	Si no aparece ningún error y en la página de listado aparece la nueva instancia.

Objetivo:	Editar
Requisitos:	• Haber iniciado sesión con un usuario administrador • La entidad debe tener al menos una instancia
Pasos a seguir:	• Acceder a la página de listado • Hacer click sobre el botón de acciones de una instancia • Seleccionar editar • Editar los campos deseados de la instancia • Clickar sobre el botón de enviar
Criterio de validez:	Si no aparece ningún error y en la página de listado aparecen los cambios de la instancia en el caso de que se listen los atributos editados.