MEMORIA DEL PROYECTO

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

GESTIÓN DE CENTROS COMPUTACIONALES

Junio de 2022

Autor

Álvaro García García

Tutores

Gabriel Villarubia González

André Felipe Sales Mendes

Héctor Sánchez San Blas

CERTIFICADO DE LOS TUTORES

D. Héctor Sánchez San Blas, D. André Felipe Sales Mendes, D. Gabriel Villarrubia González profesores/as del Departamento de Informática y Automática de la Universidad de Salamanca

CERTIFICAN:

Que el trabajo titulado “Gestión de centros computacionales” ha sido realizado por D. Álvaro García García, con DNI 45138930G y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 4 de julio de 2022

SANCHEZ SAN BLAS HECTOR - 70901148Z
Digitally signed by SANCHEZ SAN BLAS HECTOR - 70901148Z
Date: 2022.07.04 12:19:06 +02'00'

D. Héctor Sánchez San Blas

SALES MENDES ANDRE FILIPE - Y5601546W
Firmado digitalmente por SALES MENDES ANDRE FILIPE - Y5601546W
Fecha: 2022.07.04 17:46:56 +02'00'

D. André Felipe Sales Mendes

VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2022.07.04 12:23:31 +02'00'

D. Gabriel Villarrubia González

RESUMEN

El presente trabajo consiste en la elaboración de un sistema de monitorización y gestión de centros computacionales, para ello la mayor parte de la funcionalidad se lleva a cabo desde una aplicación web que puede utilizarse desde dispositivos medianos como pueden ser tablets a dispositivos grandes como PC y portátiles, siempre y cuando dispongan de un navegador web. Para conseguir mayor compatibilidad con navegadores se ha implementado lo que se conoce como una Web App Progresiva, que en resumen es una aplicación normal que no necesita ser instalada y que se ejecuta en los navegadores web, para esto se utilizó la tecnología Vue.

La aplicación web nos permite gestionar empresas que son dadas de alta y gestionadas por usuarios con el rol de super usuarios. Cada una de las empresas a su vez contará con diferentes trabajadores, que podrán tener el rol de administradores de la propia empresa y el resto de los empleados serán usuarios normales. Los administradores serán los encargados de controlar la monitorización de los equipos de los usuarios normales, así como realizar la gestión de estos dentro de su propia empresa, pudiéndolos dar de alta, eliminarlos, etc. También proporciona la posibilidad a los administradores de crear alertas que se dispararán en base a ciertos parámetros tras la lectura de información de los equipos de la empresa, pudiendo llevar un mayor control de cómo están funcionando y de ser atendidos en caso de que sea necesario. Además, los usuarios normales tienen la posibilidad de crear tickets de incidencias que serán recibidos por los administradores de su empresa de tal forma que pueda haber una comunicación directa con los mismos cuando lo necesiten.

Para la monitorización se ha programado un script encargado de recoger información relevante de los equipos. Para ello, los usuarios deberán ejecutar el script en su máquina y este almacenará dicha información para compartirla con los administradores.

Para realizar la gestión de usuarios, así como el almacenamiento y el hosting de la aplicación web se ha hecho uso de framework Firebase, que es un servicio de Google compatible con Vue.

La finalidad de este proyecto es la elaboración de un sistema capaz de ofrecer un servicio a diferentes empresas que permite a sus administradores tener un control sobre el estado de los equipos de la misma, estén donde estén, sin necesidad de tener que instalar y configurar un software más complejo en cada uno de los equipos. De esta forma con tan solo ejecutar el script, el administrador podrá obtener la información que necesita de los diferentes equipos y los usuarios normales podrán tener una vía de comunicación directa con los administradores.

Palabras clave: Monitorización, Gestión, Alertas, Avisos, Incidencias, Empresas, Administrador, Super Usuario.

SUMMARY

This work consists of the development of a monitoring and management system for computer centers, for this most of the functionality is carried out from a web application that can be used from medium devices such as tablets to large devices such as PCs and laptops, as long as they have a web browser. To achieve greater compatibility with browsers we have implemented what is known as a Progressive Web App, which in short is a normal application that does not need to be installed and that runs in web browsers, for this we used Vue technology.

The web application allows us to manage companies that are registered and managed by users with the role of super users. Each of the companies in turn will have different workers, who may have the role of administrators of the company itself and the rest of the employees will be normal users. The administrators will be in charge of controlling the monitoring of the equipment of the normal users, as well as managing them within their own company, being able to register them, delete them, etc. It also provides the possibility for administrators to create alerts that will be triggered based on certain parameters after reading information from the company's equipment, being able to have a better control of how they are working and to be attended in case it is necessary. In addition, normal users have the possibility of creating incident tickets that will be received by the administrators of their company so that there can be a direct communication with them when they need it.

For monitoring, a script has been programmed to collect relevant information from the equipment. To do this, users must run the script on their machine, and it will store this information to share it with administrators.

Firebase framework, a Google service compatible with Vue, has been used to manage users, as well as to store and host the web application.

The purpose of this project is the development of a system capable of providing a service to different companies that allows their administrators to have control over the status of the equipment of the same, wherever they are, without having to install and configure more complex software on each of the computers. In this way, by simply executing the script, the administrator will be able to obtain the information he needs from the different computers and the normal users will be able to have a direct communication channel with the administrators.

Keywords: Monitoring, Management, Alerts, Warnings, Incidents, Companies, Administrator, Super User.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

En la actualidad hemos visto como ha incrementado la cantidad de empresas que han optado por el teletrabajo, y que han seguido manteniéndolo una vez pasada la pandemia, lo cual dificulta el hecho de tener un administrador “on site”. Muchas de estas empresas no tienen por qué necesitar de complejos sistemas de control o disponer de una intranet mediante la cual mantener todos los equipos conectados entre sí.

Eventualmente este tipo de empresas contactan con profesionales informáticos para que den solución problemas que podrían haberse resuelto con una simple consulta u observando cierta información relevante y que quizás no hubiesen necesitado de un desplazamiento para trabajar “on site”. Imaginemos una pequeña empresa con un servidor central montado con apache2 y un servidor sql para la base de datos. Si el equipo donde se encuentra alojado dicho servidor enviase información relevante, un administrador podría detectar una caída al instante, una bajada en el rendimiento de la máquina, perdidas de conectividad…

Por otra parte, conocer ciertos aspectos sobre cuál es el estado del procesador, de la ram, del disco duro, etc, nos permite tener una idea general de cuál es el estado de los equipos con los que se trabaja en la empresa, y de qué podría ser necesario cambiar o reparar, permitiéndonos prever o evitar fallos y errores inesperados.

Además, llevar un control sobre los dispositivos usb que se encuentra conectados en un equipo, puede proporcionar seguridad ya que pueden ser entrada directa de “malware” así como detectar cuando un dispositivo está teniendo problemas para conectarse.

El presente sistema, permite controlar la información de los equipos computacionales utilizados en una empresa, dotando a los administradores de la posibilidad de conocer cuál es el estado de los equipos.

En el caso de que los usuarios necesiten, además de asistencia directa de un administrador sobre su ordenador, podría usarse una herramienta auxiliar, como un programa de control de escritorio remoto del estilo a TeamViewer o AnyDesk, o también se podría optar por el uso de herramientas de creación de redes locales virtuales por VPN, como RadminVPN u OpenVPN, lo cual crearía una conexión directa entre los trabajadores y los administradores.

Los aspectos más relevantes durante el desarrollo y a explicar en esta memoria serán los siguientes:

- **Análisis del mercado:** Aplicaciones ya existentes cuya función u objetivos son parecidos.
- **Objetivos:** Objetivos a conseguir con el desarrollo del sistema.
- **Conceptos teóricos:** Marco teórico en base al cual se desarrolló el sistema software.
- **Técnicas y herramientas:** Diferentes tecnologías utilizadas para la elaboración de la aplicación.
- **Aspectos relevantes del desarrollo del proyecto:** El proceso seguido para el desarrollo.
- **Limitaciones encontradas:** Algunas de las limitaciones que se han encontrado durante el desarrollo de la aplicación.

- **Conclusiones y líneas de trabajo futuras:** Conclusiones obtenidas al finalizar el proyecto, así como un comentario sobre cómo se ve la aplicación en un futuro.
- **Referencias.**

2 ANÁLISIS DEL MERCADO

Antes de comenzar con el proyecto se hizo un estudio de diferentes tecnologías ya existentes que trabajan alrededor del mismo campo y que poseen funcionalidades u objetivos similares a los que se buscaba lograr con el desarrollo del proyecto, a continuación, se muestran un par de ellas que fueron las que se consideraron más relevantes.

2.1 PRTG

Es principalmente una herramienta de monitorización de redes, que permite el análisis de las distintas redes o “intranets” de las empresas, aunque además permite monitorizar información referente al hardware de los equipos y crear alertas que informen a los administradores bajo determinados parámetros. Esta herramienta, aunque funciona o se controla desde el navegador web es necesario instalarla en la máquina y una vez instalada, arrancar un servidor local al que se accede mediante el navegador.

Figura 1 Ejemplo de una vista de PRTG

2.2 NAGIOS

Nagios también es un sistema de control y monitorización de redes, en este caso de código abierto que permite controlar los servicios y los equipos de una red, y producir alertas cuando sus comportamientos no sean los adecuados. En el caso de Nagios también sería necesaria una previa instalación para su funcionamiento.

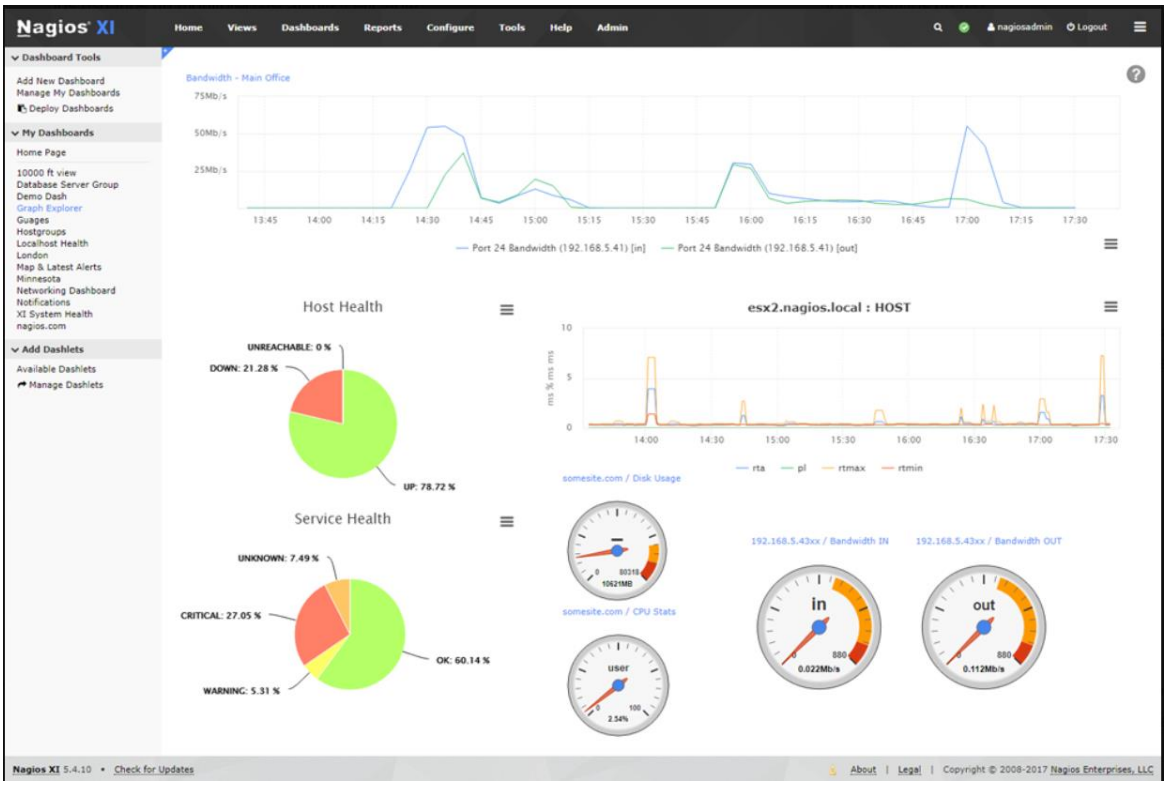

Figura 2 Ejemplo de una vista de Nagios

2.3 NEW RELIC

Es una aplicación basada en la nube que permite la monitorización de los diferentes servicios a webs, aplicaciones y dispositivos móviles. Además, permite la monitorización de redes, el rastreo de bugs y errores. Permite la creación de alertas, así como la representación de la información mediante gráficos.

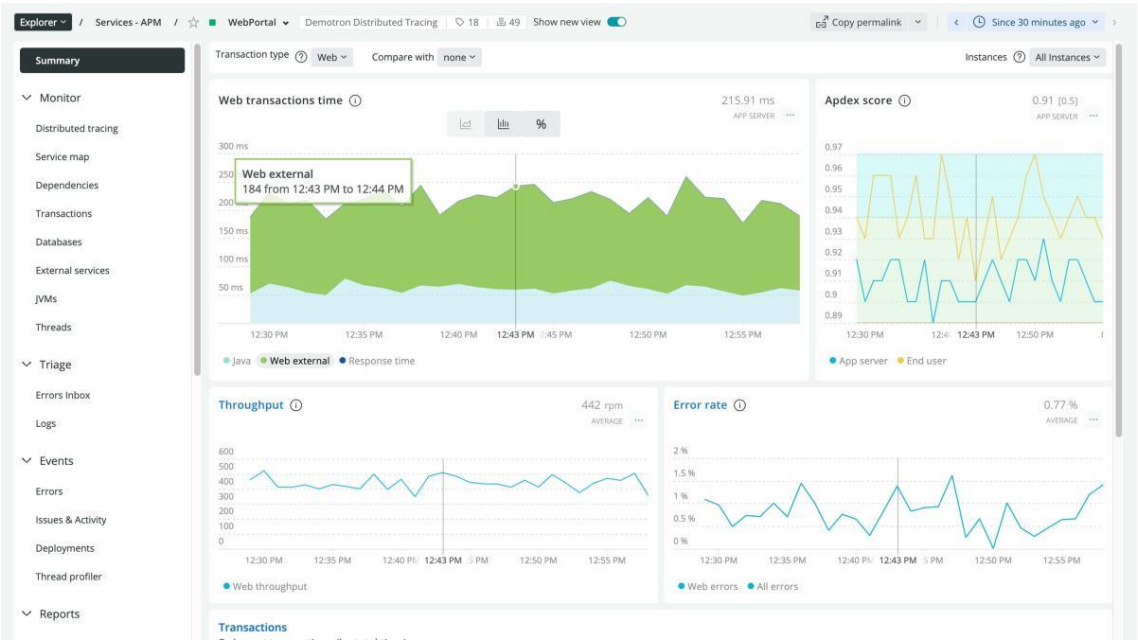

Figura 3 Ejemplo de apariencia y vista de Relic

2.4 ZABBIX

Es una aplicación dedicada a la monitorización de redes, servidores, cloud, aplicaciones servicios y bases de datos. Es necesaria su instalación para su uso. Ofrece soluciones para disintas industrias como la financiera, energética, saludo e ingeniería de las comunicaciones. Además es capaz de mostrar información con potentes gráficas.

Figura 4 Ejemplo de apariencia y vista de Zabbix

3 OBJETIVOS

A continuación, se expondrán los diferentes objetivos que el sistema debe cumplir, así como objetivos personales que se buscan conseguir con el desarrollo del proyecto.

- **OBJETIVOS DEL SISTEMA**

El principal objetivo del sistema es la monitorización y control de los equipos, así como la gestión de empresas y la gestión de sus correspondientes usuarios. Proporcionando una interfaz sencilla e intuitiva que no necesitase ser instalada para su uso.

- **GESTIÓN DE USUARIOS**

El sistema realiza las funciones básicas de gestión de usuarios, como altas, bajas, modificación de estos, restablecimientos de contraseñas, etc. Se controla el acceso al sistema y a las diferentes partes del mismo en base a sus respectivos roles.

- **GESTIÓN DE EMPRESAS**

El sistema debe llevar un control de las empresas que se encuentran registradas dentro de la aplicación, así como de los empleados que están asociadas a las mismas. Además, se pueden llevar a cabo acciones de gestión de empresas, como altas y bajas de las mismas dentro del sistema.

- **GESTIÓN Y CONTROL DE EQUIPOS**

El sistema es capaz de monitorizar la información de los diferentes equipos computacionales de los usuarios registrados, y en caso de que se produzcan anomalías, enviar avisos. Se proporcionan funcionalidades como el alta y la baja de equipos asociados a los empleados de la empresa, así como la visualización de diferentes aspectos relevantes, como el estado del hardware o los servicios arrancados y desactivados dentro de un equipo.

- **GESTIÓN Y CONTROL DE TICKETS**

El sistema permite rellenar y consultar tickets de incidencias, de tal forma que se produce una comunicación directa entre los empleados normales y los administradores de una empresa, dando la posibilidad de resolver problemas que se puedan producir con los equipos computacionales.

- **GESTIÓN DE ALERTAS**

El sistema permite a los administradores crear y trabajar con alertas que controlen el estado de los equipos computacionales. Estas alertas sirven para generar avisos de alertas que se envían a los administradores cuando durante la monitorización, se detecta que se han producido comportamientos anómalos o lecturas no deseadas.

- **COMPATIBILIDAD DEL SISTEMA**

El sistema es compatible con la mayor cantidad de dispositivos posibles, así como navegadores web.

- **DISPONIBILIDAD DEL SISTEMA**

El sistema deberá mantenerse disponible la mayor cantidad de tiempo posible puesto que la gestión de los diferentes equipos y usuarios dependerá de que ello.

- **ESCALABILIDAD**

Se busca que el sistema sea escalable con vistas al uso de la aplicación por varias empresas y en base a la cantidad de trabajadores que cada empresa pudiera tener.

3.1 OBJETIVOS PERSONALES

Uno de los objetivos es la oportunidad de aprender cómo funcionan las aplicaciones web, así como la potencia y las posibilidades que estas tienen, ya que es un tipo de aplicación que está en auge en estos momentos. Además de la posibilidad de aprender a desarrollar aplicaciones con nuevas tecnologías relevantes que están ganando fuerza como Vue.

Además, he escogido el TFG de gestión de centros computacionales porque dentro del ámbito informático, lo que más me ha gustado siempre y más me ha llamado la atención ha sido la administración de sistemas y el control y gestión de redes, y con este trabajo tenía la oportunidad de aplicar y aprender algo relacionado con ello.

Anteriormente ya había creado redes entre los diferentes ordenadores y dispositivos de mi domicilio, y había montado servidores apache y servidor sql con gestión de usuarios incluidos. También había utilizado protocolos web que permiten la obtención y compartición de ciertos tipos de información entre los equipos, sin embargo, nunca lo había hecho a escala algo mayor, con interfaces bien definidas y pudiendo realizar toda la parte de gestión de usuarios con interfaz gráfica y sin tener que escribir

constantemente comandos sobre la terminal para llevar a cabo esta gestión. Y pudiendo además consultar toda esta información mediante un servidor centralizado ajeno a mi red doméstica y que no necesita de complicadas y tediosas configuraciones sobre texto plano.

4 CONCEPTOS TEÓRICOS

En este apartado se especificarán y explicarán diferentes aspectos teóricos que se han tenido en cuenta y que se han llevado a la práctica durante la realización del sistema.

4.1 API REST

Una API es un conjunto de protocolos que nos permiten integrar software de otras aplicaciones en una aplicación, facilitando la programación de ciertas funcionalidades que ya se encuentran programadas para que se hagan de forma sencilla. Un api REST será entonces una interfaz de programación para aplicaciones web que se basa en la arquitectura web y que permite la interacción con los servicios de tipo RESTful (Red Hat, s.f.).

4.2 FIREBASE

Firebase es una plataforma que facilita el desarrollo de aplicaciones web progresivas, la cual ofrece servicios de backend y también múltiples servicios útiles que facilitan el desarrollo de dichas aplicaciones, como servicios de autenticación, de cloud, de almacenamiento y de hosting.

4.3 APLICACIÓN WEB PROGRESIVA (PWA)

Las aplicaciones web progresivas serán como cualquier aplicación que conocemos y que puede ser instalada en cualquier dispositivo, pero tiene la principal característica de que precisamente no necesita ser instalada para poder funcionar, tan solo necesitará de un acceso a la misma mediante navegador web para poderla hacer funcionar, lo cual nos proporciona gran capacidad de compatibilidad con multitud de dispositivos.

4.4 BACKEND

Un backend podríamos definirlo como aquellos puntos donde se ejecutará todo código y funcionalidad que el cliente no debe poder ejecutar por seguridad. Será el encargado de ejecutar las solicitudes enviadas por el frontend (cliente). Se encontrará de cara al servidor encargado de proporcionar el servicio que se está solicitando (Arimetrics, s.f.).

4.5 FRONTEND

El frontend se definirá como aquella parte de código que se encargan de ejecutar los usuarios, somo puede ser un un script de javascript. Son aquellas solicitudes que recibe el backend.

4.6 FRAMEWORK

Podemos definir un framework como una plantilla o una estructura previa que usar para el desarrollo de un proyecto (rockcontent, s.f.). Los framework nos permitirán mantener un código más limpio y agilizar el desarrollo, en este caso se utilizó un framework para aplicaciones web y fue Vue.

4.7 BASE DE DATOS NO RELACIONAL (NOSQL)

Son bases de datos cuya forma de almacenamiento está optimizado para los requisitos específicos de los datos que almacenan, y que no utilizan sentencias sql para sus consultas (Microsoft docs, s.f.).

4.8 MÓDULOS

Existen diferentes tipos de módulos dependiendo de las tecnologías o lenguajes que estemos utilizando. Los módulos son colecciones de funciones escritos en un código concreto que nos proporcionan ciertas acciones que facilitan la implementación de ciertas funcionalidades, por ejemplo, los módulos node en lenguaje JavaScript o los módulos cpan en lenguaje Perl.

4.9 PROCESO UNIFICADO

El proceso unificado es un marco de trabajo genérico que es capaz de especializarse para el desarrollo de diversos sistemas software, sin tener en cuenta el área de aplicación, la organización o los tamaños de los proyectos (García Peñalvo, García Holgado, & Vázquez Ingelmo). Utilizará UML para ello.

4.10 UML

UML es un lenguaje que permite el modelado para visualizar, especificar, construir y documentar las partes de un sistema software desde distinto puntos de vista, y que podrá utilizarse con cualquier proceso de desarrollo a lo largo de todo el ciclo de vida (García Peñalvo, Moreno García , García Izquierdo, & Vázquez Ingelmo). En otras palabras, nos permitirá la completa representación del sistema a lo largo de todo el proceso de desarrollo, así como en sus posteriores etapas, desde lo mayores niveles de abstracción hasta los niveles más bajos de abstracción.

4.11 HERRAMIENTAS CASE

Las herramientas CASE son un conjunto de aplicaciones informáticas utilizadas para el desarrollo de software incrementando la productividad y la calidad en dicho desarrollo.

5 TÉCNICAS Y HERRAMIENTAS

El siguiente apartado de la memoria se dedicará a la especificación y explicación de las diferentes herramientas utilizadas para el desarrollo del sistema, incluyendo bibliotecas, módulos, frameworks, entre otras.

5.1 HERRAMIENTAS CASE

5.1.1 VISUAL PARADIGM

Es una herramienta case que nos ayuda durante el desarrollo de programas informáticos partiendo de la planificación, hasta el análisis, diseño, código fuente y la posterior documentación. Nos permitirá soportar el ciclo completo del proceso de desarrollo software proporcionándonos la posibilidad de crear todo tipo de diagramas (CU, diagramas de secuencia, diagramas de paquetes, diagrama de despliegue…) (EcuRed, s.f.).

Se ha escogido esta herramienta puesto que es muy completa y durante el grado se ha utilizado a menudo, por lo tanto, se tienen previos conocimientos sobre la misma.

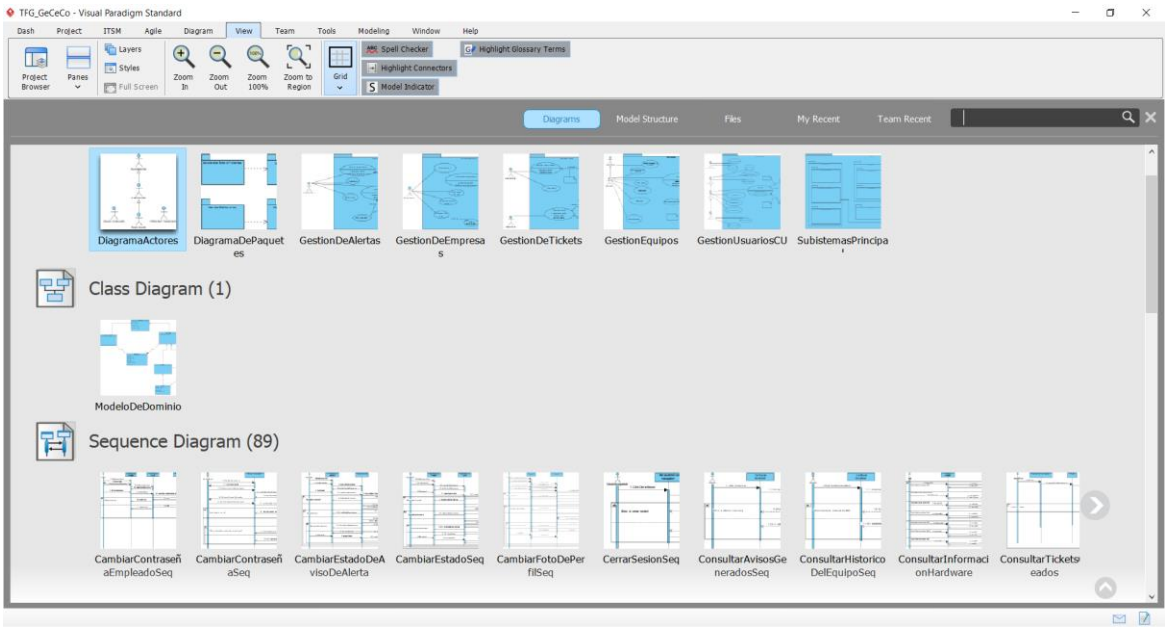

Figura 5 Herramienta Visual Paradigm

5.1.2 REM

Es una herramienta que nos permite realizar la gestión de requisitos software durante la fase de ingeniería de los requisitos (Universidad de Sevilla, s.f.). Nos facilita la creación y especificación de los diferentes elementos de esta fase, como puede ser la creación de tablas de caos de uso, especificación de los objetivos, definición de los atores, creación de la matriz de rastreabilidad, entre otras.

Se ha usado REM porque facilita en gran medida la creación de tablas de CU así como su posterior exportación.

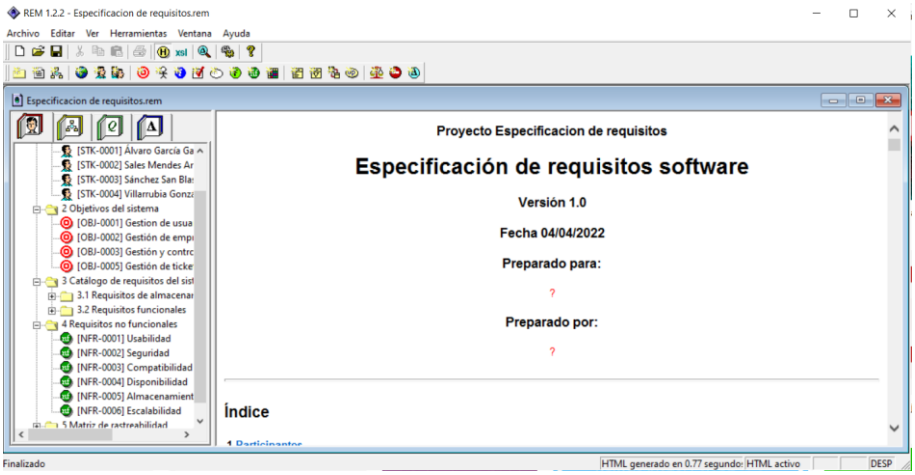

Figura 6 Herramienta REM

5.1.3 MICROSOFT PROYECT

Es un software diseñado por Microsoft, que nos permite la posibilidad de gestionar presupuestos, evaluar ritmos, cargas laborales, asignar recursos, entre otros. Además, hace uso de diagramas de Grantt, lo cual nos permite conocer cuál es el mejor camino a seguir a la hora de desarrollar un proyecto, maximizando el rendimiento y el uso de los recursos (esan business, s.f.).

Se ha elegido Microsoft proyect puesto que ya había sido utilizado previamente y es una herramienta muy completa.

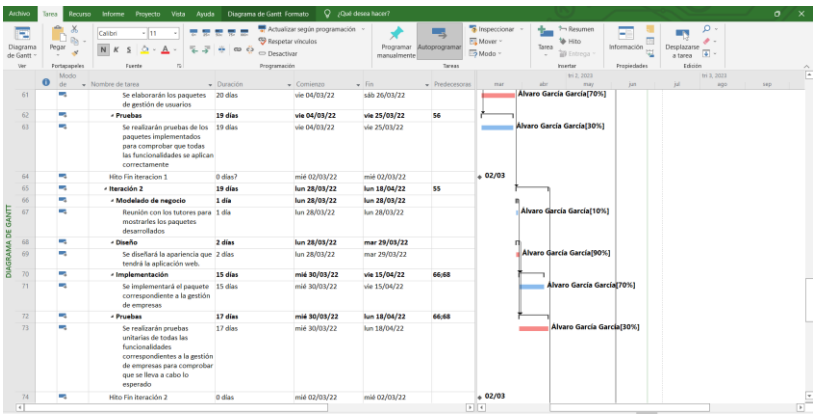

Figura 7 Herramienta Microsoft Project con diagrama de Grantt

5.1.4 EZESTIMATE

Es una herramienta que permite estimar el esfuerzo de un proyecto a par ir de los puntos de casos de uso (García, s.f.).

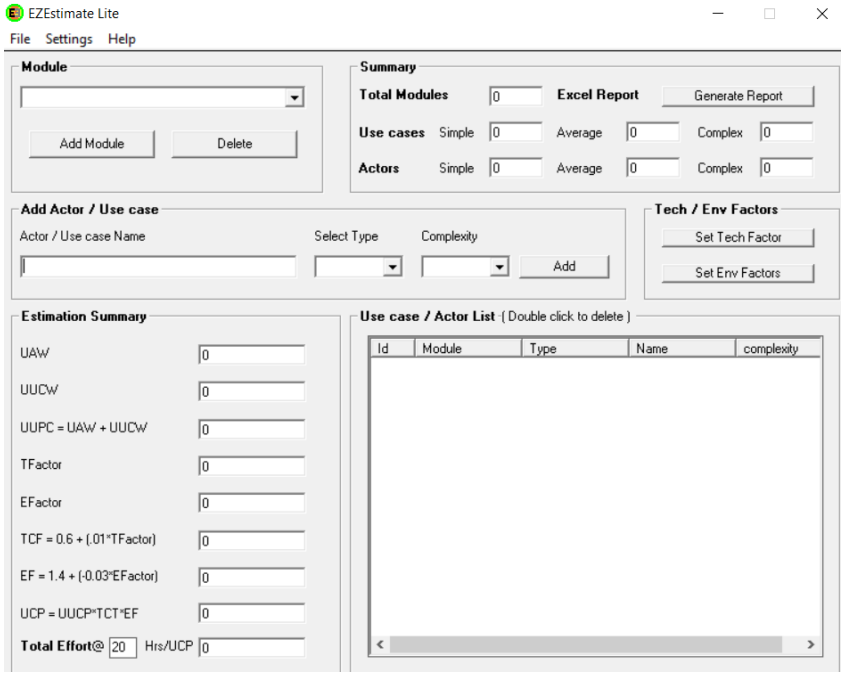

Figura 8 Herramienta EzEstimate

5.2 FRAMEWORKS

5.2.1 VUE

Es un framework progresivo utilizado para desarrollar interfaces de usuario, está diseñado para ser utilizado incrementalmente y está principalmente enfocado a la capa de visualización. Es fácil de integrar con otras librerías y proyectos existentes y es comúnmente utilizado para el desarrollo de aplicaciones de una sola página (VUEJS, s.f.).

Se ha escogido este framework porque es una tecnología que está en auge para el desarrollo de aplicaciones web, además tiene muy buena documentación y existen multitud de foros con respuestas a problemas y errores. Es fácil de utilizar y compatible con multitud de frameworks complementarios.

5.2.2 VUETIFY

Es un framwork de interfaz de usuarios construido sobre Vue, el cual proporciona herramientas para la creación de componentes de forma sencilla con buena apariencia, como pueden ser tablas, diálogos, botones, interacciones, etc. Además, vuetify implementa lo que se conoce como enfoque móvil y es que sus componentes pueden visualizarse correctamente desde cualquier tipo de equipo mediante el que accedamos a la página.

Se ha escogido este framework porque es fácil integrarlo con Vue y posee una documentación muy completa (también en Español), además de múltiples ejemplos de componentes que permiten comprender fácilmente el funcionamiento de los mismos.

5.3 ENTORNOS DE DESARROLLO

5.3.1 VISUAL CODE

Es un entorno de desarrollo integrado desarrollado por Microsoft que nos permite la estructuración de los diferentes componentes de un sistema software, así como la escritura de código. Es capaz de reconocer múltiples lenguajes de programación además de que es posible instalarle nuevos componentes y herramientas que faciliten el proceso de desarrollo. Cuenta además con diversas herramientas como terminal (incluyendo la posibilidad de utilizar la poweshell de Windows) y compilador de código.

Se utiliza este entorno puesto que era el más recomendado a la hora de desarrollar una aplicación web en múltiples foros consultados.

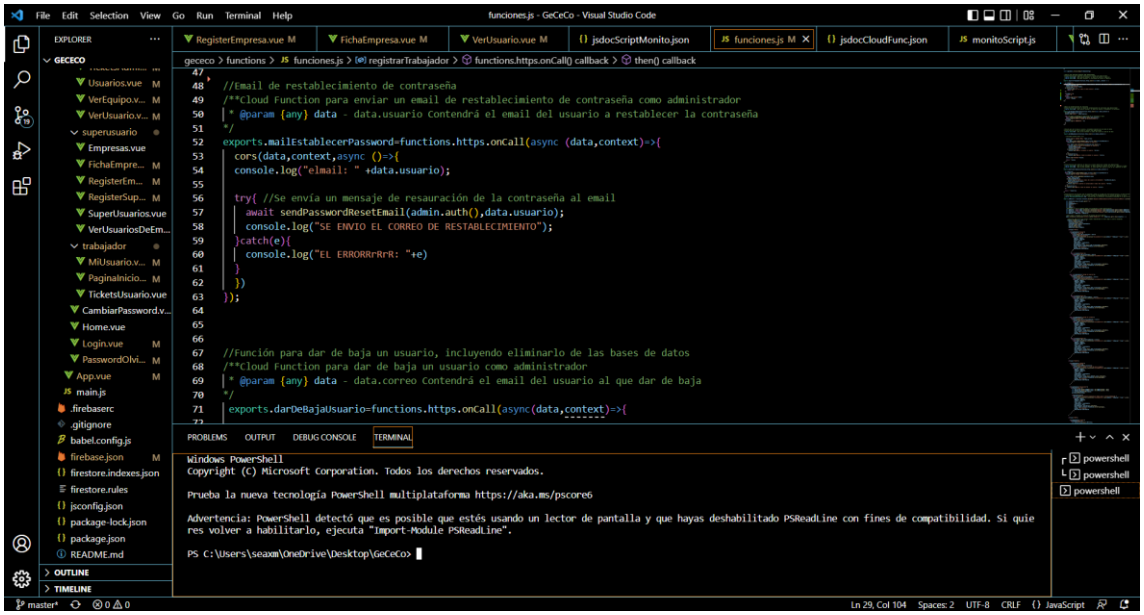

Figura 9 Entorno Visual Studio Code

5.3.2 SUBLIME TEXT

Es un editor de código compatible con cualquier sistema operativo, es capaz de reconocer múltiples lenguajes de programación, además, es muy liviano.

Se utiliza este editor de código puesto que, aunque la mayor parte del proyecto está desarrollado en Windows, las pruebas y la codificación en Linux se hacen utilizando este editor.

5.4 MÓDULOS

5.4.1 FIREBASE

Módulo que nos permite hacer uso de los diferentes servicios proporcionados por Firebase, como la autenticación o el uso de la base de datos Firestore

5.4.2 SYSTEMINFORMATION

Módulo que nos permite la obtención de la información de los diferentes equipos, utilizado para realizar la monitorización de estos.

5.4.3 FIREBASE-ADMIN

Módulo que nos proporciona las funcionalidades propias de un administrador, como registrar usuarios o darlos de baja. Estas funcionalidades se ejecutarán siempre desde el lado del backend.

5.4.4 JS-DOC

Utilizado para la generación de la documentación del código fuente escrito en Java Script, utilizando unas reglas concretas para comentar los métodos, nos permite la generación de una página web donde podemos consultar cada uno de los diferentes métodos, así como los parámetros de entrada que reciben.

En el proyecto se usa tanto para la documentación de las Cloud Functions de Firebase como para la documentación de las funciones del script de monitorización.

5.4.5 VUE-STYLEGUIDIST

Utilizado para la documentación de los ficheros de código ".vue". Nos permite al igual que JSDoc, en base a unas reglas específicas de comentado, documentar los diferentes métodos y la creación de una página web (o fichero .html) donde poder consultar dicha información.

En el proyecto se uda para la documentación de todos los ficheros ".vue".

5.4.6 NODEMAILER

Módulo que nos permite la creación y envío de correos electrónicos utilizando una cuenta de correo electrónico como remitente.

5.4.7 NODECRON

Nodecron es un módulo que nos perite programar tareas en Java Script de forma similar a como se hacen con los crontab en Linux, utilizan la misma sintaxis.

5.5 LENGUAJESDE PROGRAMACIÓN

5.5.1 JAVASCRIPT

Es un lenguaje de programación ligero interpretado y orientado a objetos. Aunque puede utilizarse para desarrollar diversos programas, es comúnmente utilizado para proporcionar funcionalidades en páginas web que serán ejecutadas del lado del cliente.

Tanto las funcionalidades ejecutadas por el cliente, como las Cloud Functions de Google, como el script de monitorización, se implementan utilizando este lenguaje.

```
}

/** Función encargada de comporbar si existe un equipo con el correo asociado para dicho tabajador
 *  @param {string} id - Correo del usuario que quiere iniciar la monitorización
 */
async function hacerConsultaIdEmpleado(id){

    var existe="0";
    const q = query(collection(firestoreDB, "Equipos"), where("CorreoTrabajador", "==", id));

    const querySnapshot = await getDocs(q);
    try{
        querySnapshot.forEach((doc) => {
            idEmpresaUsuario=idEmpresaUsuario+""+doc.get("IdUsuario");
            IdUs=id;
            existe="1";
        });
        if(existe=="1"){
            comprobarDatosIntroducidos(idEmpresaUsuario);
        }else{
            console.log("\n(ERROR): Los datos introducidos no son correctos");
        console.log("\nVuelve a iniciar el programa")
        cerrarPrograma();
        }
    }
    catch{
        console.log("\n(ERROR): Los datos introducidos no son correctos");
        console.log("\nVuelve a iniciar el programa")
        cerrarPrograma();
    }
}
```

Figura 10 Lenguaje Java Script

5.5.2 HTML

Es un lenguaje de marcado, es el código utilizado para la estructuración del contendido dentro de una página web (mdn web docs, s.f.). Se utilizarán diferentes etiquetas para la representación de los diferentes elementos.

```
<template>
  <v-app>
    <div id="app">
      <v-layout>
        <router-link to="/">
          <v-img  max-height="50" max-width="50" src='./images/Gececo.jpg'></v-img>
        </router-link>
      </v-layout>
      <router-view/>
    </div>
  </v-app>
</template>
```

Figura 11 Lenguaje HTML

5.5.3 JSON

Es un formato ligero para el intercambio de datos, es sencillo de leer y escribir para los humanos y también de interpretarlo y generarlo para las máquinas (json.org, s.f.). En el presente proyecto se usa para la configuración de ciertos parámetros y además para la configuración de las bases de datos de Firestore.

Figura 12 Lenguaje Json

5.6 BACKUPS Y CONTROL DE VERSIONES

5.6.1 GITHUB

Es una página que permite el alojamiento de código de las aplicaciones para los diferentes desarrolladores, está basado en Git y principalmente se utiliza para realizar backups y control de versiones. En el presente proyecto se utiliza como medio para salva guardar los avances de la aplicación, así como para tener la posibilidad de volver a una versión anterior en caso de que sea necesario.

5.7 COMPILACIÓN Y EJECUCIÓN

5.7.1 NODEJS

Es un entorno en tiempo de ejecución multiplataforma para la capa del servidor que está basado en JavaScript, además se encuentra controlado por eventos diseñado para el desarrollo de aplicaciones escalables (ITDO, s.f.).

5.7.2 NODE PACK MANAGER (NPM)

Es el gestor de paquetes de NodeJS, el cual nos permite la instalación de los diferentes módulos de forma sencilla, así como la ejecución de diferentes órdenes, como puede ser el arranque del servidor local, la construcción del proyecto, o la generación de documentación para el código entre otras. Además, a través de un fichero de configuración, mediante npm, podemos instalar los todos los módulos, así como las versiones necesarias para cada uno con una simple orden.

5.7.3 WEBPACK

Es una herramienta de compilación y empaquetación de todos los ficheros de un proyecto con todas sus dependencias para frontend. Es decir, permite la generación de un único archivo con todos los módulos necesarios para que una aplicación funcione.

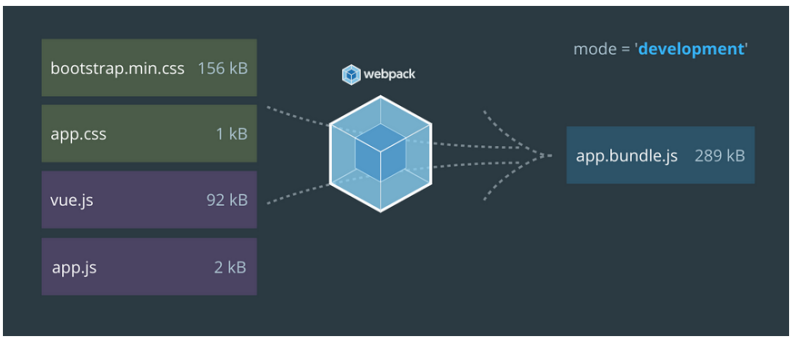

Figura 13 Representación WebPack

6 ASPECTOS RELEVANTES DEL DESARROLLO

En el siguiente apartado se tratarán los aspectos más relevantes producidos durante las distintas fases del desarrollo de la aplicación.

6.1 MARCO DE TRABAJO

Como marco de trabajo se hace uso del proceso unificado, el cual está guiado por casos de uso, centrado en la arquitectura y es iterativo e incremental, concordando con la característica evolutiva del software moderno. Se centra en la importancia de la arquitectura del software fijándose en las metas, el rendimiento, teniendo en cuenta futuras modificaciones y reutilización de software (Universidad César Vallejo).

- **Casos de uso:** Son una técnica que permiten la especificación de los requisitos funcionales y que se encuentra incorporada a UML. Nos permite modelar las funcionalidades del sistema de la forma que lo perciben los actores, los cuales serán aquellos agentes que interactúan con el sistema (García Peñalvo, García Holgado, & Vázquez Ingelmo, Fundamentos de la vista de casos de uso).
- **Centrado en la arquitectura:** "La arquitectura se representará mediante las vistas del modelo. Se podrá anotar como arquitectura de referencia el modelo de arquitectura de 4+1 vistas, que está propuesto por Philippe Kruchten (1995), en el que cada vista será una parte del modelo". Los modelos serán la forma de visualizar, especificar, construir y documentar la arquitectura (García Peñalvo, García Holgado, & Vázquez Ingelmo).
- **Iterativo e incremental:** El proceso se dividirá en iteraciones, y en cada una de estas se identificarán los casos de uso relevantes, creando un diseño basado en la arquitectura definida. El diseño será implementado mediante componentes y se verificará que los componentes satisfacen los casos de uso. Cuando una iteración cumpla con los objetivos especificados se saltará a la siguiente iteración. Cada una de las iteraciones irá desarrollando el sistema de forma incremental (García Peñalvo, García Holgado, & Vázquez Ingelmo).

El proceso unificado estará dividido en un total de cuatro fases, las cuales serán inicio, elaboración construcción y transición, las cuales a su vez estarán divididas en una serie de iteraciones donde se llevarán a cabo unas disciplinas.

Las cuatro fases en las que se divide el proceso unificado son las siguientes:

1. **Inicio:** Se definirá el alcance del proyecto y se elaborarán los casos de uso
2. **Elaboración:** Se corresponde con la planificación del proyecto software y se definen los casos de uso, así como el diseño de la arquitectura software.
3. **Construcción:** Será la propia fase de implementación del sistema software.
4. **Transición:** Es la fase en la que contamos con una primera versión funcional y la dedicamos al corregimiento de errores, bugs y optimización del sistema. (García Peñalvo, García Holgado, & Vázquez Ingelmo)**.**

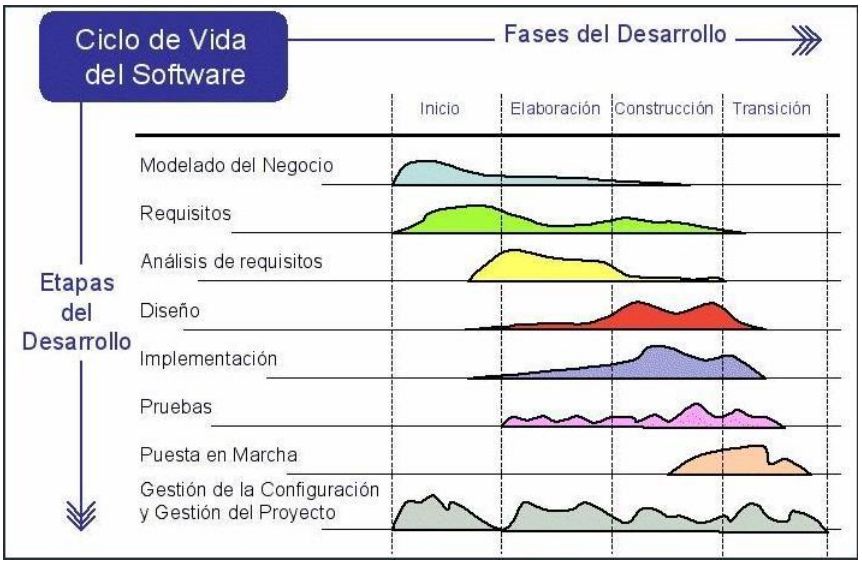

Figura 14 Representación del Proceso Unificado

6.2 ESTIMACIÓN DE COSTE Y PLANIFIACIÓN TEMPORAL

6.2.1 ESTIMACIÓN DE COSTE

La estimación de costes consiste en realizar predicciones sobre las cantidades más probables de esfuerzo que se requerirán para la construcción del sistema software, entendiendo como esfuerzo la relación entre tiempo y personal precisado para desarrollar el proyecto. La unidad utilizada para medir este esfuerzo son los meses de persona.

Se utilizarán para esto modelos que son técnicas que definen las claves que contribuyen al esfuerzo, generando una fórmula matemática que relaciona estas claves con el esfuerzo necesario (García, Estimación del esfuerzo).

Para este proyecto se usa la técnica de complejidad de los casos de uso, para determinar el nivel de complejidad que tienen cada uno de los casos de uso deberemos analizar el número de transacciones que se necesitarán para llevar a cabo las funcionalidades que describen cada uno de los casos de uso. Se tienen en cuenta las siguientes especificaciones:

- **Simple:** Tres o menos transacciones.
- **Medio:** Entre cuatro y siete transacciones (ambos incluidos).
- **Complejo:** Más de siete transacciones.

También es necesario determinar la complejidad de los actores del sistema:

- **Simple:** El actor es un sistema y la aplicación se comunica con el mediante una API.
- **Medio:** El actor es un sistema y la aplicación se comunica con el mediante un protocolo web.
- **Complejo:** El actor es un usuario que interacciona con una interfaz gráfica

Posteriormente en base a unas fórmulas predefinidas, se calcula el impacto de los factores de complejidad técnica, así como los factores de complejidad del entorno.

- **Factores de complejidad técnica:** Representará el esfuerzo que supondrá para el desarrollador o desarrolladores la implementación de dicho factor en el desarrollo
- **Factores de complejidad de entorno:** Representarán la experiencia que tiene el desarrollador o desarrolladores en los diferentes factores en cuanto a lo que se tiene que desarrollar.

Factor	Descripción	Peso
T1	Sistema distribuido.	2
T2	Objetivos de performance o tiempo de respuesta.	1
T3	Eficiencia del usuario final.	1
T4	Procesamiento interno complejo.	1
T5	El código debe ser reutilizable.	1
T6	Facilidad de instalación.	0.5
T7	Facilidad de uso.	0.5
T8	Portabilidad.	2
T9	Facilidad de cambio.	1
T10	Concurrencia.	1
T11	Incluye objetivos especiales de seguridad.	1
T12	Provee acceso directo a terceras partes.	1
T13	Se requiere facilidades especiales de entrenamiento a usuario.	1

$$TFactor = \Sigma (Valor*Peso)$$

$$FCT = 0.6 + (0.01 * TFactor)$$

Figura 15 Formula y descripción de los factores de complejidad técnica

Factor	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado.	1.5
E2	Experiencia en la aplicación.	0.5
E3	Experiencia en orientación a objetos.	1
E4	Capacidad del analista líder.	0.5
E5	Motivación.	1
E6	Estabilidad de los requerimientos	2
E7	Personal part-time	-1
E8	Dificultad del lenguaje de programación	-1

$$FE = \Sigma (Valor * Peso)$$

$$FE = 1.4 + (-0.03 * EFactor)$$

Figura 16 Fórmula y descripción de los factores de complejidad del entorno

A continuación, tras definir y determinar los distintos valores tanto para la complejidad de los casos de uso, como para los factores de complejidad técnica y de entorno, se ha hecho uso de la herramienta EZEstimate, la cual nos permite agilizar el cálculo de las horas de persona. Tan solo tendremos que proporcionarle dichos valores y aplicará las fórmulas necesarias para la obtención de las mismas.

Esto se puede encontrar más detallado en el Anexo I del presente trabajo.

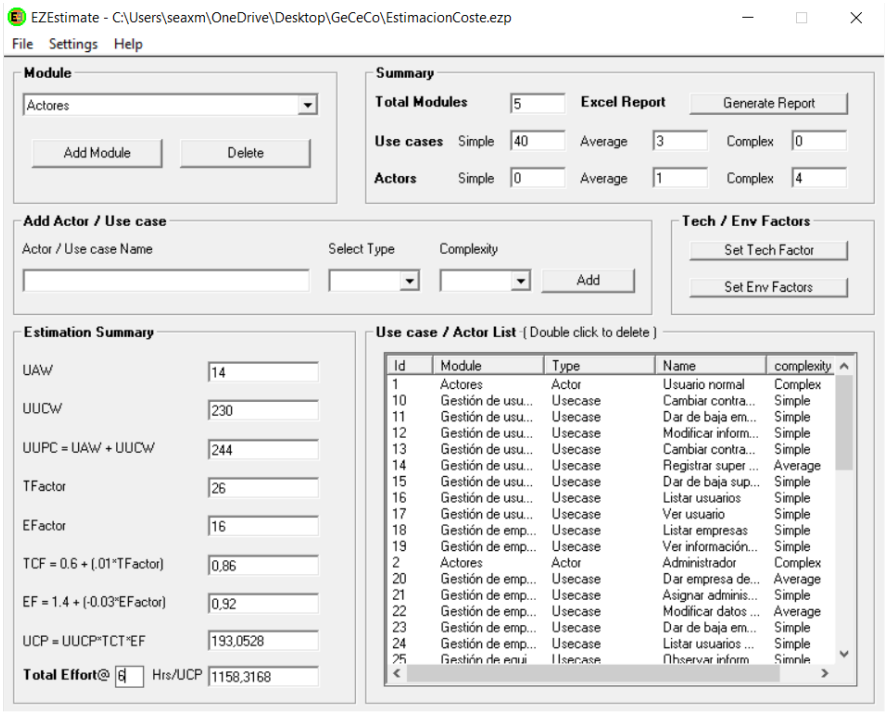

Figura 17 Coste de los casos de uso en EZEstimate

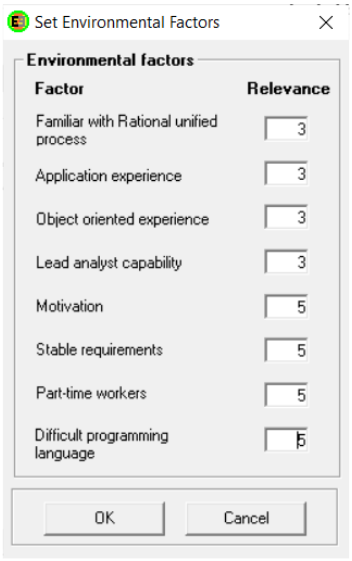

Figura 18 Valores asignados en factores de complejidad del entorno en EZEstimate

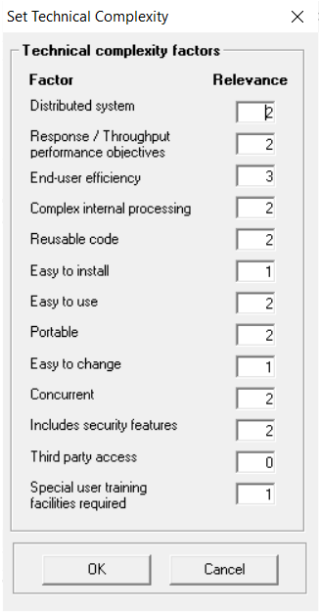

Figura 19 Valores asignados a los factores de complejidad técnica en EZEstimate

6.2.2 PLANIFICACIÓN TEMPORAL

Durante el desarrollo de un software, la planificación temporal nos permite identificar, definir y programar las diferentes tareas que debemos llevar a cabo para la obtención de un software de calidad en un tiempo controlado.

Como se explicó anteriormente para el desarrollo del software se usa proceso unificado, el cual se dividía en cuatro fases, estas fases a su vez se dividen en seis disciplinas que dependiendo de la etapa en la que nos encontremos en el desarrollo del software tendrán mayor o menor peso. Estas disciplinas son las siguientes, y se han utilizado de la siguiente forma:

- **Modelado de negocio:** Se centra en reuniones con los tutores y se extiende a lo largo de todo el proyecto.
- **Requisitos:** El objetivo definir e implementar todos los requisitos, para este proyecto los requisitos deben quedar completamente definidos y estipulados en las dos primeras fases (inicio y elaboración). Tras esto no deben ser cambiados ni modificados.
- **Análisis:** Se analizan diferentes aspectos referentes al sistema.
- **Diseño:** Se idea la manera de construir e interconectar los diferentes componentes del sistema, así como la creación de su estructura y el funcionamiento total del sistema. El diseño se alarga hasta la fase de construcción donde se termina por diseñar la apariencia que tiene la aplicación web final.
- **Implementación:** Se corresponde principalmente con la construcción y desarrollo del software durante la fase de construcción, aunque comienza un poco antes con algunos testeos de código.

- **Pruebas:** Se centra en la búsqueda de errores y bugs para su posterior corrección, se desarrolla principalmente durante la etapa de construcción para comprobar que todo se realiza correctamente, y también se desarrolla ampliamente durante la fase de transición, donde se trata de encontrar y arreglar aquellos bugs que se hayan podido pasar por alto durante el resto del proceso.

Para realizar la planificación temporal en este proyecto se ha usado el software Microsoft Proyect, en el cual se pueden determinar las diferentes actividades a llevar a cabo durante el desarrollo del sistema, así como la duración estimada para cada una de ellas. Además, se indican los recursos necesarios para llevarlas a cabo. Una vez establecidas las etapas, las disciplinas y las tareas correspondientes, el programa alabora un diagrama de Grantt que nos permite comprobar el camino crítico. También, nos proporciona una estimación sobre el tiempo en días que dudará el desarrollo del proyecto.

Esto puede encontrarse más detallado también en el Anexo I del presente trabajo.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1		**GeCeCo_TFG**	**128 días?**	**lun 07/02/22**	**mar 05/07/22**		
2		**Inicio**	**14 días**	**lun 07/02/22**	**mar 22/02/22**		
3		**Iteración 1**	**4 días**	**lun 07/02/22**	**jue 10/02/22**		
4		**Modelado de negocio**	**2 días**	**lun 07/02/22**	**mar 08/02/22**		
5		Reunión junto con los tutores para analizar el tema de TFG	2 días	lun 07/02/22	mar 08/02/22		Álvaro García García
6		**Requisistos**	**1 día**	**mié 09/02/22**	**mié 09/02/22**		
7		Establecer mínimas funcionalidades	1 día	mié 09/02/22	mié 09/02/22	5	Álvaro García García
8		**Diseño**	**1 día**	**jue 10/02/22**	**jue 10/02/22**		
9		Primera propuesta arquitectónica	1 día	jue 10/02/22	jue 10/02/22	7	Álvaro García García
10		Hito Fin Iteración 1	0 días	lun 07/02/22	lun 07/02/22		
11		**Iteración 2**	**10 días**	**vie 11/02/22**	**mar 22/02/22**	**3**	
12		**Modelado de negocio**	**1 día**	**vie 11/02/22**	**vie 11/02/22**		
13		Se definen los principales objetivos a llevar a cabo	1 día	vie 11/02/22	vie 11/02/22		Álvaro García García
14		**Requisitos**	**3 días**	**sáb 12/02/22**	**mar 15/02/22**	**12**	
15		Establcer requisitos funcionales, no funcionales y de información	3 días	sáb 12/02/22	mar 15/02/22		Álvaro García García[60%]
16		Definición de los actores del sistema	2 días	sáb 12/02/22	lun 14/02/22		Álvaro García García[40%]
17		**Análisis**	**6 días**	**mié 16/02/22**	**mar 22/02/22**	**12;14**	
18		Selección de lenguajes a usar	6 días	mié 16/02/22	mar 22/02/22		Álvaro García García[20
19		Selección de tecnologías a usar	6 días	mié 16/02/22	mar 22/02/22		Álvaro García García[20%]
20		Selección de software a usar	6 días	mié 16/02/22	mar 22/02/22		Álvaro García García[20
21		**Diseño**	**5 días**	**mié 16/02/22**	**lun 21/02/22**	**12;14**	

Figura 20 Tareas, tiempos y recursos en Microsoft Project

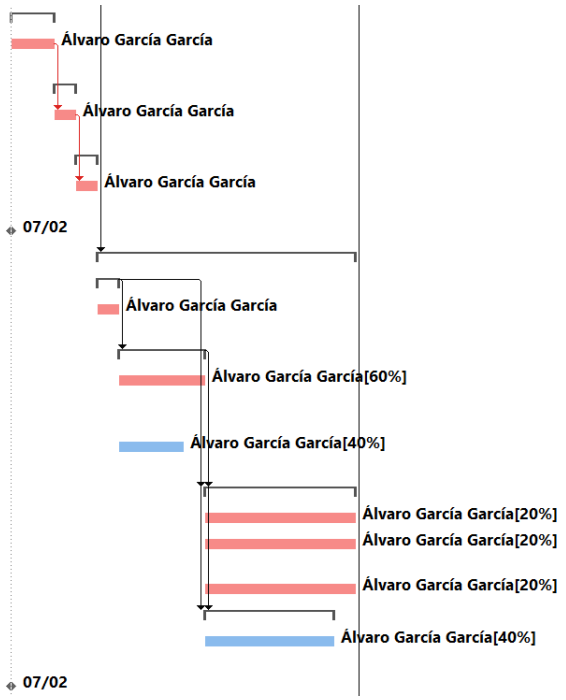

Figura 21 Diagrama de Grantt con camino crítico generado en Microsoft Project

6.3 ESPECIFICACIÓN DE REQUISITOS

En esta etapa del trabajo se lleva a cabo la especificación de los requisitos del software, en el cual se definen cuáles serán los participantes del proyecto, los objetivos a cumplir, cuáles son los requisitos de información, los actores con los que interactuará el sistema, especificación de los requisitos funcionales mediante la elaboración y descripción de los diagramas de casos de uso y la especificación de los requisitos no funcionales.

A continuación, se mostrarán algunos ejemplos de cada una de estas partes nombradas para dicho proyecto, sin embargo, la especificación de requisitos completa puede encontrarse en el Anexo II.

6.3.1 PARTICIPANTES DEL PROYECTO

- Álvaro García García
- André Felipe Sales Mendes
- Héctor Sánchez San Blas
- Gabriel Villarrubia González

Participante	Álvaro García García
Organización	Universidad de Salamanaca
Rol	Desarrollador/Programador
Es desarrollador	Sí
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 1 Ejemplo participante del proyecto

6.3.2 OBJETIVOS A CUMPLIR

- **Gestión de usuarios:** El sistema puede realizar las funciones básicas de gestión de usuarios, como altas, bajas, modificaciones, restablecimientos de contraseña, etc.
- **Gestión de empresas:** El sistema lleva un control de las empresas que se encuentran registradas dentro de la aplicación.
- **Gestión y control de equipos:** El sistema es capaz de monitorizar la información de los diferentes equipos computacionales de los usuarios registrados. En caso de que se produzca alguna anomalía el sistema es capaz de enviar alertas.
- **Gestión de tickets:** El sistema permite rellenar tickets de incidencias que son almacenados.
- **Gestión de alertas:** El sistema permite crear y trabajar con alertas que controlan el estado de los equipos computacionales.

OBJ-0003	Gestión y control de equipos
Versión	1.0 (04/04/2022)
Autores	• Álvaro García García
Fuentes	• Sales Mendes André Felipe • Sánchez San Blas Héctor • Villarrubia González Gabriel
Descripción	El sistema deberá *poder monitorizar la información de los diferentes equipos computacionales de los usuarios registrados. En caso de que se produzca alguna anomalía el sistema deberá enviar alertas.*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 2 Ejemplo representación objetivo del proyecto

6.3.3 REQUISITOS DE INFORMACIÓN

Los requisitos de información son los datos con los que el sistema trabaja y almacena.

- **Información de usuarios**: El sistema almacena la información correspondiente a los usuarios.
- **Información de empresas**: El sistema almacena la información correspondiente a las empresas.
- **Información de equipos:** El sistema almacena la información correspondiente a los equipos computacionales.
- **Información de tickets:** El sistema almacena la información correspondiente a los tickets que se pueden crear y consultar, serán breves incidencias que deberán ser atendidas.
- **Información de alertas:** El sistema almacena la información correspondiente a las alertas creadas por el sistema para avisar de anomalías.

IRQ-0002	Información de empresas	
Versión	1.0 (04/04/2022)	
Autores	• Álvaro García García	
Fuentes	• Sales Mendes André Felipe • Sánchez San Blas Héctor • Villarrubia González Gabriel	
Dependencias	• [OBJ-0002] Gestión de empresas	
Descripción	El sistema deberá almacenar la información correspondiente a *las empresas.* En concreto:	
Datos específicos	• Nombre • Correo • Dirección • CP • Teléfono • IdEmpresa • NumEmpleados	
Tiempo de vida	**Medio**	**Máximo**
	-	-
Ocurrencias simultáneas	**Medio**	**Máximo**
	-	-
Importancia	vital	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 3 Ejemplo representación requisito de información

6.3.4 REQUISITOS FUNCIONALES

Para la especificación de los requisitos funcionales, primeramente, se ha elaborado un diagrama de paquetes que define las dependencias que existen entre los mismos. Posteriormente se han definido los actores con los que interactuará el sistema y se ha elaborado un diagrama que define la herencia existente entre los mismos. Por último, se especificaron los casos de uso, se elaboraron diagramas de casos de uso y se elaboraron las tablas de casos de uso que permiten su descripción detallada.

6.3.4.1 DIAGRAMA DE PAQUETES

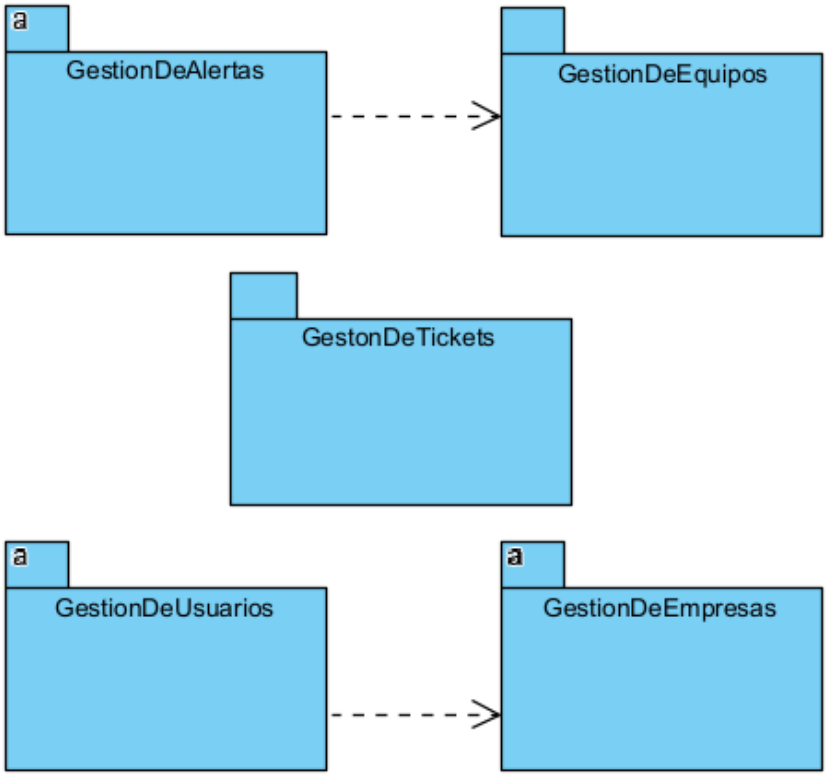

Figura 22 Diagrama de paquetes

6.3.4.2 DEFINICIÓN DE LOS ACTORES

Se definieron un total de cinco actores:

- Usuario Normal
- Administrador
- Super Usuario
- <<Sistema>>Temorizador
- Usuario no logueado

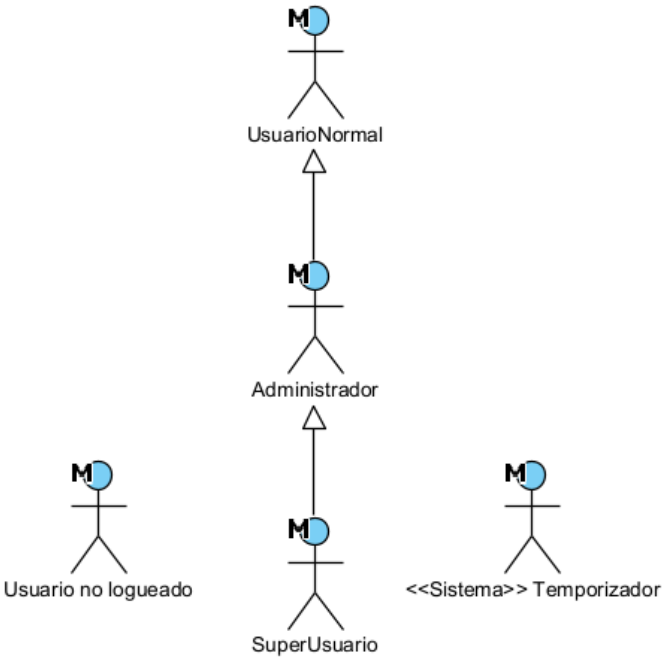

Figura 23 Diagrama herencia de los actores

ACT-0003	**Administrador**
Versión	1.0 (04/04/2022)
Autores	• Álvaro García García
Fuentes	• Sales Mendes André Felipe • Sánchez San Blas Héctor • Villarrubia González Gabriel
Descripción	Este actor representa *los administradores de las empresas, encargados de desarrollar la principal funcionalidad de gestión de la aplicación. Se le proporcionarán funcionalidades avanzadas.*
Comentarios	Ninguno

Tabla 4 Ejemplo representación de un actor del sistema

6.3.4.3 DEFINICIÓN DE LOS CASOS DE USO

1. Definir casos de uso
2. Elaboración de diagramas de casos de uso
3. Rellenar tablas de casos de uso

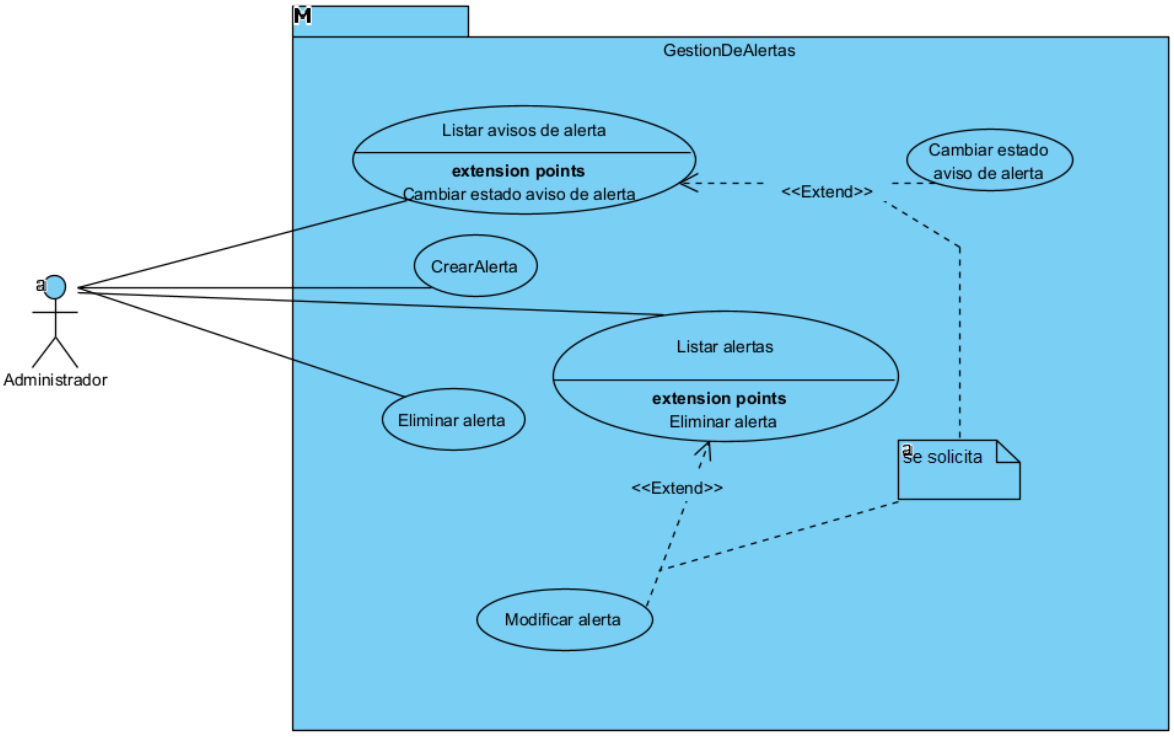

Figura 24 Diagrama de casos de uso Gestión de alertas

UC-0038	**Crear alerta**
Versión	1.0 (05/05/2022)
Autores	• Álvaro García García
Fuentes	• Sales Mendes André Felipe • Sánchez San Blas Héctor • Villarrubia González Gabriel
Dependencias	• [OBJ-0005] Gestión de alertas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se solicita crear un nuevo tipo de alerta*
Precondición	Haber listado las alertas

Secuencia normal	**Paso**	**Acción**
	1	El actor Administrador (ACT-0003) *solicita al sistema la creación de un nuevo tipo de alerta*
	2	El sistema *solicita al usuario la información correspondiente a la nueva alerta*
	3	El actor Administrador (ACT-0003) *proporciona la información solicitada por el sistema*
	4	El sistema *crea y almacena la nueva alerta creada*
Postcondición	Nueva alerta creada	
Excepciones	**Paso**	**Acción**
	4	Si *falta información para crear la alerta,* el sistema *avisa al usuario* , a continuación este caso de uso *continúa*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada	1 veces por semana(s)	
Importancia	vital	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 5 Ejemplo representación tabla de caso de uso

6.3.5 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales afectan principalmente a la calidad del software final desarrollado. Para el presente proyecto se han definido los siguientes requisitos no funcionales:

- **Usabilidad:** Facilidad de uso.
- **Seguridad:** Control de seguridad.
- **Compatibilidad:** Capacidad para funcionar en cualquier dispositivo.
- **Disponibilidad:** Porcentaje de tiempo que el servicio está disponible.
- **Almacenamiento:** Capacidad para almacenar información.
- **Escalabilidad:** Capacidad para crecer y soportar más clientes, usuarios usando la aplicación a la vez, peticiones, etc.

NFR-0006	Escalabilidad
Versión	1.0 (04/04/2022)
Autores	• Álvaro García García
Fuentes	• Sales Mendes André Felipe • Sánchez San Blas Héctor • Villarrubia González Gabriel
Dependencias	• [OBJ-0003] Gestión y control de equipos • [OBJ-0001] Gestion de usuarios • [OBJ-0002] Gestión de empresas
Descripción	El sistema deberá *ser escalable ya que necesitará poder soportar el número de usuarios que puedan acabar utilizando la aplicación, con vistas al uso de la aplicación por varias empresas, y dado que la cantidad de trabajadores que pueda tener cada una es diversa*
Importancia	vital
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 6 Ejemplo representación tabla requisitos no funcionales

6.4 ANÁLISIS DE REQUISISTOS

Una vez fueron obtenidos los requisitos al terminar la fase de especificación de requisitos se comenzó con la fase de análisis de los requisitos, la cual consiste en desgranar cada una de las partes que tendrá el sistema software, con el objetivo de asegurar la calidad del producto final, así como detectar y resolver conflictos durante la especificación de requisitos (García Peñalvo & García Holgado, Ingeniería de requisistos).

Todo el proceso de análisis de requisitos puede consultarse completo en el Anexo III del presente trabajo.

Los pasos seguidos fueron los siguientes:

1. **Modelo de dominio:** Se ha elaborado un diagrama de clases que representa los requisitos de datos dentro de la aplicación.
2. **Realización de casos de uso:** Mediante diagramas de secuencia se representa el intercambio de mensajes producido por cada uno de los casos de uso especificados en el Anexo II de este trabajo.
3. **Clase de análisis:** Con las clases representadas durante los diagramas de secuencia se crean las relaciones entre las mismas para definir sus uniones.
4. **Vista de arquitectura del modelo de análisis:** Se realiza una representación de la arquitectura inicial.

6.4.1 MODELO DE DOMINIO

En un modelo de dominio se van a representar los diferentes elementos o conceptos con los que va a trabajar nuestro sistema y como están relacionados entre ellos, pero no se representan componentes del software ni objetos.

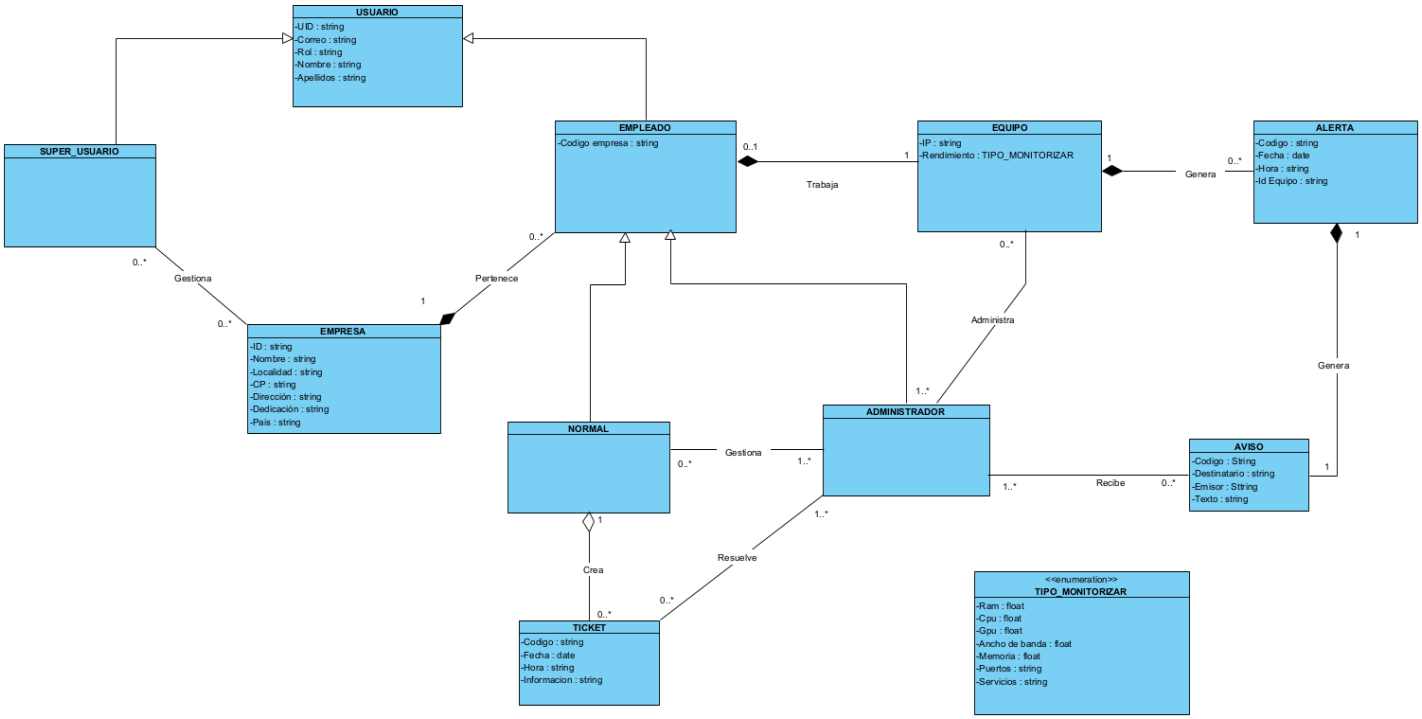

Figura 25 Diagrama de clases o modelo de dominio

6.4.2 REALIZACIÓN DE LOS CASOS DE USO

La realización de los casos de uso consiste en la representación de cómo se producen los diferentes casos de uso mediante el intercambio de mensajes entre objetos (interfaz, controlador y modelo). Para esta representación se hace uso de los diagramas de secuencia.

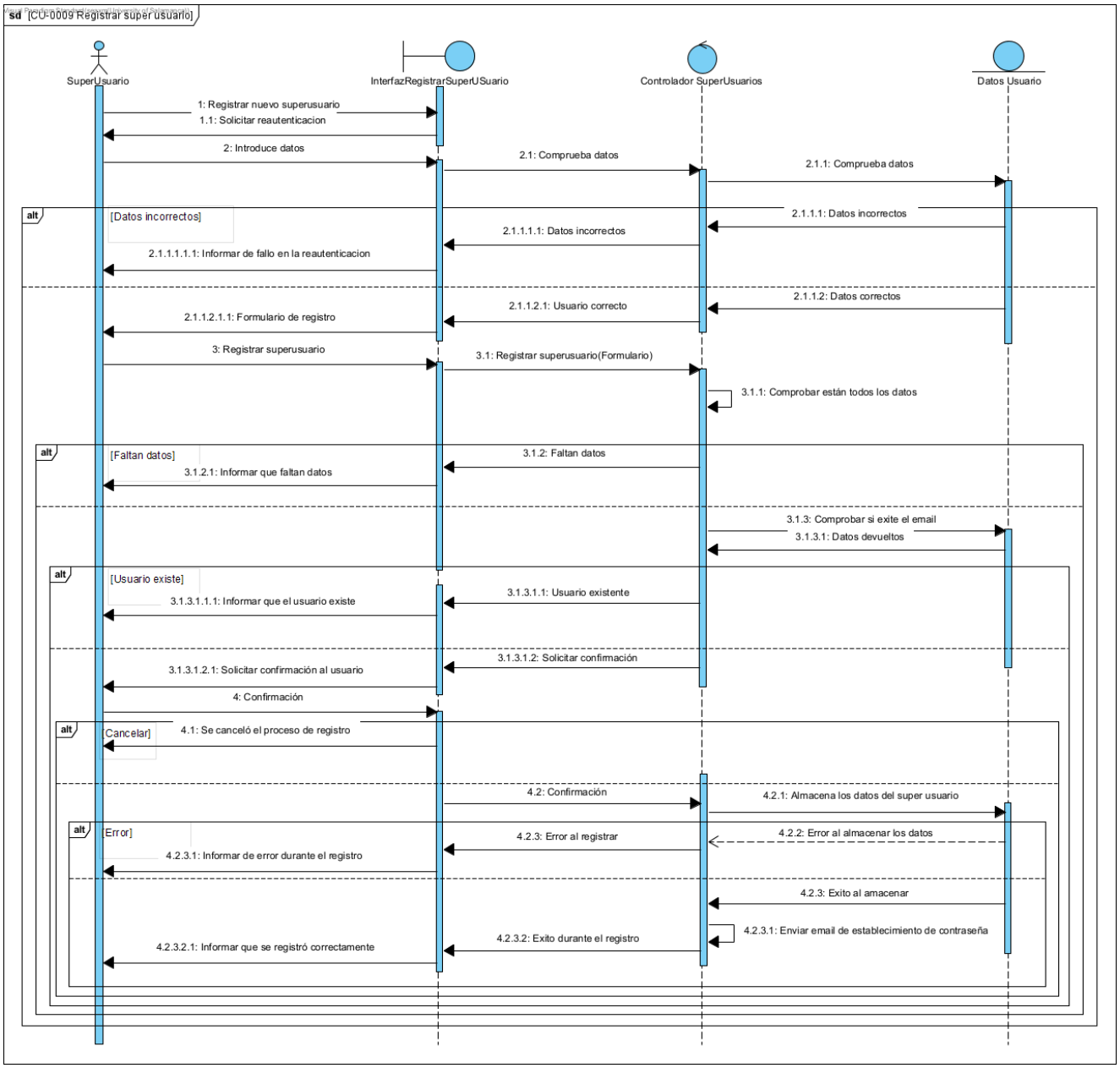

Figura 26 Diagrama de secuencia de un caso de uso

6.4.3 CLASE DE ANÁLISIS

Consiste en la agrupación de las diferentes clases de análisis en sus correspondientes paquetes para relacionarlos en función de su comunicación.

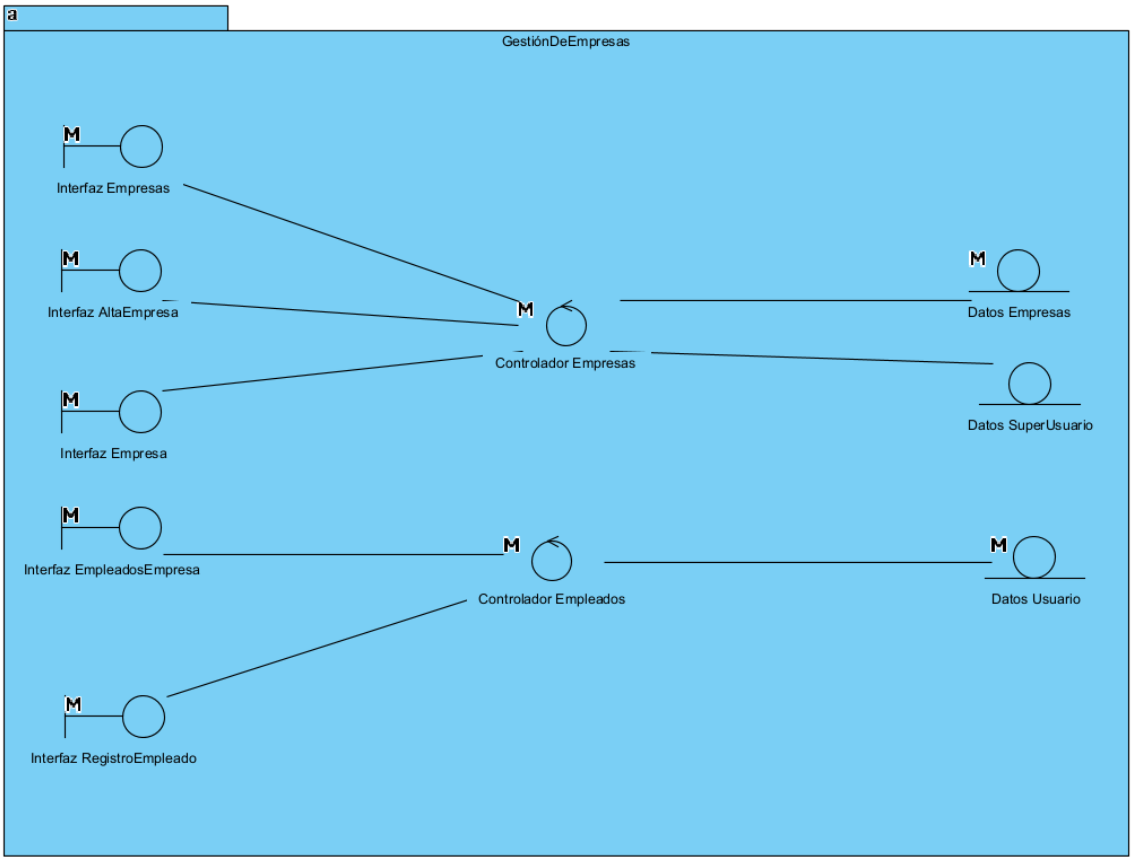

Figura 27 Ejemplo de clase de análisis

6.4.4 VISTA DE ARQUITECTURA

Cada una de las clases se encontrará contenida dentro de un paquete en función de si forma parte del modelo, de la vista o del controlador, con tal de seguir el patrón MVC (Modelo Vista Controlador), que es un patrón arquitectónico que nos permite determinar cómo van a estar distribuidas las diferentes clases.

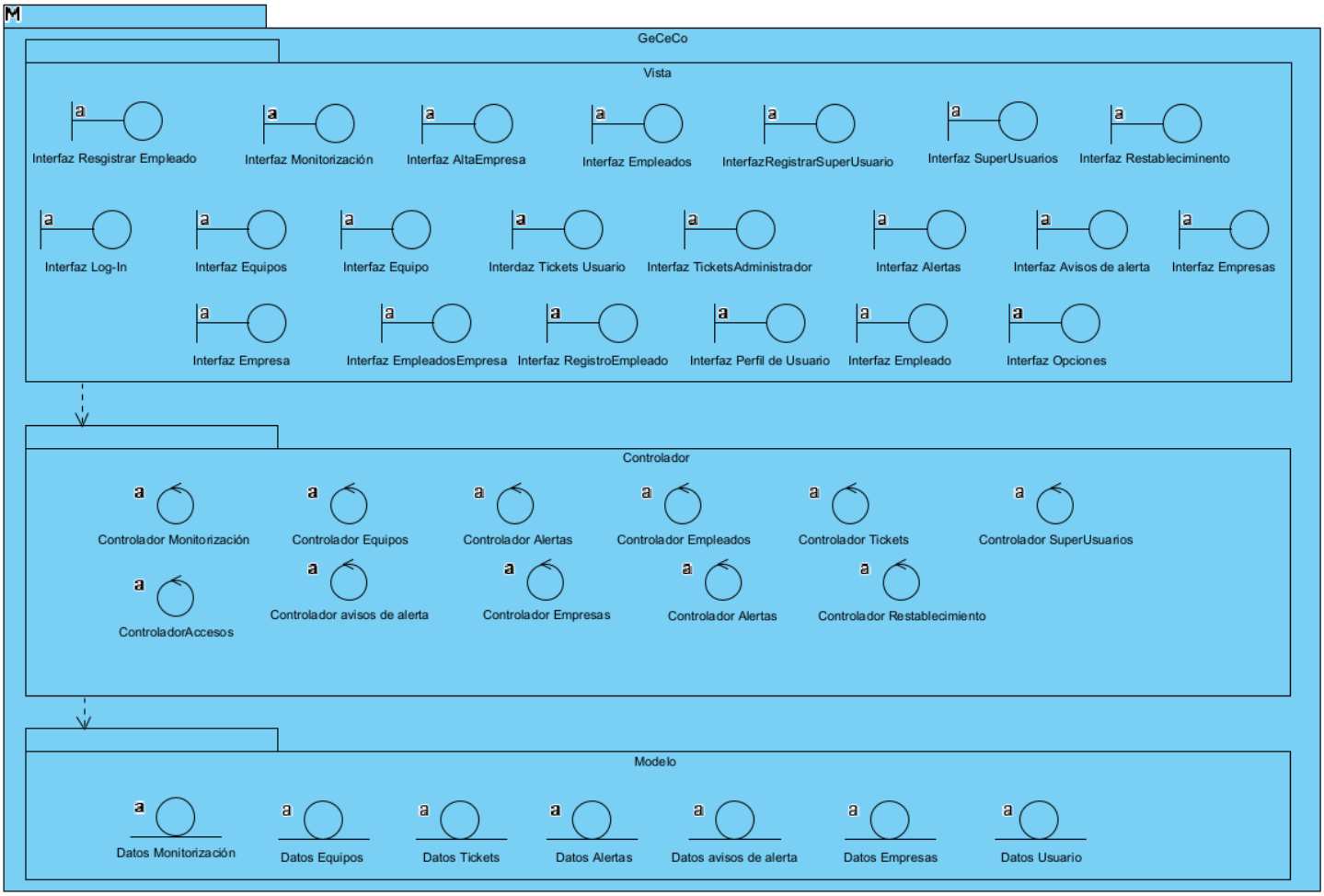

Figura 28 Vista de la arquitectura del proyecto

6.5 DISEÑO DEL SISTEMA SOFTWARE

Esta etapa del proyecto se centra principalmente en la elaboración de una representación cercana al sistema final, es lo que se conoce como el dominio de la solución. La información completa puede encontrarse en el Anexo IV del presente proyecto.

La estructura seguida en esta etapa es la siguiente:

1. **Modelo de diseño**
   - **Patrones arquitectónicos:** Se especiican los patrones utilizados para el diseño y desarrollo de la aplicación.
   - **Subsistemas de diseño:** Especificación y división de los diferentes paquetes por los que se compone la aplicación y se definen las relaciones que existen entre ellos.
   - **Clases de diseño:** Se especifican las diferentes clases por las que está compuesta la aplicación, así como los diferentes datos y atributos con los que trabajan.
   - **Vista arquitectónica:** Representación del diseño estructural de la aplicación
   - **Realización de casos de uso:** Similares a los diagramas de secuencia anteriormente ejemplificados, mediante los cuales se representa de forma más detallada y cercana el funcionamiento de los diferentes casos de uso.

2. **Diseño de la base de datos:** Representación de la información con la que trabaja la aplicación.
3. **Modelo de despliegue:** Representación de los diferentes nodos y su relación dentro del sistema.

6.5.1 MODELO DE DISEÑO

Modelo de objetos que representa la realización física de los casos de uso, tratando la forma en que los requisitos funcionales y no funcionales, tienen impacto en el sistema en desarrollo.

El modelo de diseño se representará mediante un sistema que describe el subsistema de más alto nivel (García Peñalvo, García Holgado, & Vázquez Ingelmo, Flujos de trabajo del proceso unificado)

6.5.1.1 PATRONES ARQUITECTÓNICOS

6.5.1.1.1 MODELO-VISTA-MODELO DE VISTA(MVVM)

Dado que la tecnología utilizada para el desarrollo de dicha aplicación es Vuejs, el patrón utilizado es Modelo-Vista-ModeloDeVista (MVVM).

Este es un patrón de diseño utilizado para separar los datos con los que trabaja la aplicación de la vista del usuario. Con la principal ventaja de que si se produce una modificación en los datos (el modelo) este cambio se verá representado en la interfaz (vista) (EcuRed, s.f.).

Este patrón está dirigido al desarrollo de aplicaciones que trabajan con eventos (como podría ser un click de ratón en un botón).

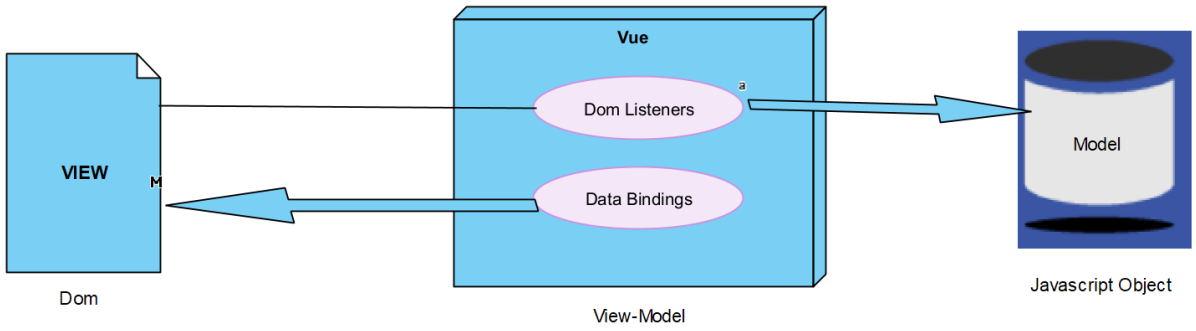

Figura 29 Representación VUEjs

Las diferentes partes son las siguientes:

- **Modelo:** Las clases del modelo son las responsables de la representación del modelo de dominio, y por lo tanto de la representación del modelo de negocio y el modelo de datos. En ningún caso se ocupará de mostrar nada que deba mostrar la vista, así como jamás se comunicará directamente con esta.

- **Vista:** La vista es la encargada de la representación de la información y de los diferentes elementos visuales de la aplicación. En ningún caso se encargará de la recuperación y manejo de la información, y tampoco se comunicará nunca directamente con el Modelo.

- **Modelo de Vista:** Implementará toda la lógica de recuperación y tratamiento de datos, así como las diferentes funcionalidades que posee la aplicación. Será la capa intermedia entre el Modelo y la Vista encargada de la comunicación entre ambas.

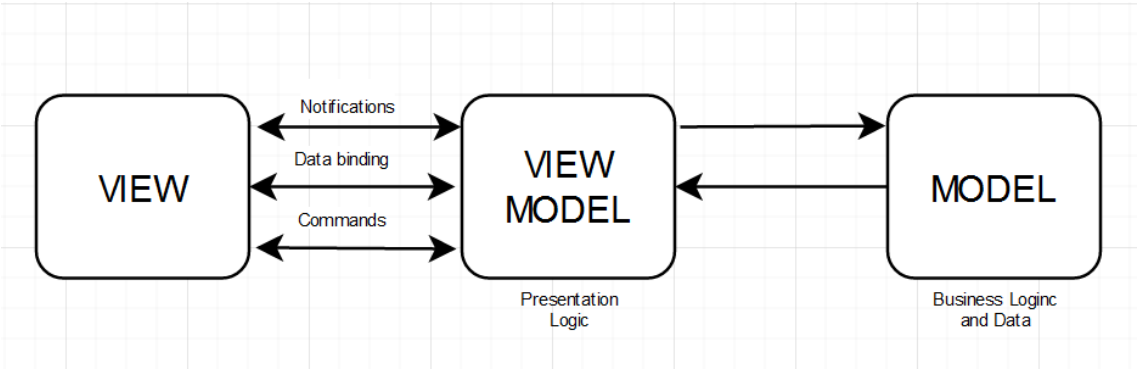

Figura 30 Esquema del patrón MVVM

6.5.1.1.2 SCHEDULED TASKS PATTERN (PATRÓN DE TAREAS PROGRAMADAS)

Una tarea programada constará de tres elementos: la propia tarea, los argumentos que definen cuando o cada cuanto se ejecutará la tarea y las ejecuciones que realiza la misma.

Comúnmente son implementados mediante el uso de cron en los servidores Linux.

Se deberán seguir una serie de pautas a la hora de implementarlas:

1. No exponer los cronjobs en internet
2. Al ejecutar la tarea no se deben comprobar los criterios para saber si tiene o no que ejecutarse, esto se hará fuera de la tarea.
3. La tarea tan solo deberá realizar la actividad o la acción para la que fue creada y nada más.

(Ali, 2016)

En este caso dicho patrón es aplicado para la creación de un “Daemon” encargado de proporcionar una monitorización periódica sobre el equipo que lo ejecuta, este Daemon se implementa mediante el uso del módulo nodecron.

6.5.1.2 SUBSISTEMA DE DISEÑO

Se representa mediante un diagrama los paquetes del sistema, así como las diferentes relaciones entre los mismos.

Esta división está repartida mediante los diferentes componentes del patrón MVVM. Dichos paquetes serán:

- **Aplicación web:** Es la propia aplicación o sistema en desarrollo, encargada de llevar a cabo la gran mayoría de las funcionalidades proporcionadas por el sistema.
- **Backend:** Encargado de las ejecuciones en servidor, para el uso de las “cloud functions” de firebase, así como de algunas otras instrucciones que no pueden ser ejecutadas por los clientes
- **Monitorizador:** Script ejecutado por el usuario encargado de realizar la monitorización periódica para la recolección de información.
- **Router:** Es el encargado de mantener las rutas protegidas de tal forma que solo los usuarios autorizados puedan acceder a determinadas rutas.

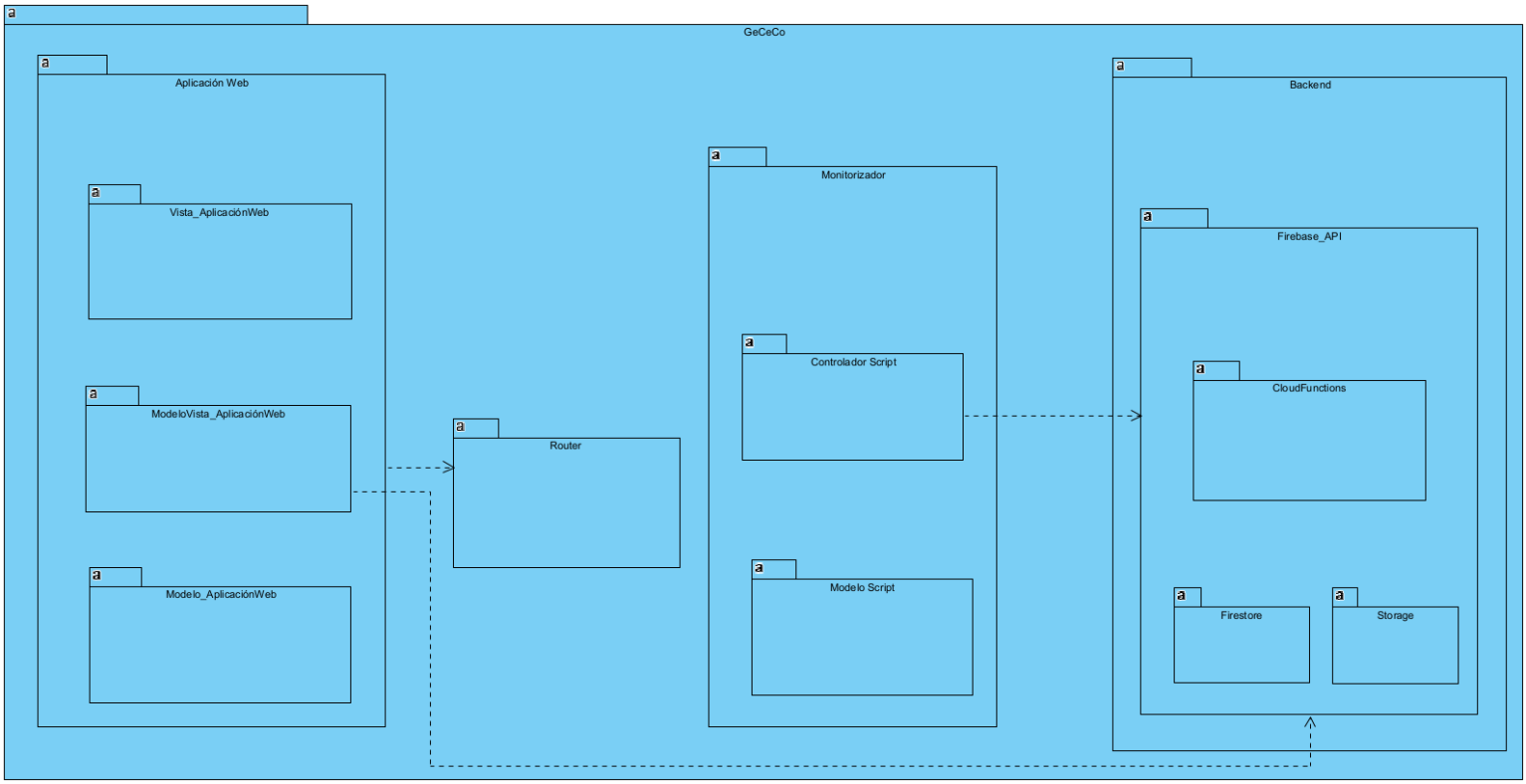

Figura 31 Subsistema de diseño

6.5.1.3 CLASES DE DISEÑO

En el siguiente apartado se define el contenido de cada uno de los paquetes, como ya se indicó, la información completa de esto se encuentra en el Anexo IV del proyecto, por lo tanto, tan solo se mostrarán las clases de diseño de la aplicación web a modo de ejemplo.

6.5.1.3.1 VISTA

Componentes Vue utilizados para proporcionar todas las vistas.

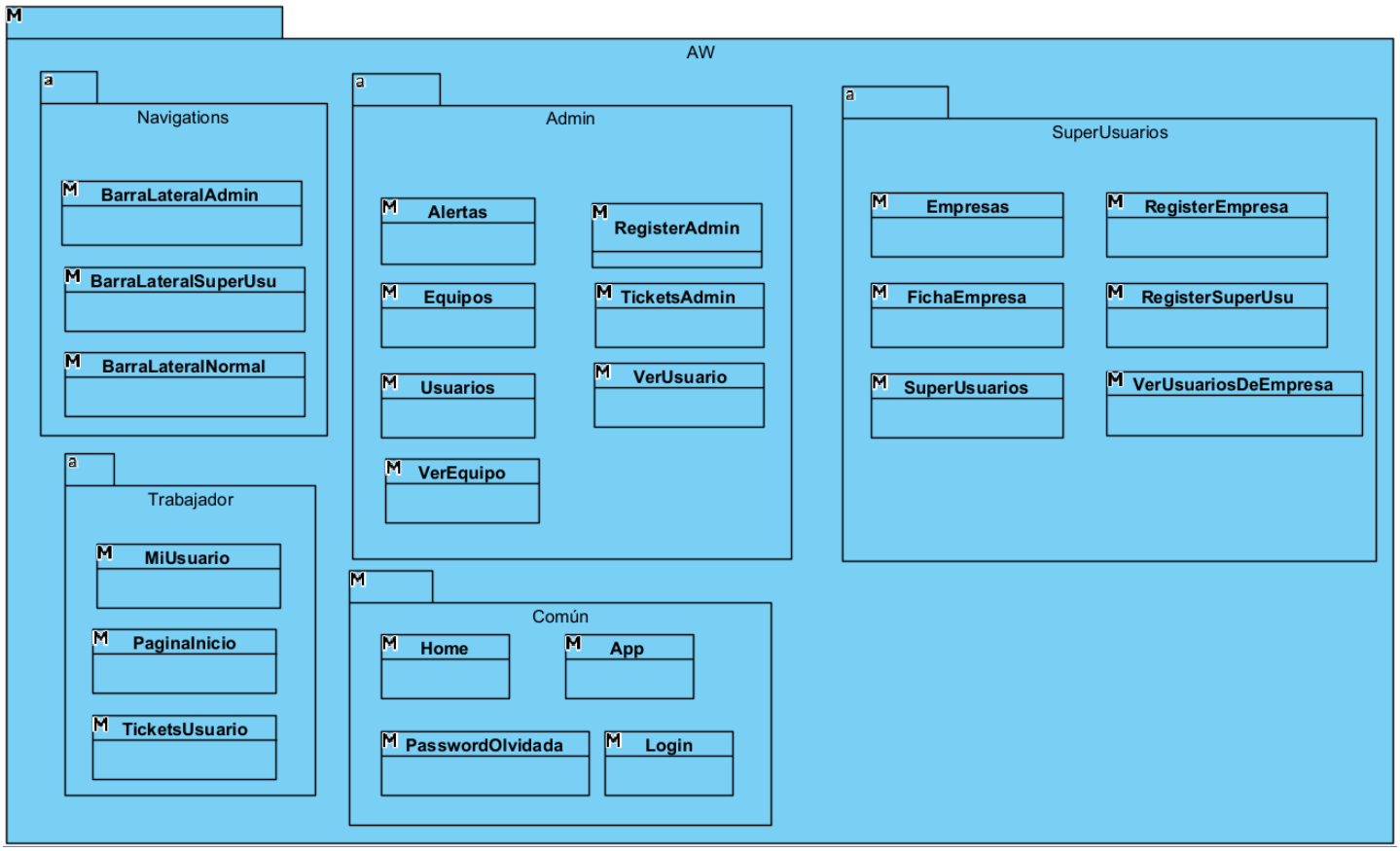

Figura 32 Vista aplicación web

6.5.1.3.2 MODELO-VISTA

Componentes Vue de la aplicación web, así como los métodos que manejan.

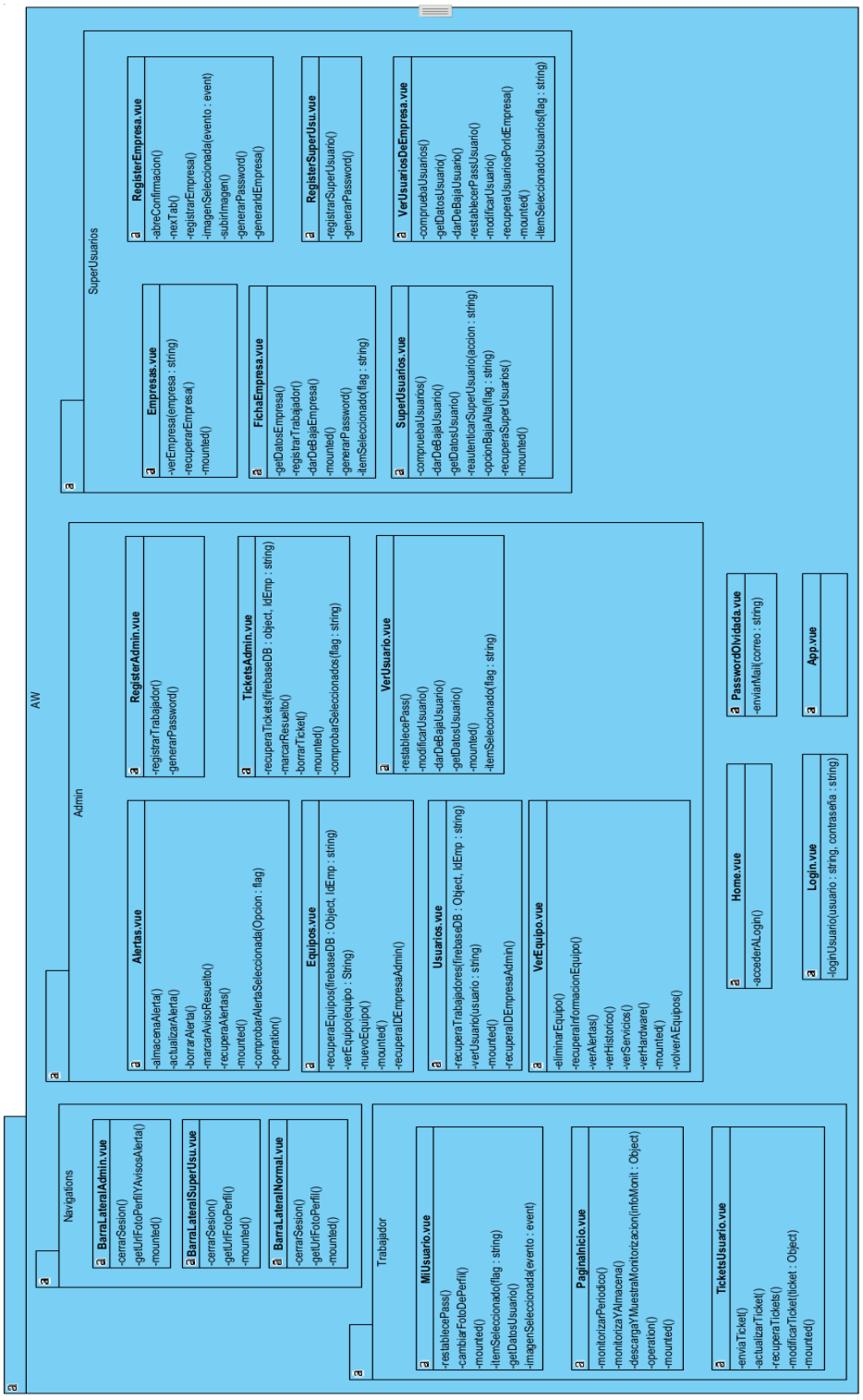

Figura 33 Modelo vista aplicación web

6.5.1.3.3 MODELO

Modelos con los que se trabaja.

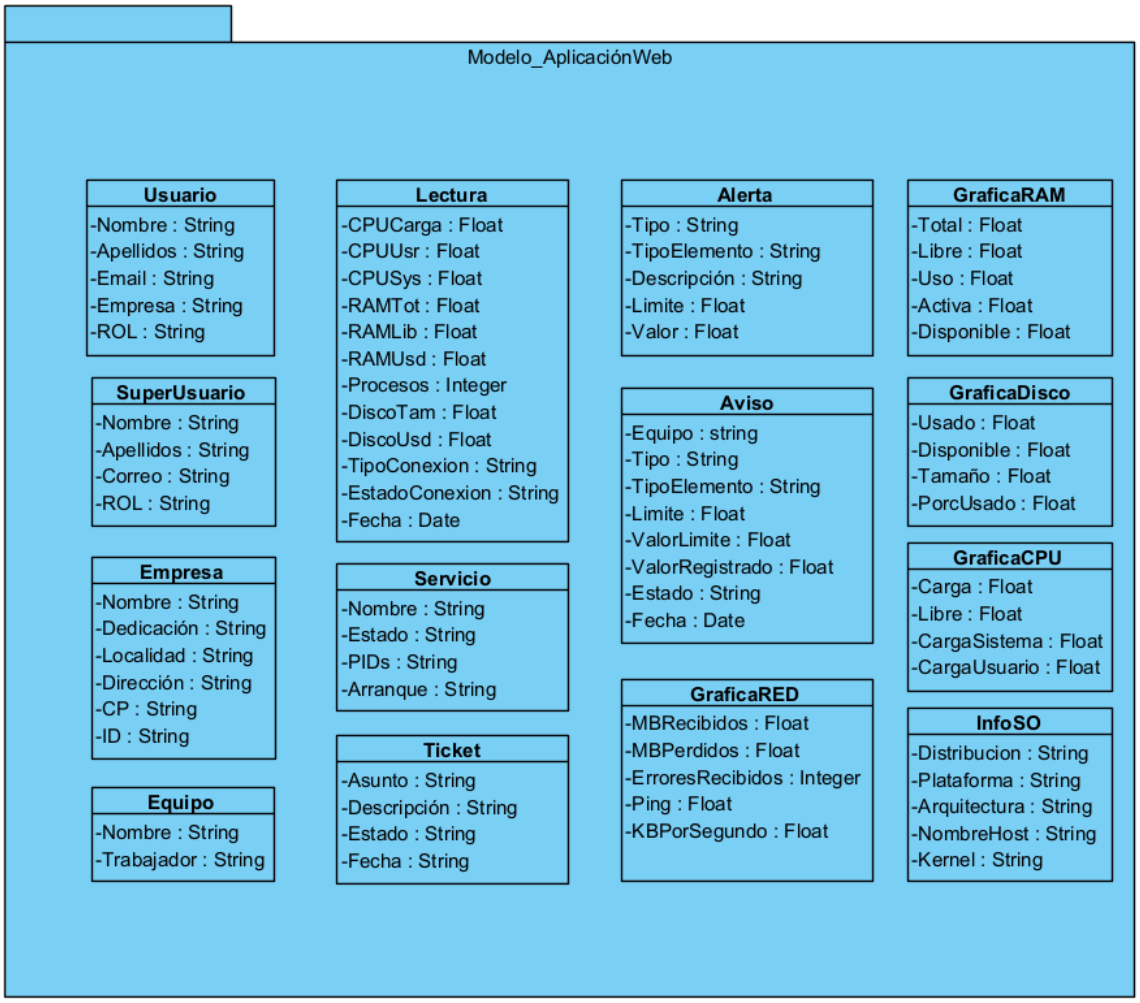

Figura 34 Modelo aplicación web

6.5.1.4 VISTA ARQUITECTÓNICA

Representación del patrón MVVM sobre el subsistema de diseño.

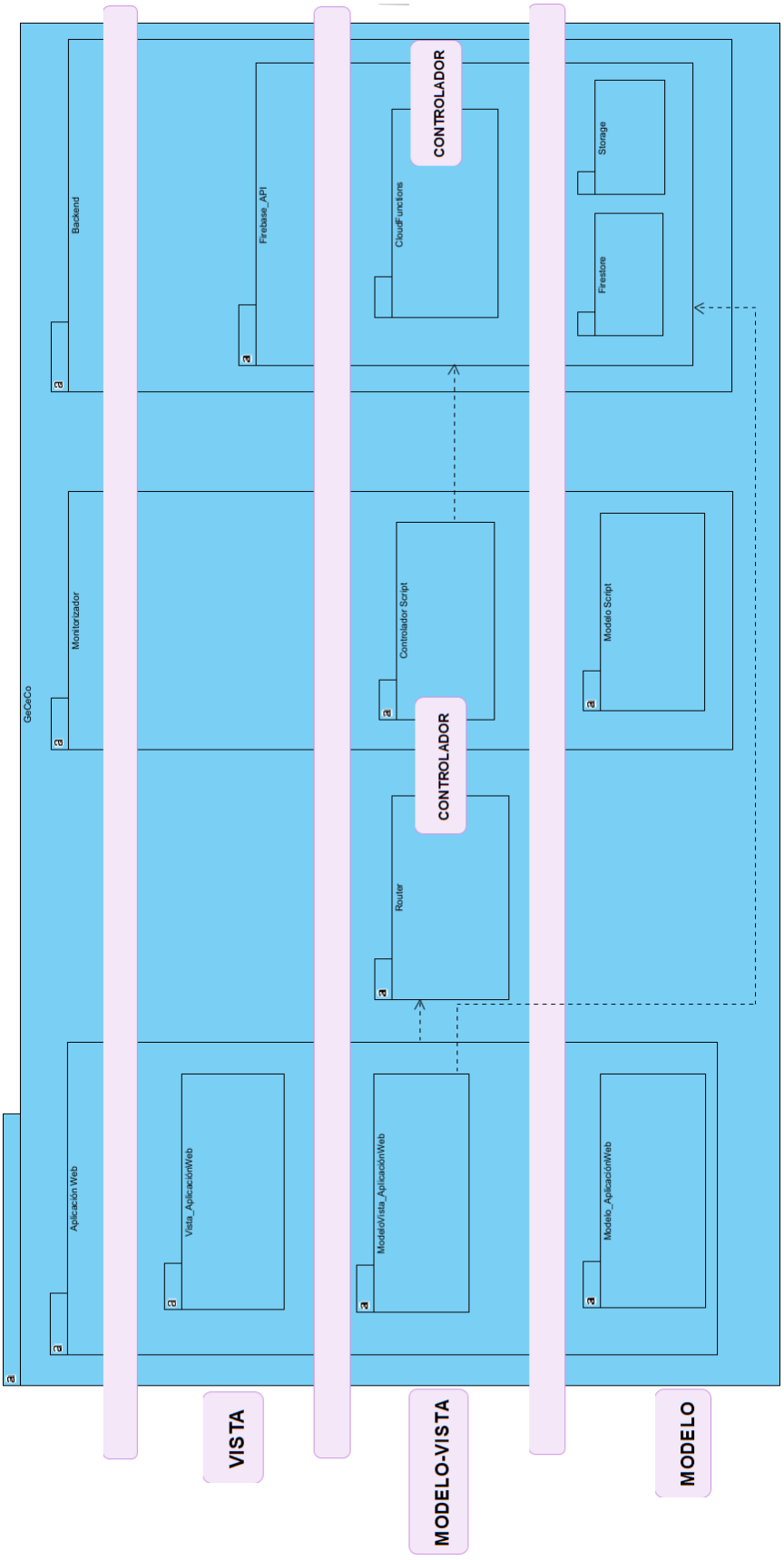

Figura 35 Vista arquitectónica

6.5.1.5 REALIZACION DE LOS CASOS DE USO

Se utilizan para representar el intercambio de mensajes entre los diferentes componentes definidos anteriormente, es lo que se conoce también como diagramas de secuencia.

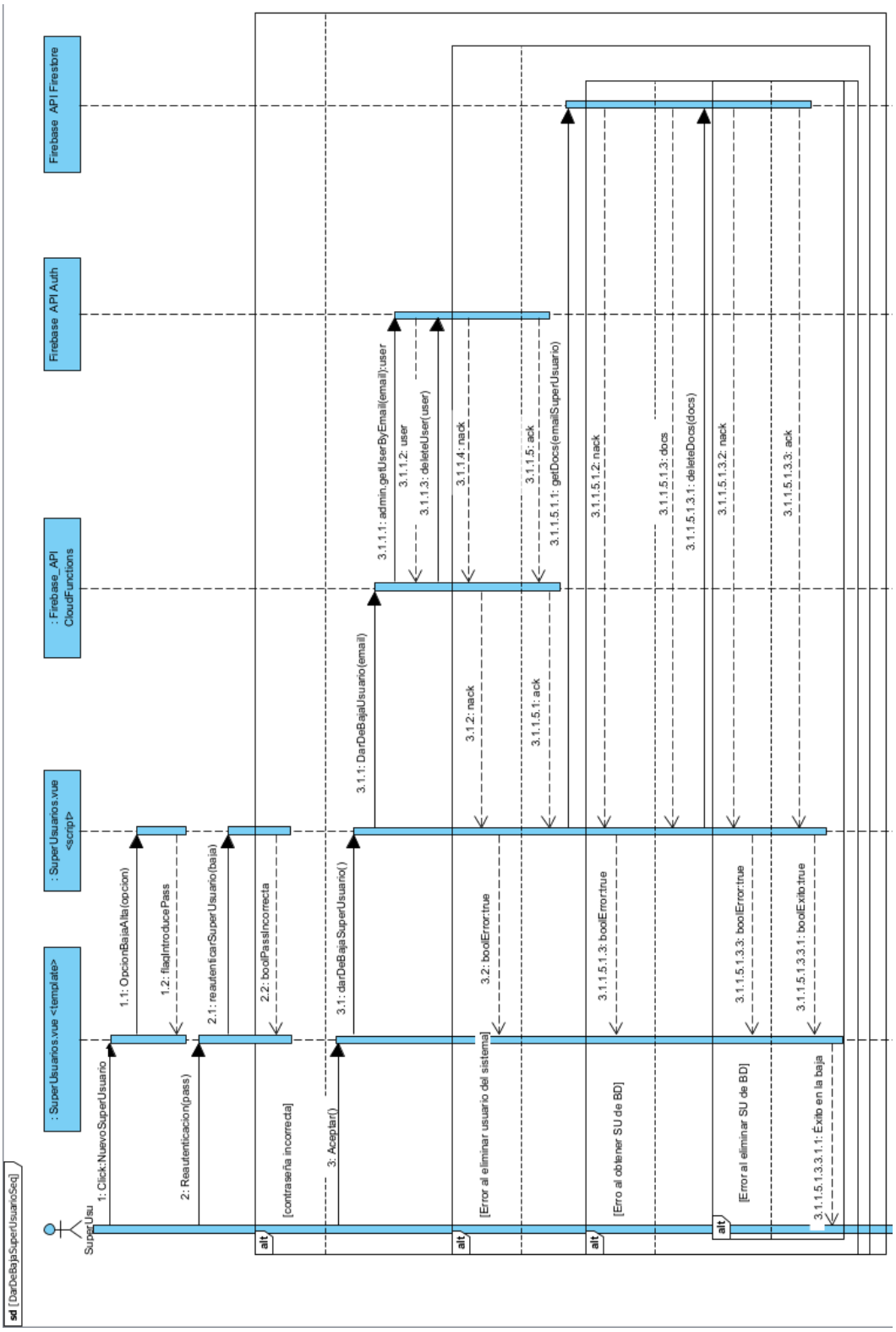

Figura 36 Ejemplo de diagrama de secuencia

6.5.2 DISEÑO DE LA BASE DE DATOS

Para la base de datos se usa una base de datos no relacional (NoSQL), como es la base de datos de Firestore. Además, dentro de un documento podremos crear o almacenar a su vez una colección con sus correspondientes documentos. Para la representación de la información con la que trabajará el sistema, se ha elaborado un diagrama comúnmente utilizado para representar el contenido de bases de datos SQL (relacionales), sin embargo, no se pretende representar una base de datos relacional, simplemente se busca representar las diferentes colecciones, así como los atributos de cada uno de los documentos.

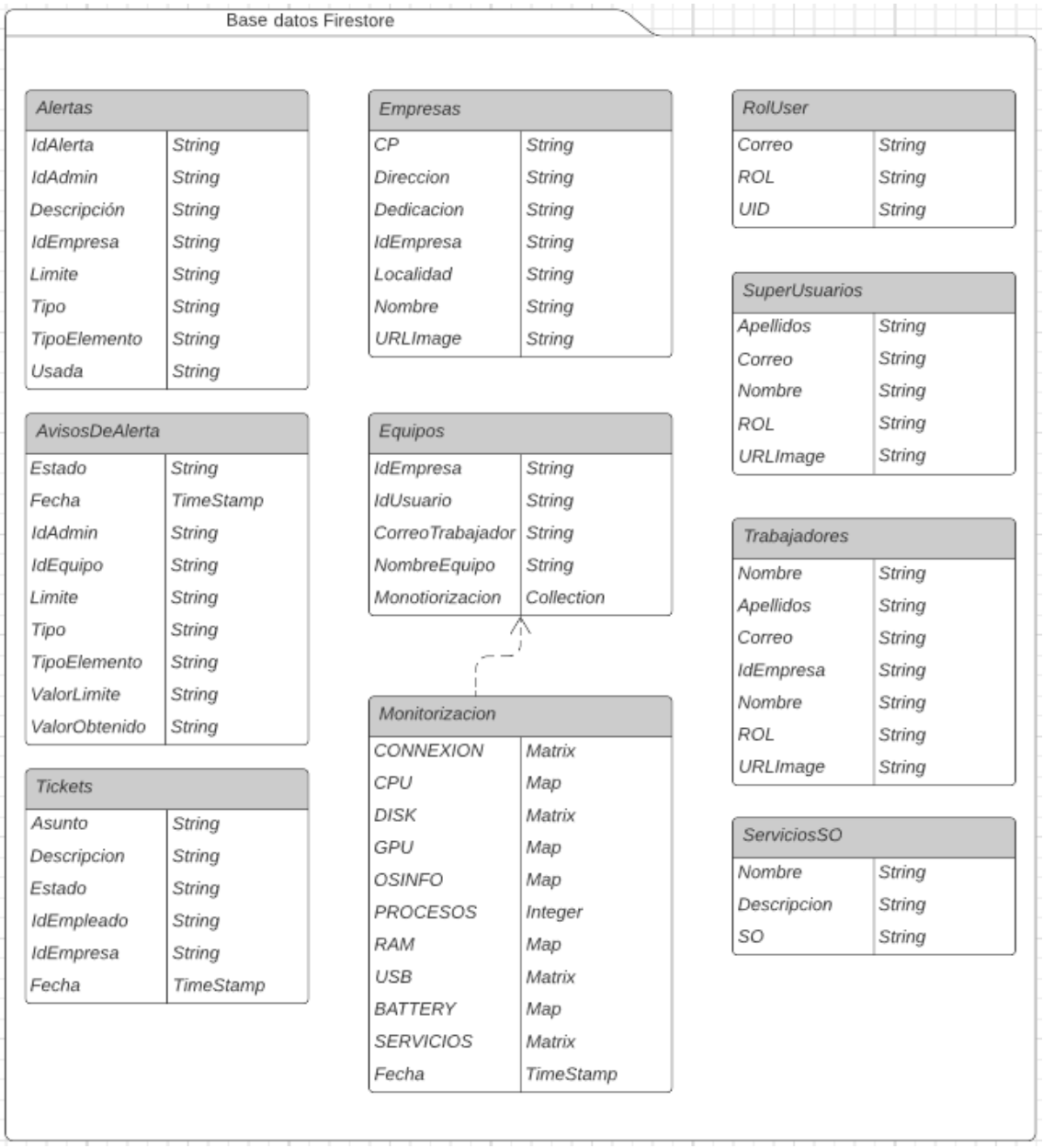

Figura 37 Diseño de la base de datos

6.5.3 MODELO DE DESPLIEGUE

A continuación, se muestra el modelo de despliegue de la aplicación, un modelo de despliegue nos permite la representación física de los diferentes componentes software en los nodos físicos (dispositivos) que conforman la red (DiagramasUML, s.f.).

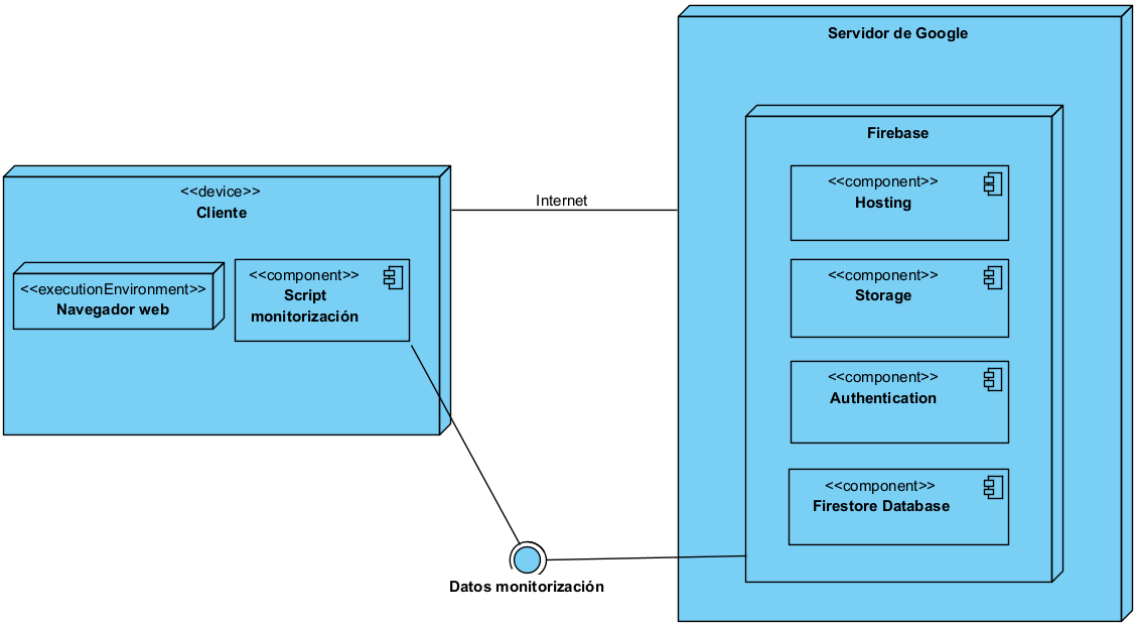

Figura 38 Diagrama de despliegue

En este caso el cliente se conectará a la aplicación web mediante un navegador web utilizando los servidores de Google. La aplicación hace uso del componente hosting de “Firebase” para su despliegue. Además, también hará uso de los componentes “Storage”, “Authentication” y “Firestore” para su funcionamiento.

Por otra parte, el script de monitorización se conectará con la base de datos de “Firebase” (Firestore) para proporcionar los datos leídos de los equipos.

6.6 IMPLEMENTACIÓN

Fase donde se realizó la codificación propiamente dicha, tanto de la aplicación web como del script encargado de la monitorización de los equipos, comenzando por la primera.

Si se quiere consultar la documentación del código se puede encontrar en el Anexo V del presente trabajo, donde se detalla el manual del programador.

6.6.1 APLICACIÓN WEB

A continuación, se irán enumerando y explicando cómo se han desarrollado las diferentes funcionalidades de la aplicación web, así como las herramientas y módulos utilizados. Las diferentes vistas correspondientes a cada una de las funcionalidades han sido implementadas a la vez que estas. Para lograr una apariencia más profesional y bonita se usa como se indicó anteriormente el módulo Vuetify.

Al iniciar el desarrollo de la aplicación web se escogió la paleta de colores que se ha utilizado y se elaboró un logo para la misma. Se escogió el nombre “GeCeCo” en base al título del trabajo que es “Gestión de Centros Computacionales”.

Figura 39 Logo de la aplicación web

6.6.1.1 HERRAMIENTAS Y MÓDULOS

- Firebase
- Firebase-admin
- Java Script
- Nodejs
- Npm
- Router
- Vue
- Vuetify

6.6.1.2 GESTIÓN DE USUARIOS

Fue lo primero a desarrollar, puesto que no tenía experiencia previa en el desarrollo de aplicaciones web ni nociones previas sobre el funcionamiento de Firebase así como sus funcionalidades, era una buena forma de familiarizarme con la creación, bajas, accesos, y modificación de los usuarios utilizando las funciones de Firebase. Además, también me permitía irme introduciendo con el uso de la base de datos de Firestore.

Comencé añadiendo a la base de datos las colecciones que se encargada de controlar los roles de los usuarios, los usuarios registrados y los super usuarios.

Figura 40 Colecciones para la gestión de usuarios

Una vez tenía esto programé la funcionalidad que permite acceder a los usuarios y cree tres vistas, una a la que solo podían acceder los super usuarios, otra los administradores y otra los usuarios normales.

De esta forma comencé a programar las rutas protegidas, con el fin de que solo los usuarios permitidos puedan acceder a determinadas ventanas de la aplicación web en función de su rol.

A continuación, implementando las Cloud Functions de Google y utilizando el módulo de administrador de Firebase, programé las funcionalidades que permiten dar de alta un usuario, darlo de baja y modificar su información, como su correo electrónico o su contraseña.

Utilizando el storage de Firebase se implementó la posibilidad de que los usuarios puedan establecer una imagen de perfil.

Una vez terminé con la gestión de usuarios me puse con la gestión de tickets.

6.6.1.3 GESTIÓN DE TICKETS

Escogí la gestión de tickets como segunda parte a desarrollar puesto que me ayudaba a aprender cómo funcionan los componentes en vuetify, así como a recuperar información de las bases de datos y poder mostrarlos dentro de los mismos.

Añadí entonces una nueva colección a la base de datos de Firestore, para almacenar los diferentes tickets que pueden crear y modificar los usuarios normales, y que pueden también eliminar los administradores.

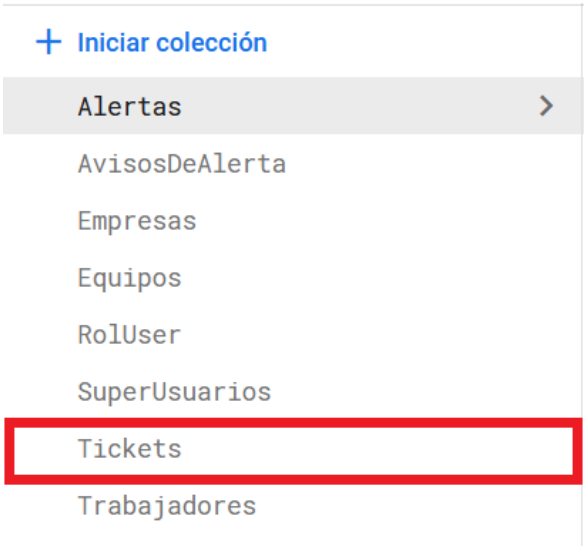

Figura 41 Colección para la gestión de tickets

De esta forma entendí en qué consistían los documentos dentro de las bases de datos no relacionales de Firestore y aprendí a crearlos modificarlos y eliminarlos. Obteniendo de esta forma una gestión de tickets funcional en la que solo los administradores de una empresa pueden acceder a los tickets de su misma empresa, pero no a los del resto y actualizando el estado de los tickets en tiempo real.

6.6.1.4 GESTIÓN DE EMPRESAS

Se continuó con la parte de gestión de empresas implementando la creación de empresas, así como las bajas de las mismas. Incluyendo las bajas de todos los usuarios del sistema en el caso de eliminar una empresa. Para ello lo primero que se hizo fue añadir una nueva colección a la base de datos de Firestore. También utilizando el servicio de storage se implementó la posibilidad de establecer una imagen corporativa de la empresa que podría visualizarse al acceder a la información de la empresa.

Además, se implementó la funcionalidad por la cual se permite que los super usuarios puedan gestionar los usuarios de las empresas a través de estas, tanto a administradores como a usuarios normales.

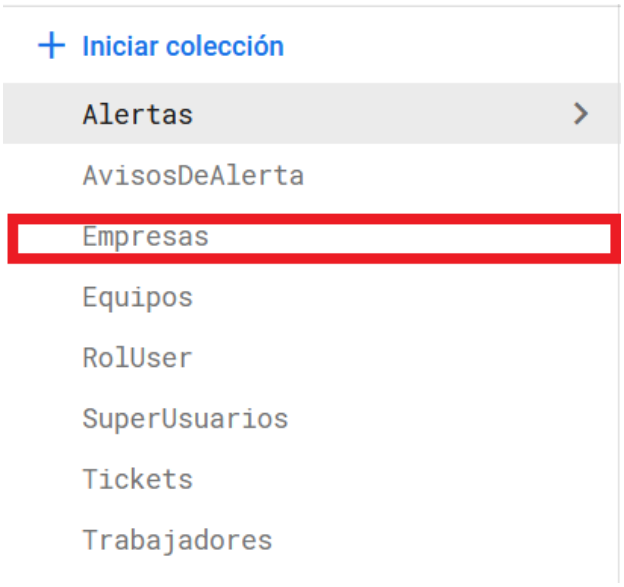

Figura 42 Colección para la gestión de empresas

6.6.1.5 GESTIÓN DE ALERTAS

Una vez se tenía ya un cierto control sobre como funcionaban las Cloud Functions así como las operaciones con la base de datos se optó por desarrollar una de las funcionalidades más relevantes del sistema, que es tanto la creación, modificación y eliminación de alertas que pueden ser disparadas, como el almacenamiento de la información que se muestra al administrador en caso de que una alerta sea disparada, para ello se crearon dos nuevas colecciones en la base de datos de Firestore, una que almacenaría las alertas creadas por los administradores y otra que almacenaría los avisos que generasen dichas alertas tras una monitorización conflictiva

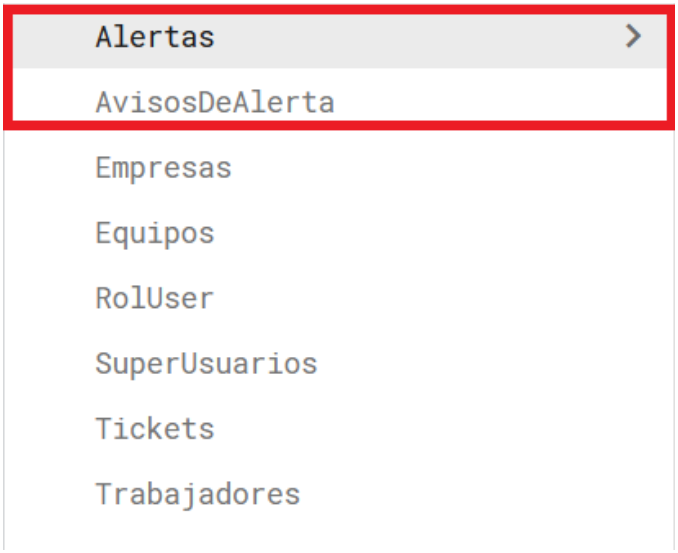

Figura 43 Colección para la gestión de alertas

Además, se creó una colección de datos de forma manual en la que se han introducido un número considerable de servicios tanto para Linux como para Windows, esta colección se usa para la creación de alertas también y se explicará más adelante.

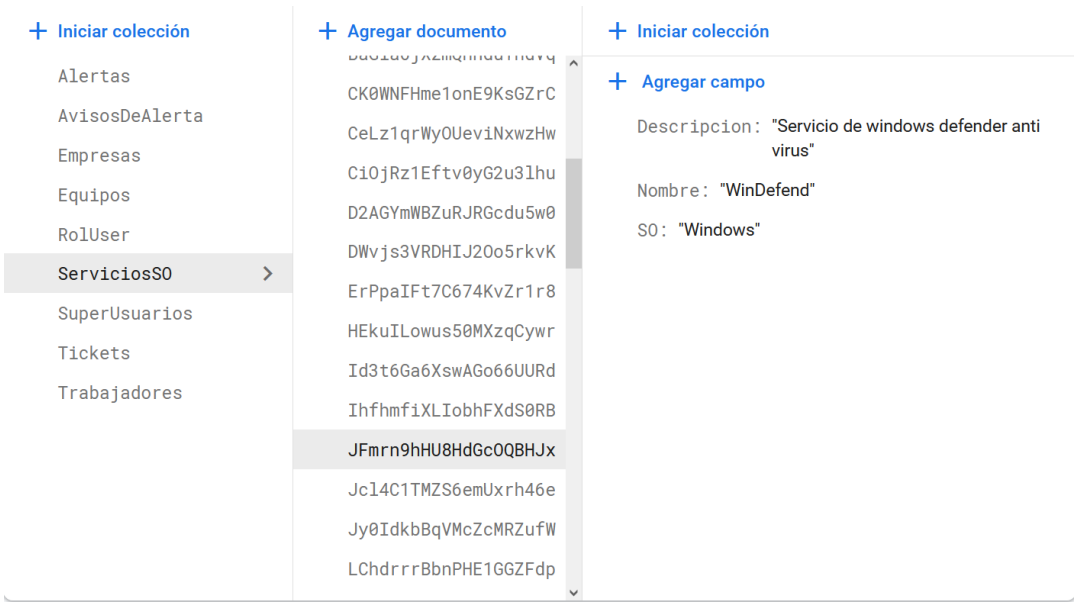

Figura 44 Coleccion de servicios para monitorizar

Para la creación de los avisos de alerta se añadió una nueva función a las Cloud Función cuyo cometido es que cada vez que se añade una nueva monitorización a la base de datos (todavía no explicado), comprobando la empresa a la que pertenece el equipo, se evalúan los resultados con los parámetros de las alertas creadas por los administradores, de tal forma que si se cumple la condición de alguna o de algunas alertas, se genera un aviso de alerta que es almacenado y recibido por el administrador correspondiente que creó la alerta que ha saltado.

6.6.1.6 GESTIÓN DE EQUIPOS

Para la gestión de los equipos se creó una nueva colección dentro de la base de datos. Cada uno de los documentos se corresponde con un equipo registrado dentro del sistema. Y cada uno de estos equipos contará con una colección de monitorizaciones.

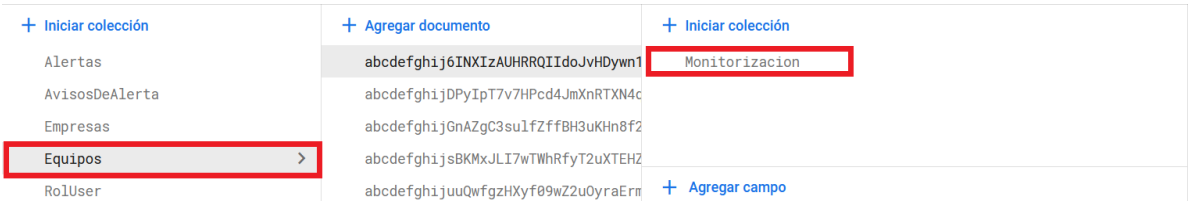

Figura 45 Colección de equipos y de monitorizaciones

Se implementó la posibilidad de crear y dar de baja equipos asociados a usuarios que existiesen ya dentro del sistema.

Tras una monitorización la información obtenida es almacenada en la base de datos y posteriormente analizada con las diferentes alertas existentes. De esta forma obtenemos un histórico de lecturas para un equipo que puede ser consultado y analizado.

6.6.1.7 DESPLIEGUE DE LA APLICACIÓN

Por último, se realizó el despliegue de la aplicación para que se mantenga funcional y accesible a través del navegador web, para esto se hizo uso de unas de las herramientas que nos proporciona Firebase (Hosting). Tan solo se tuvo que hacer una “build” del proyecto y desplegarlo con el comando “deploy” de Firebase. Esta plataforma nos proporciona una forma sencilla de desplegar nuestras aplicaciones, así como de poder volver a versiones anteriores si fuese necesario.

6.6.2 SCRIPT DE MONITORIZACIÓN

Para el script de monitorización se hizo uso del lenguaje Java Script y se utilizó un módulo llamado systeminformation. Este módulo nos permite la recolección de información referente al equipo que ejecuta los métodos que proporciona. También se utiliza el módulo nodecron, programando la propia monitorización como una tarea que se repite periódicamente en el tiempo y que va realizando las lecturas.

6.6.2.1 HERRAMIENTAS Y MÓDULOS

- Firebase
- Java Script
- Nodejs
- Npm
- Nodecron
- systemctl

6.7 PRUEBAS

Las pruebas exhaustivas son una parte fundamental durante el desarrollo de un software, permiten encontrar errores, bugs y funcionamientos incorrectos. Contribuyen a mejorar la fiabilidad y la calidad del sistema software.

Para cada funcionalidad implementada se han realizado pruebas unitarias exhaustivas, para confirmar el correcto funcionamiento. Probando a introducir diferentes tipos de datos, diferentes acciones o secuencias de acciones, proporcionar información incorrecta, e incluso buscar producir directamente los errores. De esta forma se ha podido limpiar la aplicación de fallos y en algunos casos volverla tolerante a estos. Las pruebas unitarias, nos permiten identificar pequeños fallos que, sin la realización de las mismas, podrían pasar desapercibidas en un primer momento pero que luego arrastrasen al sistema completo graves problemas funcionamiento.

Cuando un subsistema estaba completo se han realizado pruebas de integración con los anteriores subsistemas ya implementados y que se conocía que debían funcionar correctamente.

Además, una vez se iban teniendo los diferentes subsistemas completos se han ido probando en su totalidad como se haría en un momento de trabajo real, así como las pruebas de interacción entre distintos subsistemas, como podría ser la creación de una empresa y posteriormente la creación de varios usuarios para esa empresa, creación de sus respectivos equipos, realizar monitorizaciones, etc.

Se probó además la compatibilidad con distintos dispositivos y se determinó que el sistema es viable usarlo en Tablet, PC y ordenador portátil.

6.8 FUNCIONALIDAD DE LA APLICACIÓN Y DEL SCRIPT

A continuación, se hará un breve resumen de las funcionalidades tanto de la aplicación web como del script. Para obtener la información completa en lo referente a esto se puede consultar el Anexo VI del presente trabajo (Manual del usuario).

6.8.1 GESTIÓN DE USUARIOS

El sistema permite dar de alta, dar de baja y modificar la información de los usuarios del sistema. Un administrador puede realizar estas acciones para los usuarios de su empresa y los Super Usuarios pueden realizarlo para los usuarios de cualquier empresa. Todos los usuarios tienen la capacidad modificar su contraseña dentro del sistema, así como modificar su foto de perfil. Además, existe implementado un sistema de login, de cierre de sesión y de restablecimiento de contraseña vía email en caso de que sea necesario.

Este es un ejemplo de la ventana para acceder al sistema, el usuario introducirá su usuario y contraseña y accederá. En caso de que las credenciales sean incorrectas o incompletas el sistema informará de ello al usuario a través de un mensaje por pantalla.

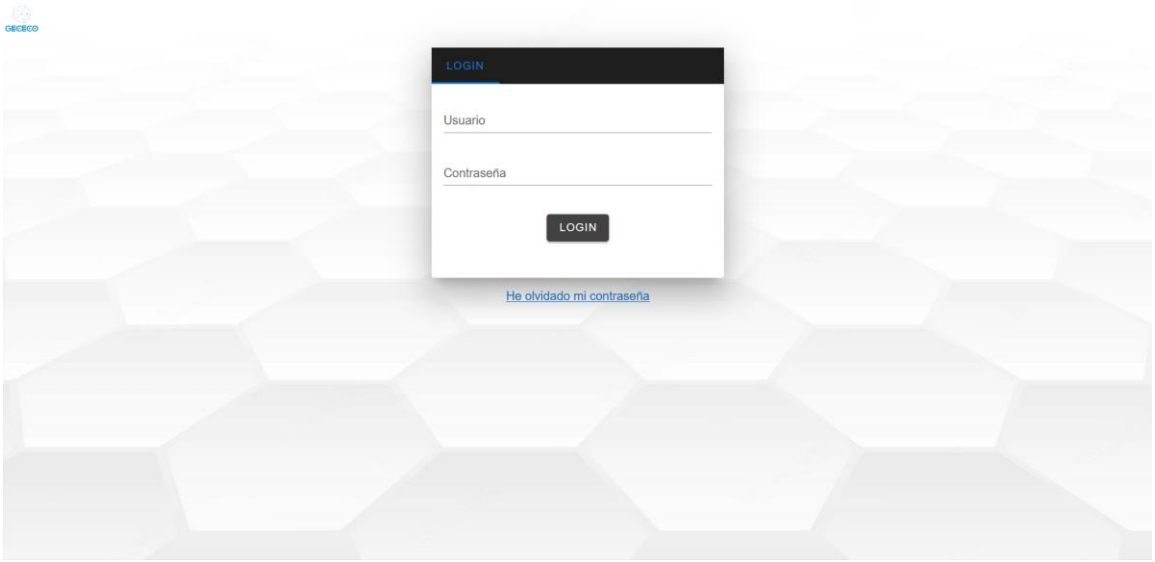

Figura 46 Pantalla de Log-In

Ejemplo del envío de un email de restablecimiento de contraseña en base a un correo electrónico proporcionado.

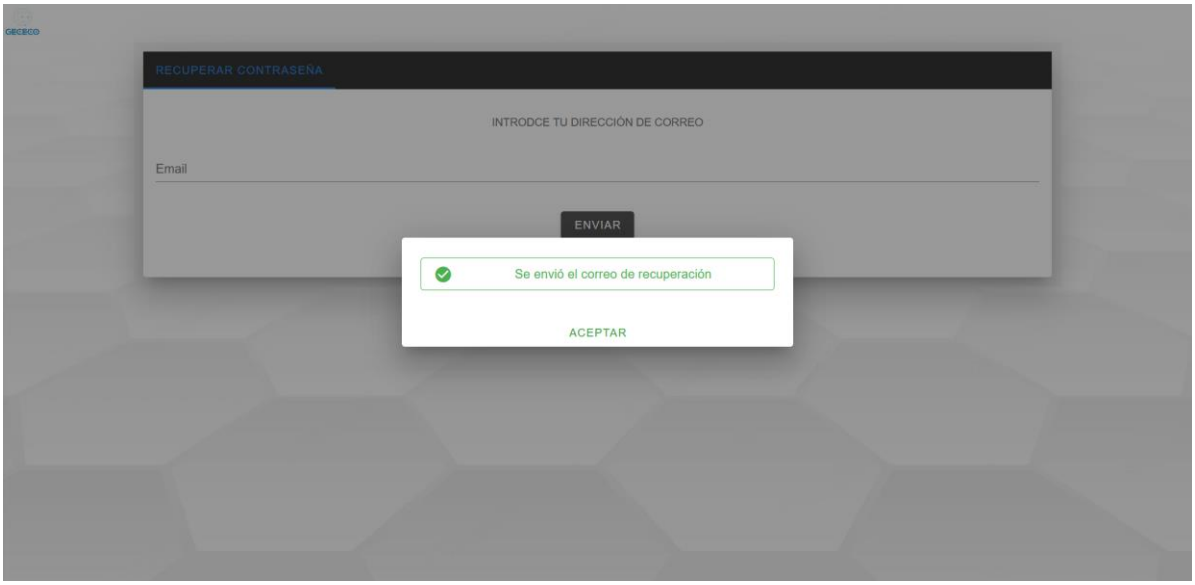

Figura 47 Restablecimiento de contraseña

Acciones que un usuario puede realizar sobre su propio perfil, las cuales serán modificar su foto de perfil y cambiar su contraseña de acceso.

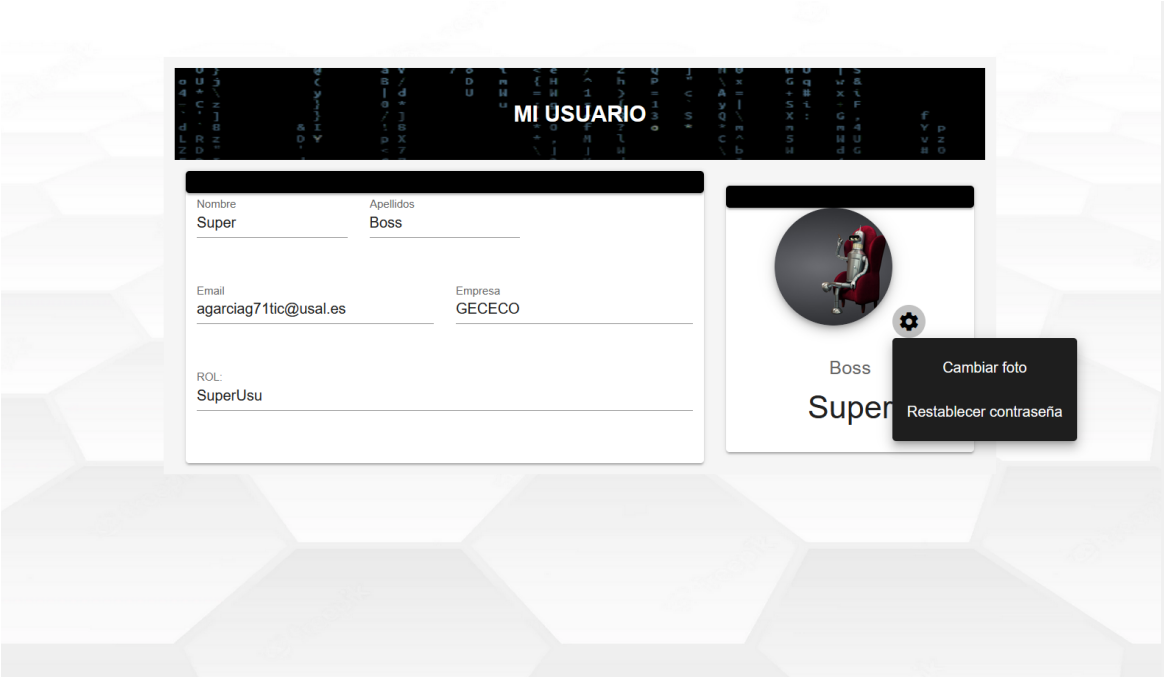

Figura 48 Opciones de un usuario

Esta será la vista mostrada cuando un usuario quiera modificar su contraseña de acceso, para ello deberá proporcionar su anterior y su nueva contraseña, en caso de que la contraseña antigua sea incorrecta o no se proporcionen ambas contraseñas, el sistema arrojará un mensaje de error informando de la causa correspondiente.

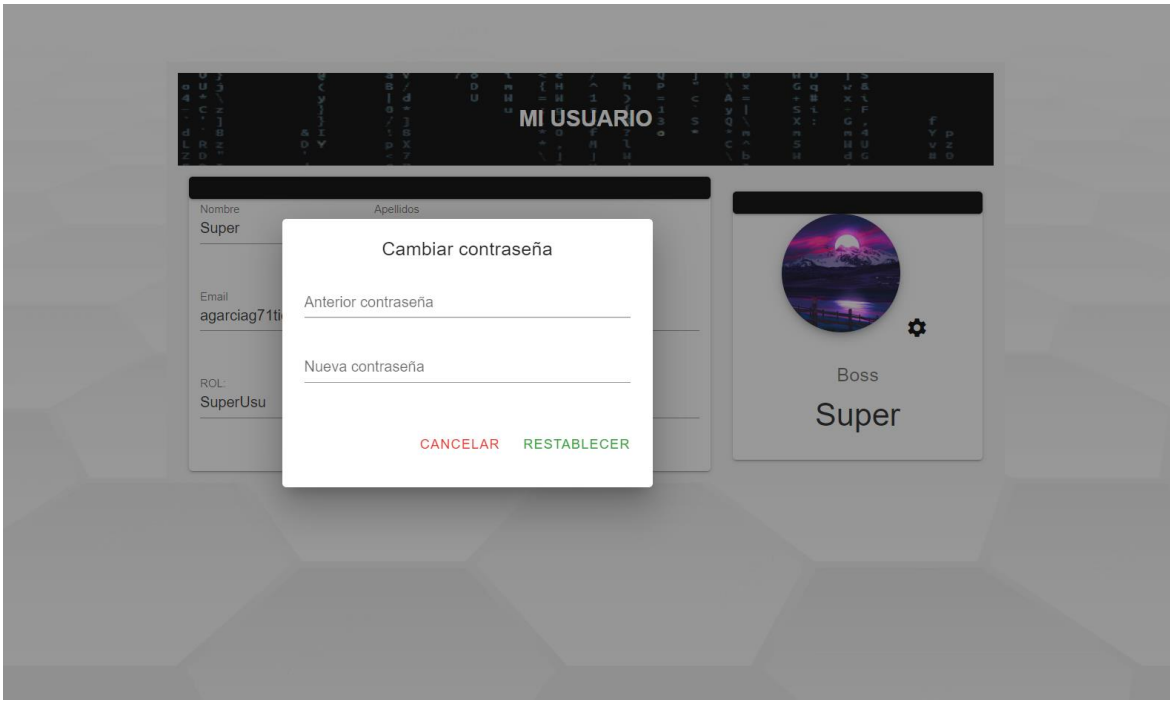

Figura 49 Pantalla de cambio de contraseña

Esta será la ventana de registro de trabajadores deberán proporcionarse todos los datos solicitados, como el nombre apellidos, un correo y el rol que tendrá el nuevo trabajador. En caso de que el nuevo usuario ya exista dentro del sistema o que no se proporcionen todos los datos solicitados el sistema arrojará un mensaje de error informando de ello.

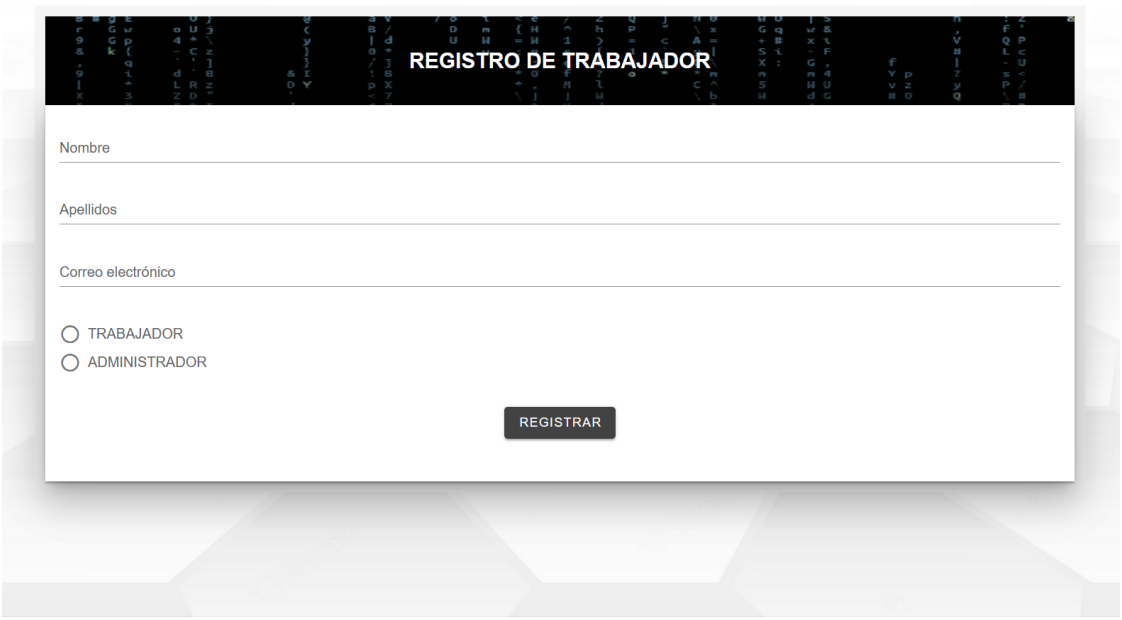

Figura 50 Pantalla de registro de un usuario

6.8.2 GESTIÓN DE EMPRESAS

El sistema permitirá dar de alta nuevas empresas, darlas de baja, así como gestionar los usuarios de dicha empresa, todo esto por parte de los Super Usuarios.

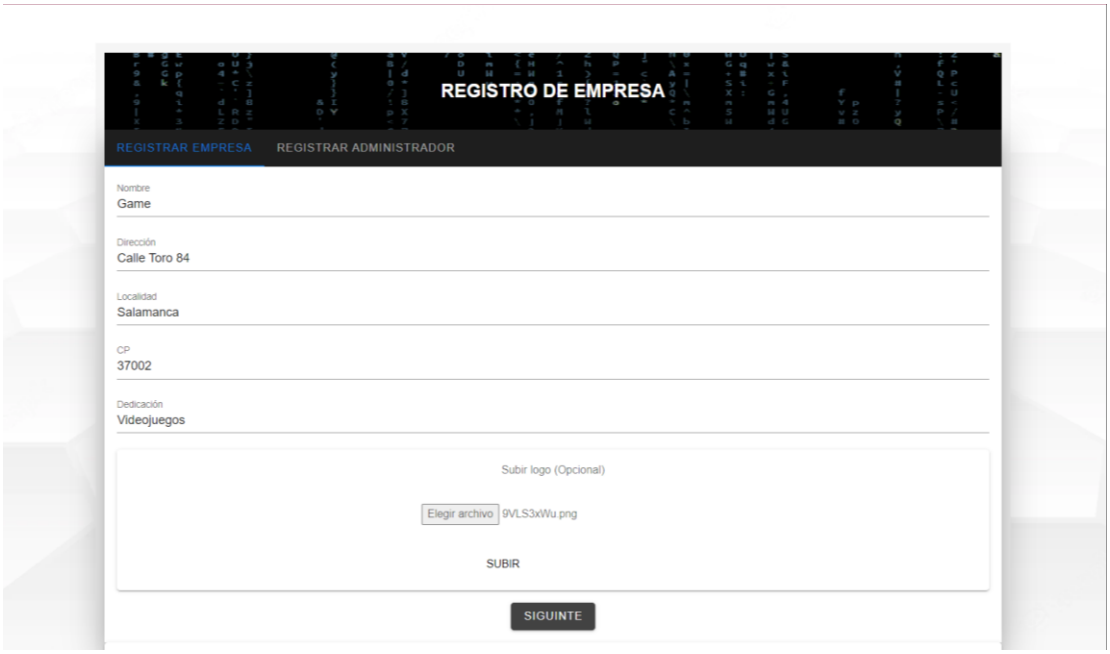

Figura 51 Pantalla de registro de una empresa

Esta será la vista de una empresa registrada dentro del sistema

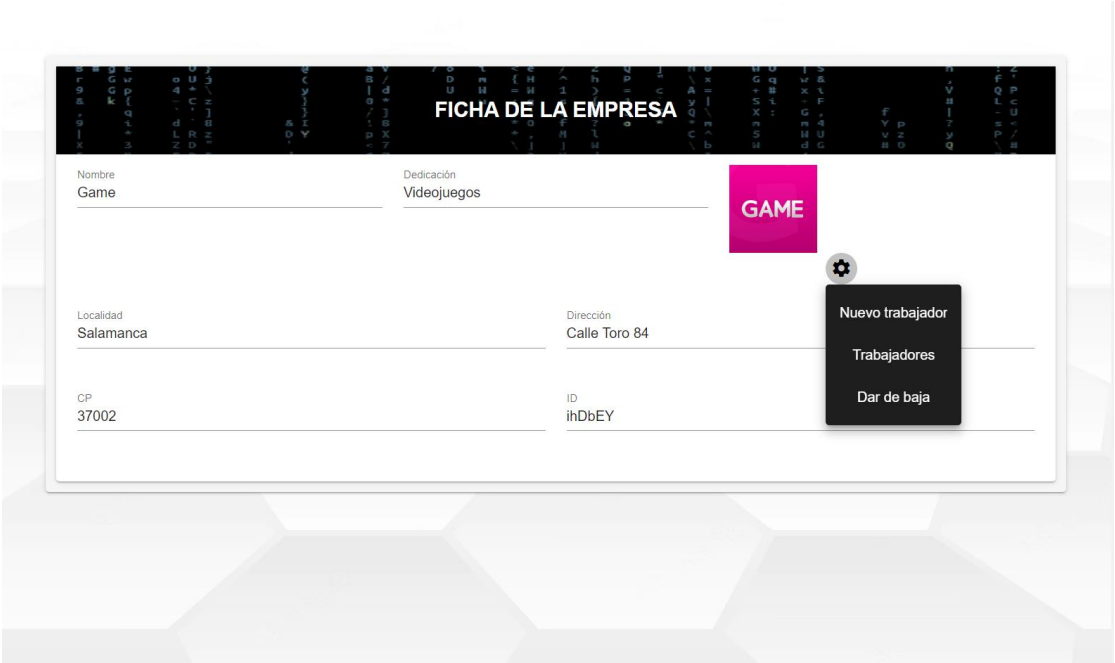

Figura 52 Ficha de una empresa y opciones de gestión

6.8.3 GESTIÓN DE TICKETS

El sistema permite la creación y modificación de tickets de incidencia por los usuarios normales de una empresa para que se produzca una comunicación directa entre trabajadores y administradores de su empresa. Además, los administradores podrán marcar estos tickets como resueltos o eliminarlos en caso de que sea necesario.

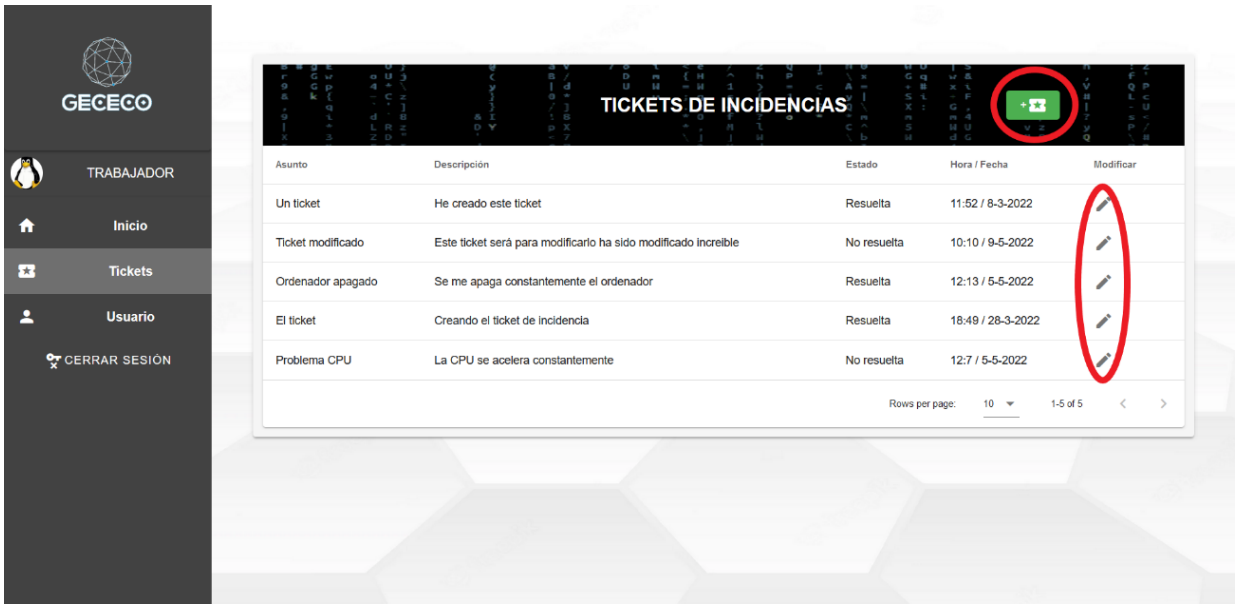

Figura 53 Pantalla de tickets (Usuario normal)

Vista correspondiente a la creación de un nuevo ticket de incidencia.

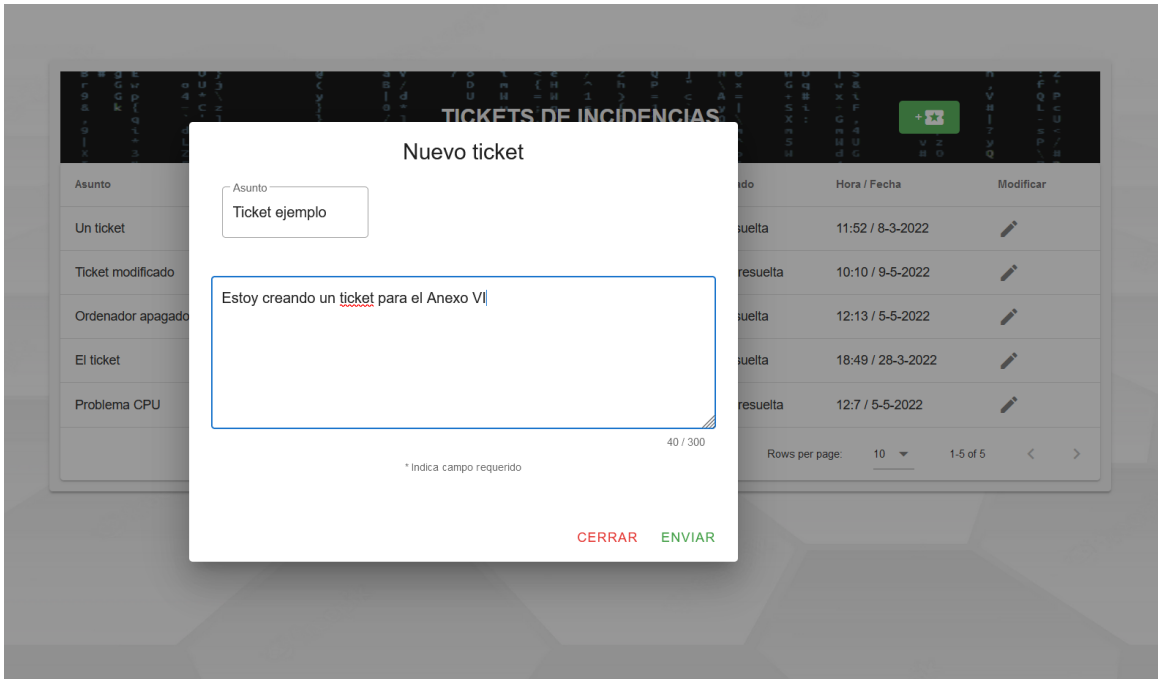

Figura 54 Pantalla de creación de un ticket

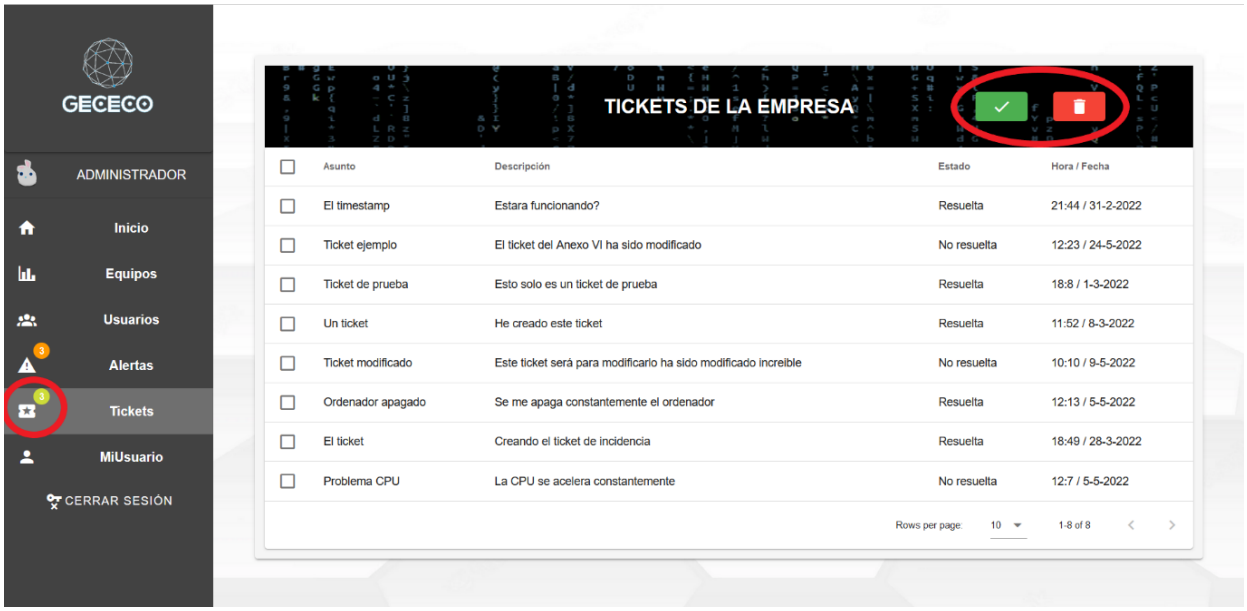

Figura 55 Pantalla de tickets (Administrador)

6.8.4 GESTIÓN DE ALERTAS

El sistema es capaz de crear alertas por parte de los administradores que se dispararán en base a los parámetros establecidos y en función a las lecturas de los equipos de su empresa. Además, se pueden editar los parámetros de estas alertas e incluso eliminarlas si ya no se necesita. Por otra parte, las alertas disparadas generan avisos de alertas que son recibidos por el administrador que creó la alerta que se disparó. Estos avisos pueden marcarse como atendidos en caso de que se resuelva el problema que los provocó.

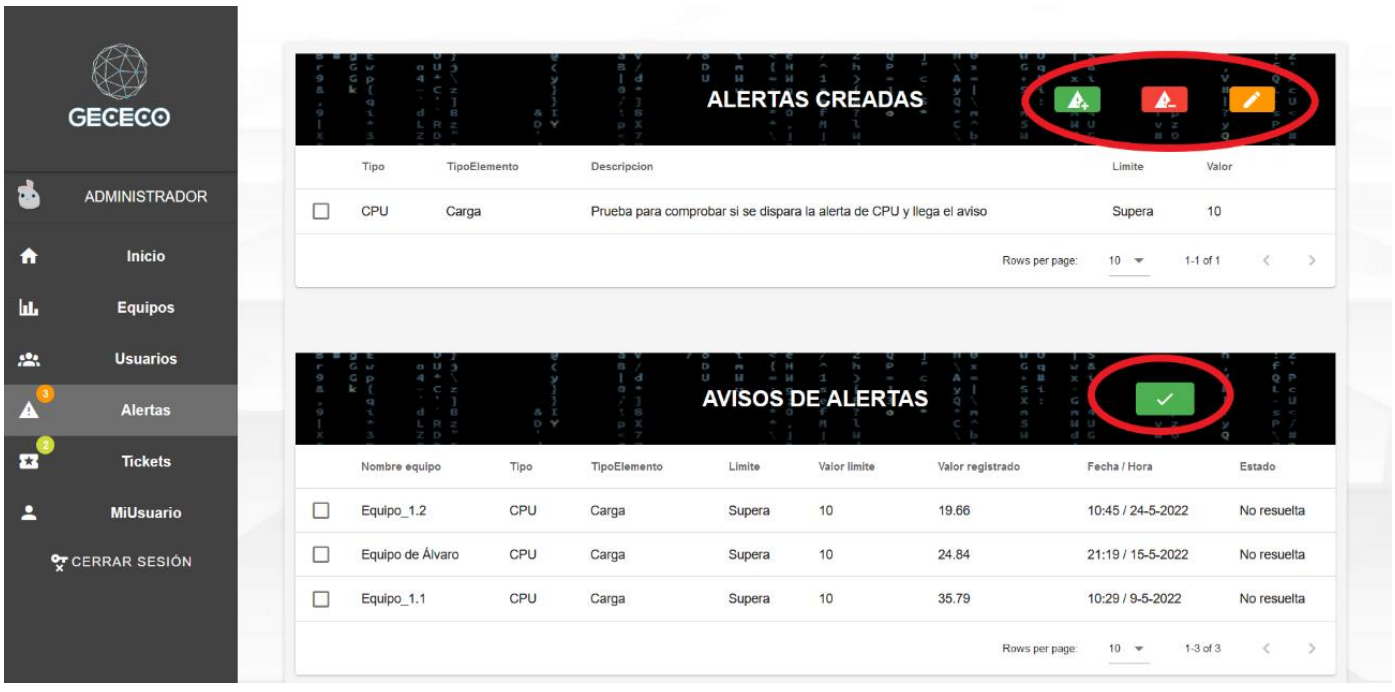

Figura 56 Pantalla de alertas y avisos

Vista de creación de un nuevo tipo de alerta dentro del sistema.

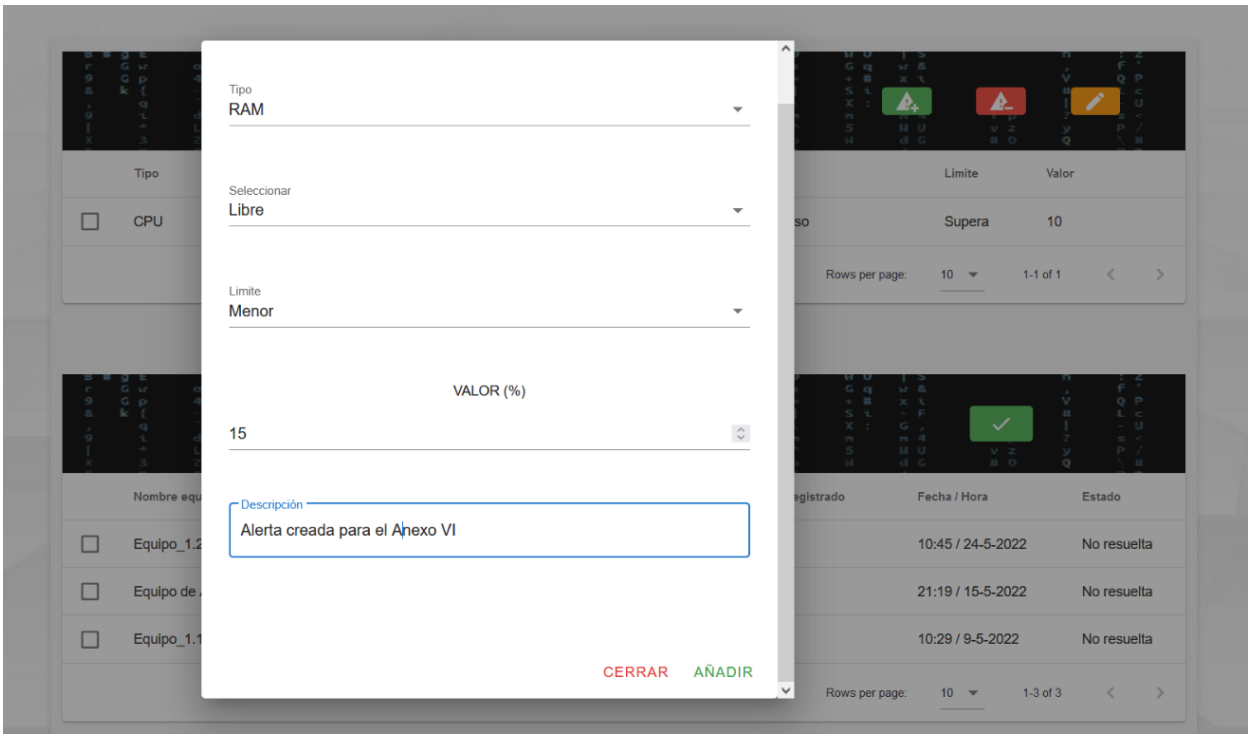

Figura 57 Pantalla de creación de una alerta

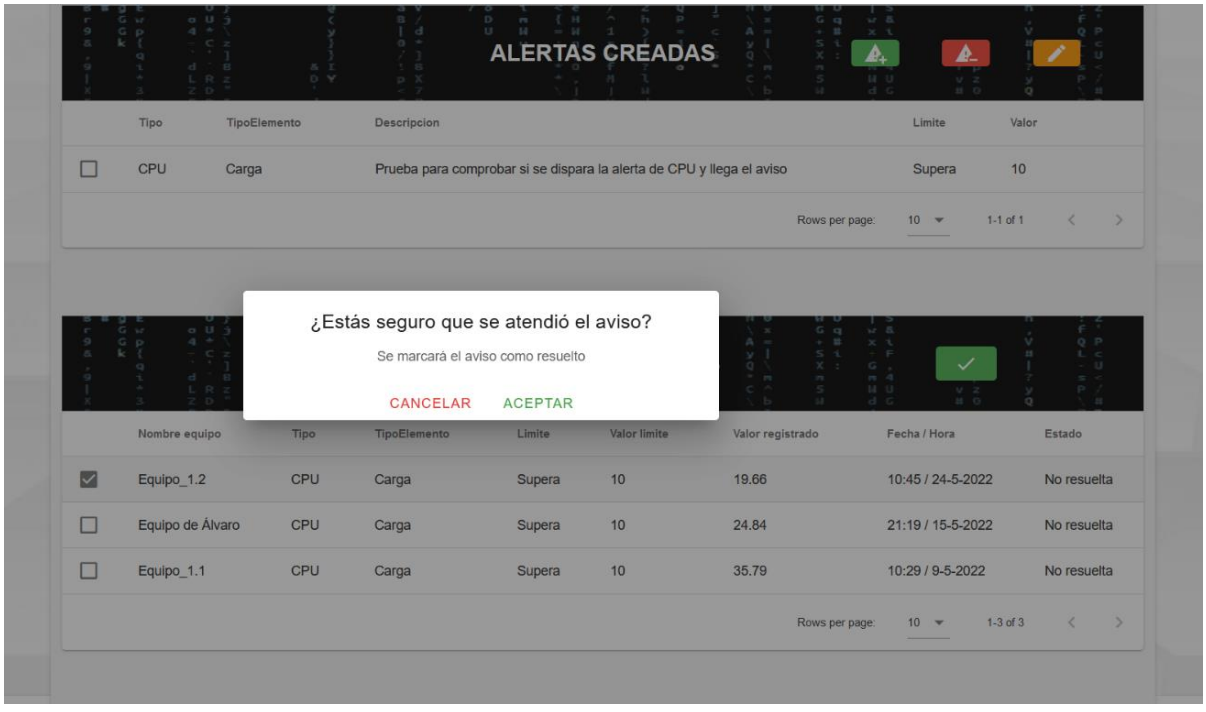

Figura 58 Marcando aviso como atendido

A continuación, se muestra también un ejemplo de la creación de una alerta de servicios caídos, así como una pequeña parte de la tabla de los diferentes servicios para los que podemos crear una alerta.

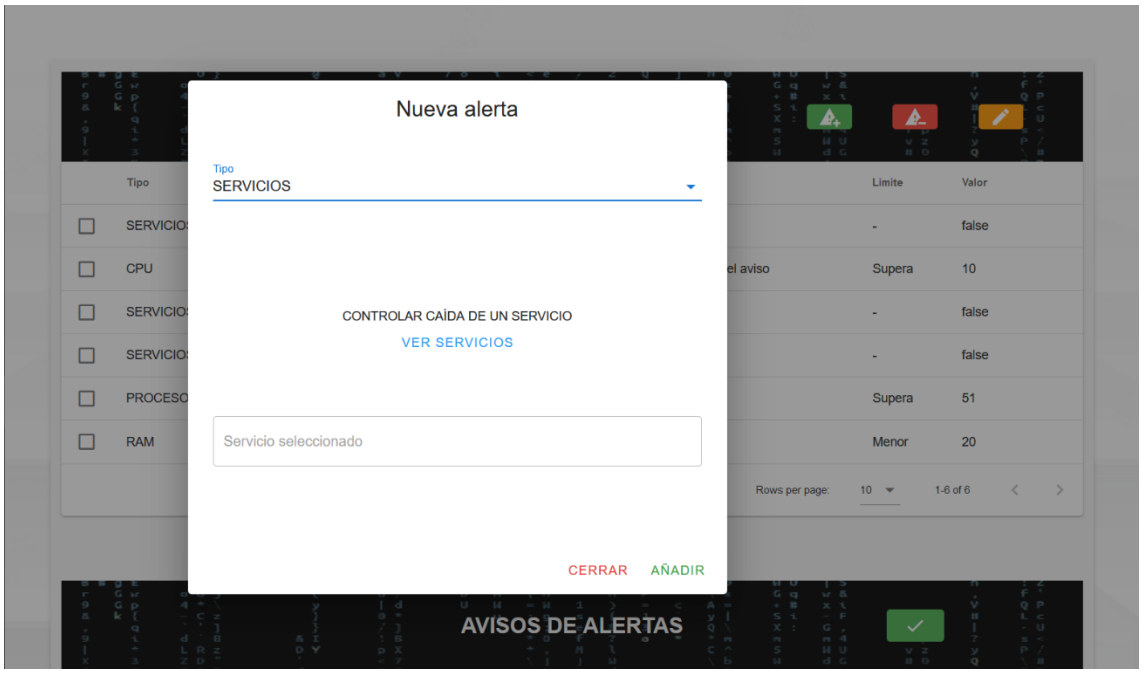

Figura 59 Creación de alertas de servicios

Figura 60 Ejemplo de algunos servicios

6.8.5 GESTIÓN DE EQUIPOS

Se pueden registrar equipos en el sistema para los usuarios registrados en las empresas (también darlos de baja del sistema). Esto permite que dichos usuarios puedan realizar la monitorización y que dicha información sea almacenada y analizada. La información de los equipos puede ser consultada por los administradores de las empresas (evidentemente tan solo los equipos de su propia empresa). Entre la información que puede ser consultada de un equipo estará el estado del hardware, los dispositivos conectados al equipo, los servicios que tiene activos, así como su PID, el número de procesos, los avisos de alertas que ha generado y un histórico con todas las lecturas que se han realizado sobre dicho equipo.

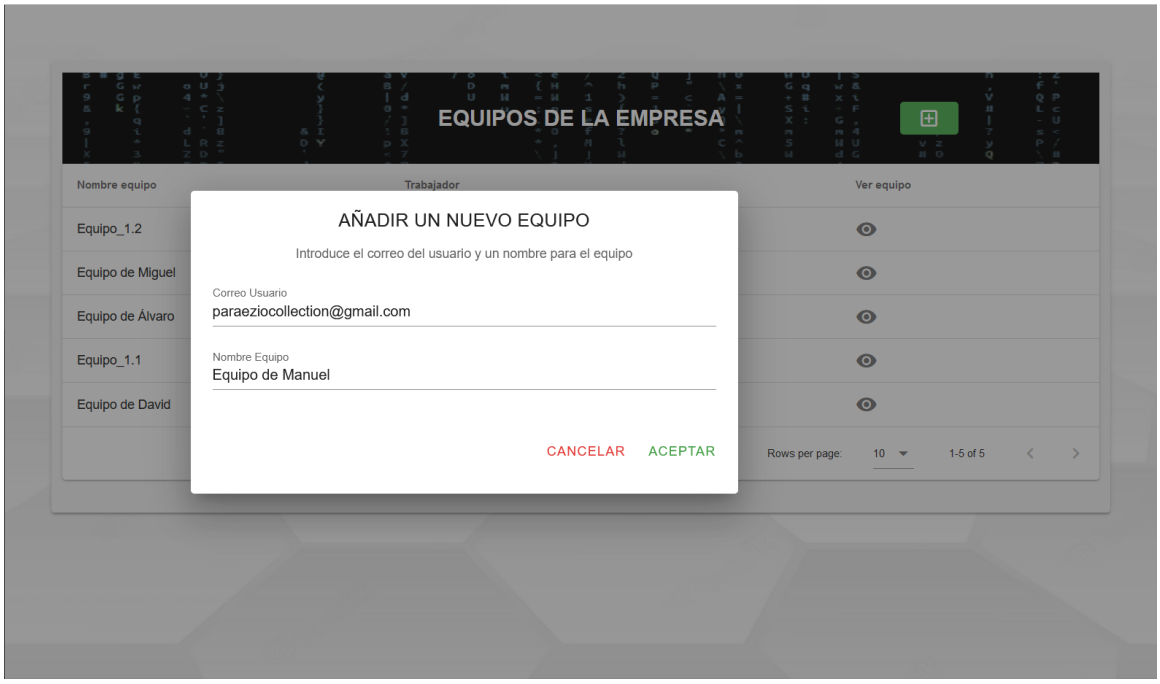

Figura 61 Pantalla añadir nuevo equipo

Diferentes componentes software y hardware que pueden ser consultados en un equipo

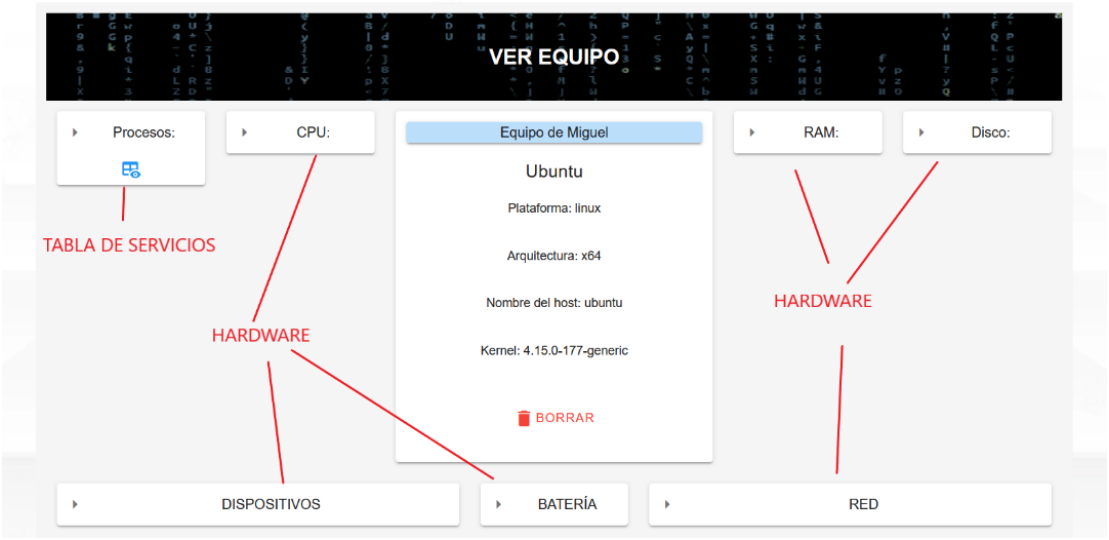

Figura 62 Pantalla de información de un equipo

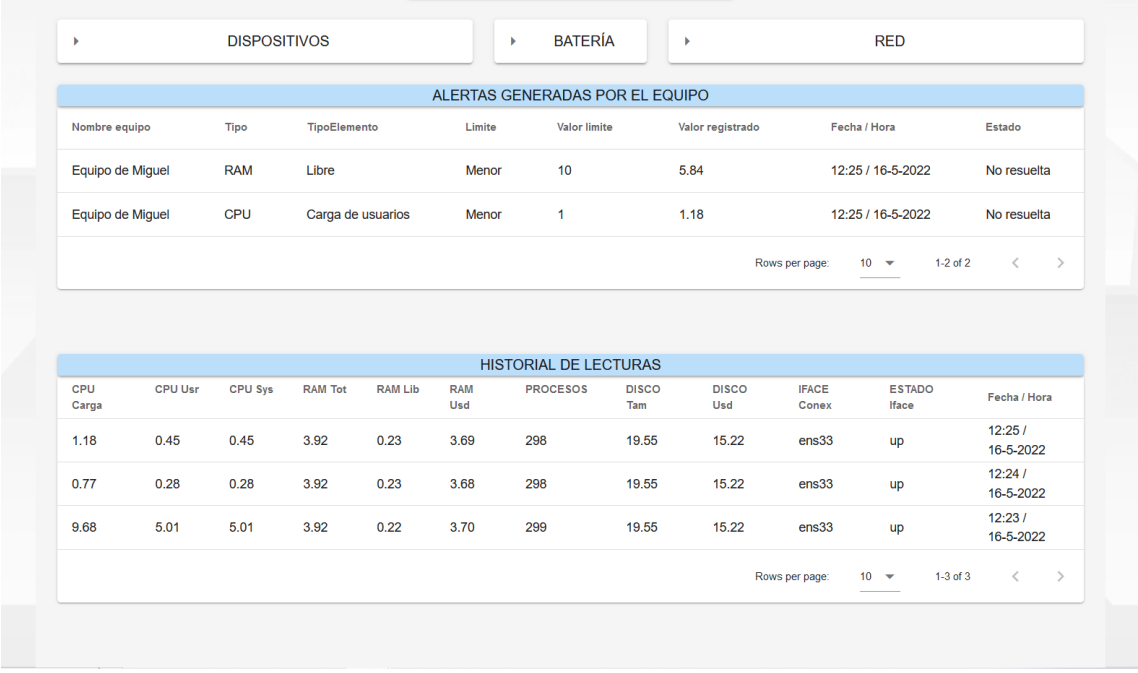

Figura 63 Lecturas y avisos de un equipo

6.8.6 SCRIPT DE MONITORIZACIÓN

Funciona tanto para Linux como para Windows, permite la recolección de datos sobre el equipo que lo ejecuta. Para utilizarlo es necesario que un administrador de nuestra empresa nos haya registrado un equipo dentro del sistema. Posteriormente introduciendo el id de nuestra empresa y nuestro correo electrónico asociado al sistema, comenzará a hacer lecturas periódicas que serán almacenadas en base de datos y analizadas junto con las alertas creadas por los administradores de nuestra empresa.

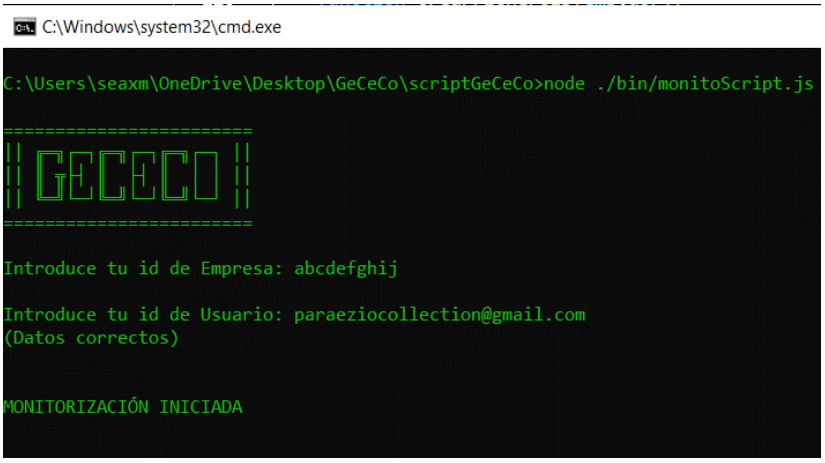

Figura 64 Script de monitorización arrancado en Windows

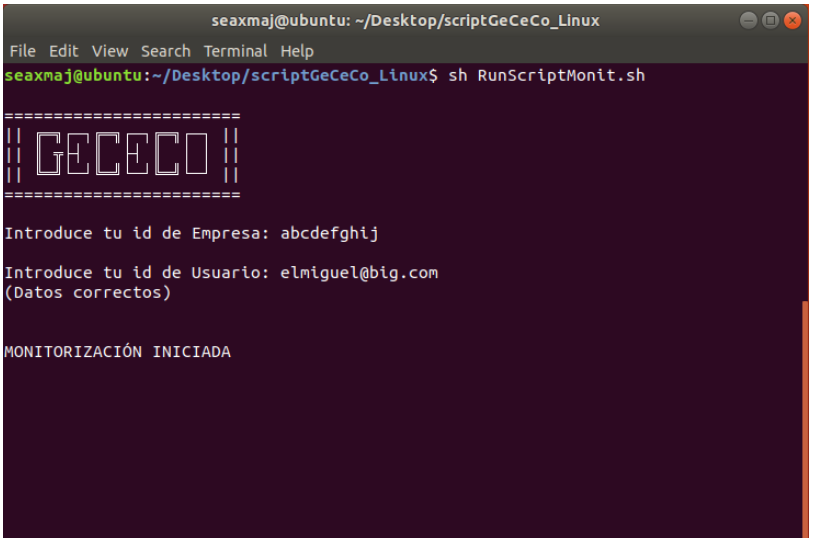

Figura 65 Script de monitorización arrancado en Linux

La ejecución puede ser ejecutada de forma manual por el usuario o configurada por un administrador en los equipos de los trabajadores para que esta sea ejecutada cuando el equipo se arranque (sin embargo previamente será obligatorio que se realice una primera ejecución manual donde se proporcione la información del equipo, para que se cree el fichero de configuración), a continuación, se muestra un pequeño ejemplo para Linux, en el anexo VI se puede encontrar mucho más completo tanto para Linux como para Windows.

Lo primero a realizar será añadir el programa de monitorización al directorio **"/usr/bin/"** para ello sobre el directorio en el que se encuentre la carpeta del programa de monitorización, se ejecutará el siguiente comando: **"cp -r scriptGeCeCo_Linux /usr/bin".** Este comando lo que hará será copiar el programa al directorio /usr/bin. En este directorio se encuentra la mayoría de los programas que se ejecutan como servicios.

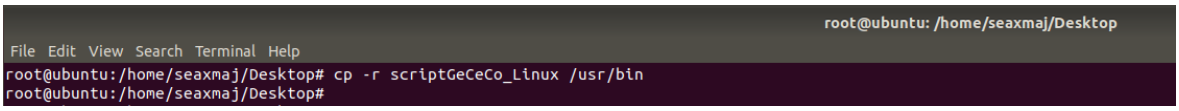

Figura 66 Copia del programa al directorio

Habrá que asegurarse que el script de arranque posee permisos de ejecución, para ello comprobaremos los valores de X mediante un comando **"ls -la"**.

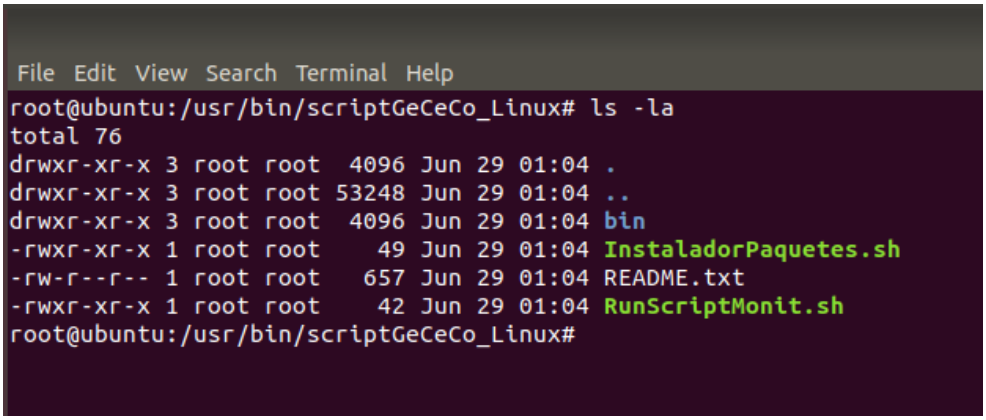

Figura 67 Comprobación de los permisos de ejecución

Habrá que programar a continuación el servicio correspondiente, en este caso lo llamaremos **“GeCeCo_Monitoring.service”.**

Se definirá la opción de ExecStart para indicar que se ejecuta en el arranque del equipo, y se proporciona la ruta al script que debe ser ejecutado, que en este caso es el encargado del arranque del monitorizador.

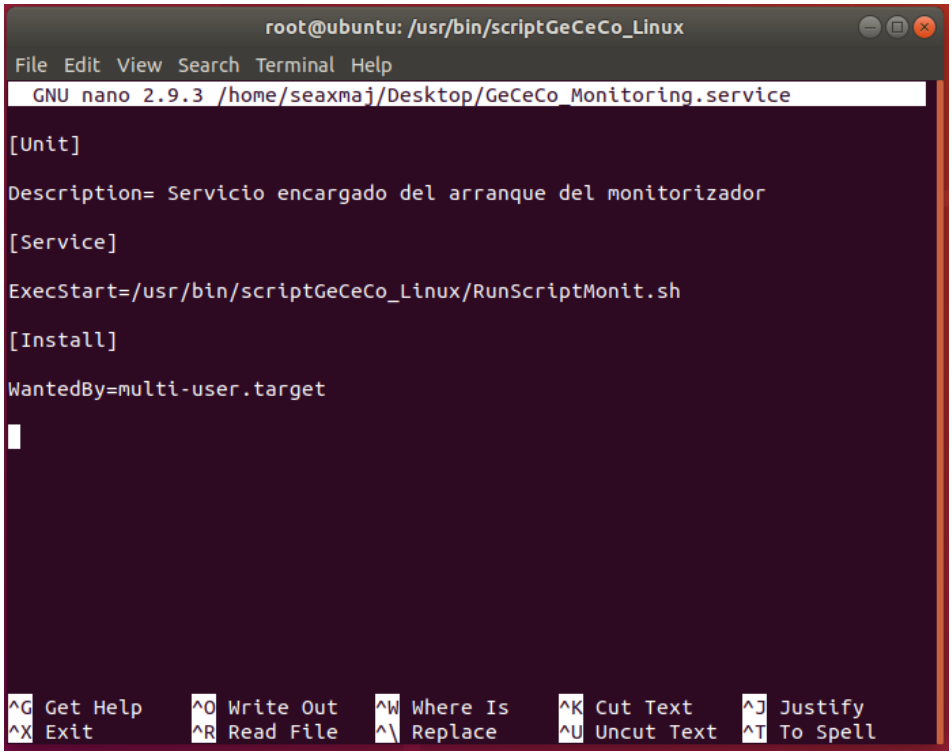

Figura 68 Servicio encargado de ejecutar el script en el arranque

El servicio creado deberá ser introducido en el directorio **“/etc/systemd/system” ”,** para ello sobre el directorio en el que se haya creado el “archivo.service” deberemos ejecutar el siguiente comando: **“cp GeCeCo_Monitoring.service /etc/systemd/system”.** Este comando copiará el servicio creado al directorio donde se encuentran todos los servicios del sistema.

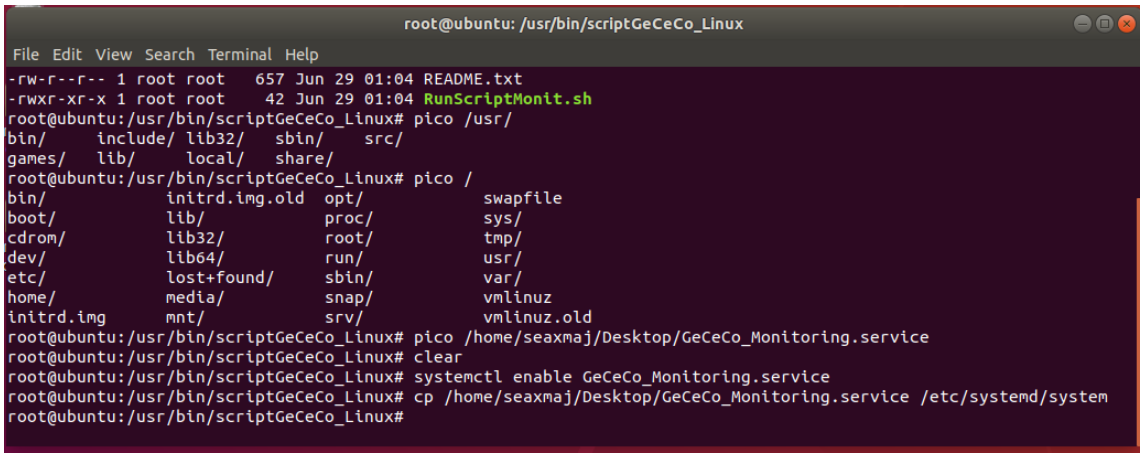

Figura 69 Copia del servicio al directorio system

Por último, habrá que activar el servicio, para ello ejecutaremos el **comando "systemctl enable GeCeCo_Monitoring.service".** Se puede observar en la última orden introducida que el servicio se encuentra ya habilitado. Para comprobar el estado del servicio y si se encuentra habilitado y o encendido o apagado, ejecutaremos el comando **"systemctl status GeCeCo_Monitoring.service".** De esta forma podemos comprobar que se haya hecho correctamente. En el parámetro **"Loaded"** debe aparecer como "**enabled"** y aunque en un primer momento en el parámetro **"Active"** pueda parecer como "**inactive"**, tras un reinicio debe aparecer como "**active".**

Figura 70 Servicio habilitado

7 LIMITACIONES DE LA APLICACIÓN

La principal limitación encontrada en cuanto al sistema en sí podría ser que, aunque es cierto que cumple con lo propuesto, ya que es capaz de monitorizar, enviar alertas y permite a los administradores observar la información de los equipos de su empresa, en ocasiones pueden echarse en falta ciertos aspectos o funcionalidades relevantes durante la administración de equipos, como puede ser un control remoto o el acceso directo a equipos cuando se encuentran conectados mediante una red directa. Referente a esto también, no se puede tener una idea realmente buena de cómo está funcionando la red, ni se pueden realizar tareas de administración sobre la red, como modificar IPs, crear grupos para ciertos equipos entre otras.

Otras de las limitaciones o problemas encontrados es que los módulos, los que se probaron que permiten obtener información de los equipos, en ocasiones arrojaban información incompleta e incluso eran compatibles con sistemas operativos y versiones de estos muy específicos, lo cual dificultó en gran medida obtener suficientes datos como para sentir que se tenía una cantidad de información relevante con la cual poder trabajar.

8 CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

En el siguiente apartado se expondrán las conclusiones obtenidas tras el desarrollo completo del software, así como las líneas de trabajo futuras para mejorar el mismo.

8.1 CONCLUSIONES

Podemos concluir que el software cumple con los requisitos y objetivos propuestos durante la especificación de requisitos. El programa es capaz de realizar exitosamente una gestión de empresas, así como de sus trabajadores. Además, los trabajadores pueden comunicarse directamente con los administradores mediante los tickets de incidencia para recibir asistencia. Por otra parte, se ha logrado hacer una monitorización real de los equipos sustentada por alertas que permiten a los administradores estar al tanto de cuando un equipo está teniendo un funcionamiento indebido.

Además, el monitorizador funciona independientemente de que trabajemos en Linux o en Windows y con independencia también del navegador web que utilicemos para conectarnos a la aplicación web, incluyendo si es desde una Tablet, un ordenador de sobremesa o un ordenador portátil, lo cual aporta gran flexibilidad y compatibilidad a la hora de elegirlo una opción a usar.

Por último, también se cumplió el objetivo personal propuesto antes de comenzar el TFG, que era aprender nuevas tecnologías de desarrollo útiles y en fuerte uso, como lo son Vue y Firebase. Y aprender también a seguir una metodología adecuada a la hora de desarrollar un proyecto software mucho más completo que todos los anteriormente desarrollados durante el grado.

8.2 LINEAS DE TRABAJO FUTURAS

Principalmente en este apartado se abordarán propuestas o soluciones para las limitaciones encontradas y explicadas anteriormente y alguna otra propuesta que me hubiese gustado añadir pero que no hubo tiempo para ello.

- Para solucionar el hecho de que los ordenadores de una empresa no se encuentren interconectados entre sí y como aplicación de apoyo que podría resolver en cierta parte este problema, podría ser el uso de aplicaciones como Hamachi, Radmin VPN u Open VPN, las cuales son fáciles de configurar y nos permiten la creación de redes locales virtuales, conectando los equipos de forma segura a través de una VPN. De esta forma los administradores podrían conectarse directamente a las máquinas de los trabajadores para realizar tareas de administración. Además, se podrían realizar tareas de gestión directas sobre la red, como controlar IPs, crear grupos, entre otras. Además, en un futuro, aunque estas aplicaciones resolverían el problema, podrían implementarse en la

propia aplicación sus funcionalidades, de esta forma obtendríamos un sistema más completo que no precisaría de otras herramientas.

- Para el control de forma remota de los equipos podría implementarse o utilizarse algún plugin o extensión de navegador como Chrome Remote Desktop, el cual nos permite en base a un código compartido manejar de forma remota la máquina de otros usuarios, pudiendo acceder de forma directa a los ordenadores. Otra opción sería la instalación de un software del estilo a TeamViewer, aunque de esta forma se rompería un poco la idea de que precisamente para utilizar este software no es necesario instalar nada.

- Para resolver los problemas de compatibilidad de los módulos de monitorización y para hacer que sea más completo podría ser interesante el desarrollo de una serie de bibliotecas en JavaScript centrada a la recopilación de información de los equipos en Windows, en base a toda la información que se desee obtener y de la misma forma hacerlo para Linux. De esta forma nos podríamos asegurar de que toda información que queramos que pueda ser recogida, se haga correctamente.

- En un futuro también, podría ser interesante crear canales de comunicación por voz para la aplicación, como lo hace por ejemplo “Discord Web”, de esta forma, los trabajadores podrían contactar con los administradores realizando una llamada, potenciando de esta forma la comunicación.

9 REFERENCIAS

Ali, J. (2016). *Mastering PHP Design Patterns.* Birmingham, UK: Packt Publishing Ltd.

Arimetrics. (s.f.). Obtenido de https://www.arimetrics.com/glosario-digital/backend

DiagramasUML. (s.f.). Obtenido de https://diagramasuml.com/despliegue/

EcuRed. (s.f.). Obtenido de https://www.ecured.cu/Visual_Paradigm

EcuRed. (s.f.). Obtenido de https://www.ecured.cu/Modelo_Vista_Vista_Modelo

esan business. (s.f.). Obtenido de https://www.esan.edu.pe/conexion-esan/microsoft-project-su-aplicacion-en-la-gestion-de-proyectos

García Peñalvo, F. J., García Holgado, A., & Vázquez Ingelmo, A. (s.f.). *Proceso unificado.* Obtenido de https://repositorio.grial.eu/bitstream/grial/2470/1/7.%20PU-2021.pdf

García Peñalvo, F., & García Holgado, A. (s.f.). *Ingeniería de requisistos.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1143/1/IS_I%20Tema%204%20-%20Ingenieria%20de%20Requisitos.pdf

García Peñalvo, F., García Holgado, A., & Vázquez Ingelmo, A. (s.f.). *Flujos de trabajo del proceso unificado.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1945/1/IS_I%20Tema%206%20-%20Flujos%20de%20trabajo%20del%20Proceso%20Unificado.pdf

García Peñalvo, F., García Holgado, A., & Vázquez Ingelmo, A. (s.f.). *Fundamentos de la vista de casos de uso.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1950/1/UML%20-%20Casos%20de%20uso-2020.pdf

García Peñalvo, F., Moreno García , M., García Izquierdo, A., & Vázquez Ingelmo, A. (s.f.). *UML. Unified Modeling Language.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1949/1/IS_I%20Tema%208%20-%20UML.pdf

García, M. N. (s.f.). Obtenido de https://studium.usal.es/pluginfile.php/134783/mod_resource/content/1/GP-GII_Primera_practica.pdf

García, M. N. (s.f.). *Estimación del esfuerzo.* Obtenido de https://studium.usal.es/pluginfile.php/134783/mod_resource/content/1/GP-GII_Primera_practica.pdf

ITDO. (s.f.). Obtenido de https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/

json.org. (s.f.). Obtenido de https://www.json.org/json-es.html

mdn web docs. (s.f.). Obtenido de https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics

Microsoft docs. (s.f.). Obtenido de https://docs.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data

Red Hat. (s.f.). Obtenido de https://www.redhat.com/es/topics/api/what-is-a-rest-api

rockcontent. (s.f.). Obtenido de https://rockcontent.com/es/blog/framework/

Universidad César Vallejo. (s.f.). *Proceso unificado de desarrollo software.* Obtenido de https://fdocuments.es/document/1-proceso-unificado-de-desarrollo-de-software-2-eap-ingenieria-de-sistemas.html?page=4

Universidad de Sevilla. (s.f.). *Departamento de lenguajes y sistemas informáticos*. Obtenido de http://www.lsi.us.es/descargas/descarga_programas.php?id=3

VUEJS. (s.f.). Obtenido de https://es.vuejs.org/v2/guide/