ULPGC
Universidad de
Las Palmas de
Gran Canaria

Escuela de
Ingeniería
Informática

Control del horario de trabajo en la empresa mediante el uso de servicios de AWS

Trabajo Fin de Grado
Grado en Ingeniería Informática

CARLOS ESTEBAN LEÓN DÍAZ

Tutores
Javier Sánchez Pérez
Carmelo Cuenca Hernández

Las Palmas de Gran Canaria
Septiembre 2022

Resumen

Este proyecto implementa un sistema informático para que los empleados de una empresa puedan registrar sus fichajes de jornada a través de una aplicación móvil mediante un sistema de reconocimiento facial. Asimismo, el proyecto también implementa una aplicación web para que los administradores de las empresas puedan ver los registros de jornada de sus trabajadores y elaborar informes en los que podrán ver la fecha y hora de los fichajes, así como la localización en que se realizaron.

Abstract

This project develops a software so that the employees of a company can register their workdays through a mobile application using a facial recognition system. Likewise, the project also implements a web application so that company administrators can see the records of their workers' working hours and elaborate reports in which they can see the clock times, as well as the location where they were made.

Agradecimientos

Agradezco a mi familia por el apoyo durante estos años,

A los compañeros y profesores con los que me he encontrado en el camino,

Y a mis tutores, por ayudarme con el proyecto y darme la idea de este en un momento en el que tenía muchas dudas sobre cómo enfocar este trabajo.

ÍNDICE

Índice de figuras

Índice de tablas

1. Introducción

El 8 de marzo del año 2019, el gobierno de España estableció vía Real Decreto-ley una serie de medidas destinadas a reforzar la protección laboral y a disminuir la precariedad. Entre estas medidas destaca el control horario en la jornada laboral.

La motivación de este real decreto nace con el fin de eliminar los abusos en la jornada laboral de los trabajadores por parte de algunas empresas (especialmente en el sector de la hostelería), incumpliendo los horarios de estos e impagando horas extras. Desde su implantación en marzo del 2019, ya son muchas las empresas que, tras inspecciones de trabajo, han sido sancionadas con multas que oscilan desde los 60 euros hasta los 187000 euros, según el grado de severidad de las infracciones. Destaca especialmente el fallo de la Audiencia Nacional contra el banco BBVA (Banco Bilbao Vizcaya Argentaria), en el que el banco requería la autorización de un superior para la confirmación del fichaje de los empleados. [1]

En relación con esta obligatoriedad para todas las empresas de registrar la jornada laboral de sus trabajadores, se hace necesario de un software que se ajuste a la legislación vigente y que sea capaz de llevar a cabo esta gestión, así como de elaborar informes con el histórico de fichajes de los empleados en caso de que estos pudiesen ser requeridos por el ministerio de trabajo.

Este proyecto viene a satisfacer esta necesidad de las empresas de controlar el horario de la jornada laboral de sus empleados.

Para ello, se ha desarrollado un software que permite a las empresas registrarse a sí mismas y a sus empleados, pudiendo asignarles roles y lugares de trabajo, con el objetivo de que estos empleados fichen en la entrada y la salida de su lugar de trabajo asignado, enviando su ubicación y una foto de ellos mismos para que, mediante un proceso de reconocimiento facial, certifiquen su fichaje de entrada o salida del trabajo. Este software ha estado sustentado bajo una arquitectura desplegada en la nube de Amazon Web Services, haciendo uso de sus múltiples servicios (computación, machine learning, hosting, etc.) para lograr los objetivos marcados.

La metodología seguida en el desarrollo de este software es el de la metología ágil SCRUM, en la que se va agregando valor al producto de manera incremental con entregas continuas del software. Al inicio de este desarrollo, se determinaron una serie de funcionalidades requeridas en el sistema y documentadas en historias de usuario, que se han ido implementando y entregando periódicamente hasta finalizar el proyecto.

Finalmente, se logró el objetivo principal de crear un software accesible y eficaz capaz de registrar las jornadas de los empleados de una empresa mediante un proceso de reconocimiento facial, permitiendo a las empresas generar informes de las jornadas de sus trabajadores.

1.1 Objetivos

El objetivo fundamental fijado para este proyecto es el de desarrollar un software capaz de gestionar de manera sencilla y eficaz el fichaje de los empleados de una empresa mediante reconocimiento facial, usando servicios de la nube de Amazon (AWS) como soporte para nuestro software.

Amazon Web Services es una colección de servicios de computación en la nube con una amplia gama de servicios de todo tipo, computacionales, almacenamiento, análisis, seguridad, etc. Entre todas las ventajas que podemos asociar a AWS, destacan las siguientes:

Escalabilidad: AWS nos permite modificar en cualquier momento nuestros recursos en uso, ajustándose a nuestras necesidades de uso o presupuesto. Por ejemplo, podríamos crear una instancia de una máquina virtual (EC2-Elastic Compute Cloud), con una memoria RAM inicial de 4GB, y posteriormente, aumentar esta a 8GB.
Facilidad de uso: AWS cuenta con una consola de administración desde la que poder gestionar todos y cada uno de los servicios que ofrecen, de tal modo que desde un mismo portal web podemos gestionar todos los recursos de nuestra empresa.
Efectividad de costes: pago por uso. Con AWS, se paga por el número de horas empleadas y por los servicios usados.

Es por todo esto que se planteó como objetivo implementar un sistema real usando fundamentalmente recursos de la nube. Esto incluye todos los recursos, tanto hardware como software, que constituyen un sistema informático, entre otros:

- Servidor web donde alojar la aplicación web.
- Servidor de base de datos.
- Servicios web que enlazan la aplicación web con la base de datos.
- Autenticación.
- Almacenamiento.

Con este plan en mente de usar la nube de Amazon como pilar esencial del proyecto, y a instancia de mis tutores, que me dieron la idea de negocio, nace este proyecto basado en la gestión horaria de la jornada laboral.

1.2 Aportaciones

En el ámbito empresarial, son muchas las aplicaciones que gestionan de uno u otro modo el control horario de los trabajadores de una empresa. Sin embargo, una de las características observadas en el análisis del estado del arte llevado a cabo en este informe es que muchas de estas aplicaciones son excesivamente complejas, ofreciendo un elevadísimo número de características que aportan muy poco y que sobrecargan en exceso la atención del usuario. Esto supone un problema si tenemos en cuenta de que muchos de los trabajadores que usan estas aplicaciones no tienen un gran bagaje en el uso de la tecnología, por lo que sobrecargar aplicaciones con menús muy extensos y características ambiguas crean malestar en los usuarios y dificultan la tarea fundamental, que es fichar las entradas y salidas de jornada laboral.

Por ello, se ha desarrollado un software con una interfaz muy sencilla y con unas funcionalidades muy definidas, de modo que la gran mayoría de estas recaen sobre los administradores de las empresas que se registren en nuestro sistema, mientras que los trabajadores se pueden limitar a realizar sus fichajes o a ver su calendario.

Este software está dividido en dos aplicaciones desarrolladas para este proyecto:

ClockIn: Aplicación web destinada a los administradores de la empresa, que serán los encargados de registrar la empresa y gestionar a sus empleados, roles y lugares de trabajo, así como de generar informes.

ClockInConnect: Aplicación móvil destinada a los empleados, para que estos registren sus fichajes diarios mediante reconocimiento facial (sacándose una foto en el momento de fichar), comprueben sus fichajes pasados o vean sus períodos vacacionales asignados.

1.3 Organización del documento

La organización de los apartados de este informe se establece del siguiente modo:

Capítulo 2: En este apartado se analiza el estado actual del mercado en relación con este proyecto, comentando brevemente las características principales de estas aplicaciones.

Capítulo 3: Indicamos las herramientas utilizadas en el proyecto, definiéndolas brevemente y analizando su razón de uso en el proyecto.

Capítulo 4: Se describen la metodología y planificación empleadas en el desarrollo del proyecto, indicando las variaciones con respecto a la planificación inicial especificada en el documento TFT01.

Capítulo 5: En este capítulo se explicará el desarrollo llevado a cabo en este proyecto, separando en secciones los elementos principales de este, esto es, análisis, diseño, implementación y despliegue.

Capítulo 6: Se exponen las conclusiones obtenidas del proyecto y las posibles extensiones aplicables a este de cara a un trabajo futuro.

2. Estado del arte

En un mundo actual tan volcado a la digitalización en el ámbito empresarial, es frecuente encontrar un elevado número de aplicaciones, tanto web como nativas, enfocadas a tareas de gestión comunes a la gran mayoría de empresas. Una de estas tareas comunes a todas ellas, especialmente desde la entrada en vigor del Real Decreto-ley 8/2019, es la del control horario de la jornada laboral de los trabajadores. [2]

Existen un gran número de aplicaciones, fundamentalmente móviles, para el fichaje de los empleados en su jornada laboral. Algunas de ellas utilizan sistemas biométricos como un lector de huellas o una cámara de reconocimiento facial para registrar el fichaje, mientras que otras simplemente registran el momento del fichaje al hacer click en un botón.

A continuación, se mencionan algunas de las aplicaciones que podemos encontrar en el mercado destinadas a este propósito, indicando sus principales características.

Intratime

Figura 2.1 Aplicación móvil Intratime

Características

- Registra de forma automática las horas de trabajo de todos los empleados.
- Geolocaliza los fichajes y las rutas de los empleados mediante el mapa.
- Extrae informes personalizados, permite descargarlos y enviarlos por correo.

Precio	Características principales incluidas
Plan menual “basic”: 0,99€ / usuario	Control horario multidispositivo
	Calendarios
	Vacaciones
	Informes
Plan mensual “pro”: 1,95€ / usuario	Control horario multidispositivo
	Fichajes por cliente y proyecto

	Control de gastos
Plan mensual “Checkpoint POS”: 3,95€ / usuario	Control horario con visualización de ruta
	Asignación de tiempos a clientes y proyectos

Tabla 2.1 Intratime

Beebole

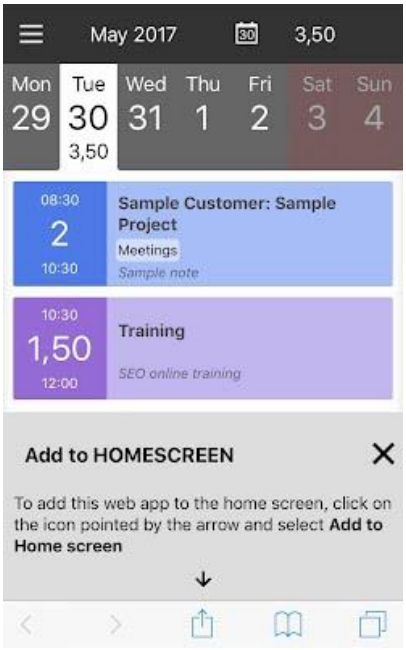

Figura 2.2 Aplicación móvil Beebole

Características

- Control de horas para clientes, equipos, proyectos y tareas.
- Informes configurables y exportables. Integrados con Google Workspace y Microsoft 365.
- Cumple con la normativa de registro de jornada a nivel nacional y con la RGPD.

Precio	Características principales incluidas
Único plan mensual: 6,99€ / usuario	Registro de horas en clientes, proyectos, y tareas
	Informes detallados y personalizados
	Soporte por chat/email
	Registro de jornada laboral fácil y rápido

Tabla 2.2 Beebole

BixPe Control Horario

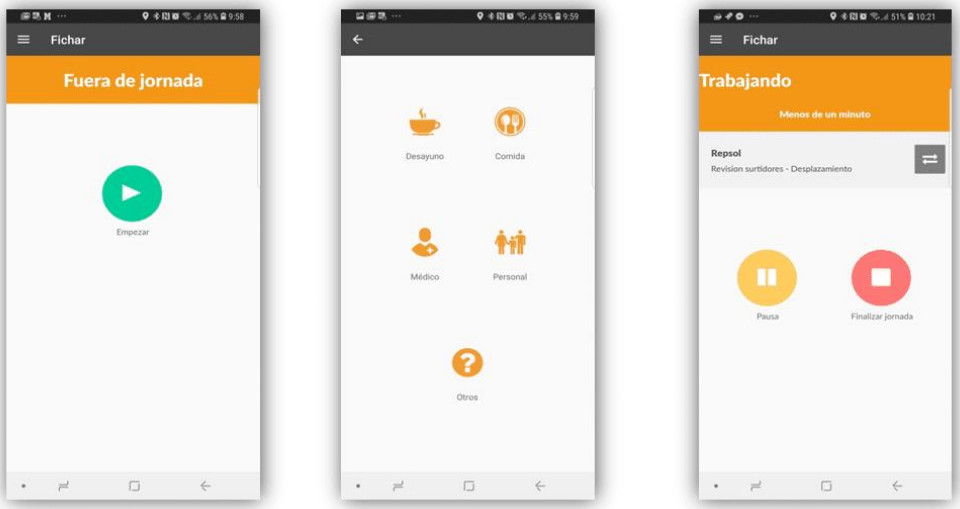

Figura 2.3 Aplicación móvil Bixpe Control Horario

Características

- Diferentes modalidades de fichaje:
  - Centralizado: tablet en el lugar de trabajo con acceso vía PIN del empleado.
  - App móvil: tomando posición GPS y fotografía del empleado.
  - Desde cualquier ordenador a través de la web de BixPe.
- Elaboración de informes.
- Histórico de fichajes.
- Gestión de vacaciones.
- Envío de notificaciones.

Precio	Características principales incluidas
Plan menual “free”: 0,00€ / usuario	Registro de jornada
	Informes básicos
	Histórico de fichajes de 1 mes
	Sin límite de usuarios
Plan mensual “*premium*”: 2€ / usuario	Registro de jornada y de pausas durante la misma, indicando el motivo
	Informes avanzados
	Histórico de fichajes de 4 años
	Gestión de vacaciones
	Gestión de horarios para el control de horas extra

Tabla 2.3 Bixpe

Cuadro comparativo

Funcionalidades/ Aplicaciones	intratime	Beebole	Bixpe Control Horario
Geolocalización	Sí	No	Sí

Sistemas biométricos para el fichaje	No	No	No
Informes	Sí	Sí	Sí
Histórico de fichajes	Sí	Sí	Sí
Control de horas para proyectos	Sí	Sí	No
Gestión de pausas	No	No	Sí
Gestión de vacaciones	Sí	No	Sí

Tabla 2.4. Comparativa aplicaciones similares (intratime, beebole y bixpe control horario)

3. Recursos utilizados

En este apartado se enumeran todos los recursos utilizados (AWS, lenguajes de programación, herramientas, etc.) durante el desarrollo de este proyecto, definiéndolos brevemente e indicando su razón de uso en nuestro proyecto.

3.1 Recursos en la nube (AWS)

Amazon Rekognition

Figura 3.1 Logo Amazon Rekognition

Servicio que se engloba dentro de los servicios de "Machine learning" de AWS para el reconocimiento de imágenes mediante el uso de redes neuronales capaces de reconocer objetos, rostros, textos, etc.

En este proyecto se usa para el principal caso de uso de nuestro sistema: el fichaje de los trabajadores a través de reconocimiento facial, tomando una foto de este y comparándola con la imagen de su rostro asociada a su usuario en el sistema.

API Gateway

Figura 3.2 Logo Amazon API Gateway

Amazon API Gateway es un servicio de AWS para la creación, la publicación, el mantenimiento, el monitoreo y la protección de las API REST, HTTP y de WebSocket a cualquier escala.

Es un elemento fundamental en nuestro sistema "serverless", pues ejerce de enlace entre nuestras aplicaciones y la base de datos.

Amazon Lambda

Figura 3.3 Logo Amazon Lambda

AWS Lambda es un servicio informático sin servidor y basado en eventos que permite ejecutar código para prácticamente cualquier tipo de aplicación o servicio backend sin necesidad de aprovisionar o administrar servidores. [3]

Representa todas las funciones de backend en nuestro sistema gracias a su integración con API Gateway, implementando el código de nuestras APIs para las conexiones con la base de datos así como el acceso a otros recursos de AWS como Amazon S3 y Amazon Rekognition, entre otros.

Amazon S3

Figura 3.4 Logo Amazon S3

Amazon S3 es un servicio de AWS para el almacenamiento de objetos de todo tipo (imágenes, vídeos, documentos, etc.) en la nube. Estos objetos se almacenan en lo que se conoce como “buckets” de S3, que funcionan a modo de directorio, en los que cada objeto se almacena identificado con una clave o “key”.

Es el servicio de almacenamiento coral de nuestro sistema. Se emplea para el almacenamiento de las imágenes de los rostros de los empleados, así como los logos de empresas registradas e imágenes de sus lugares de trabajo. Adicionalmente, empleamos este servicio para tareas de mantenimiento como el almacenamiento de archivos de backup de nuestra base de datos.

Amazon RDS

Figura 3.5 Logo Amazon RDS

Amazon Relational Database Service (Amazon RDS) es una colección de servicios administrados que facilita las tareas de configuración, operación y escalado de una base de datos en la nube. [4]

Con AWS RDS gestionamos la base de datos de nuestras aplicaciones a través del sistema de gestión de base de datos relacional SQL Server.

Amazon Elastic Beanstalk

Figura 3.6 Logo Amazon Elastic Beanstalk

Amazon Elastic Beanstalk es una plataforma dentro de AWS que se utiliza para implementar y escalar aplicaciones web. En términos simples, esta plataforma como servicio (PaaS) toma el código de una aplicación y lo implementa mientras aprovisiona la arquitectura de soporte y los recursos informáticos necesarios para que se ejecute el código. [5]

La empleamos en nuestro sistema para la API de generación de informes (empleados, fichajes, etc.) en la web.

Amazon Amplify

Figura 3.7 Logo Amazon Amplify

Amplify es un servicio de AWS para la gestión de aplicaciones web y móviles y que cuenta con numerosas herramientas de apoyo tanto frontend como backend. Destacan especialmente las herramientas de hosting de la aplicación, así como la posibilidad de desplegar las aplicaciones mediante integración continua enlazando nuestra aplicación con un repositorio git.

Este servicio lo usamos para el hosting de la aplicación web de nuestro sistema, enlazando el repositorio git de la misma para el despliegue continuo de la aplicación. Adicionalmente, utilizamos las herramientas de configuración que nos ofrece para enlazar el servicio de Cognito con las aplicaciones web y móvil para la autenticación de los usuarios.

Amazon Cognito

Figura 3.8 Logo Amazon Cognito

Amazon Cognito es un servicio de AWS que ofrece autenticación, autorización y administración de usuarios para sus aplicaciones móviles y web. Los usuarios pueden iniciar sesión directamente con un nombre de usuario y una contraseña o a través de un tercero como Facebook, Amazon, Google o Apple.

Los dos componentes principales de Amazon Cognito son los grupos de usuarios y los grupos de identidades. Los grupos de usuarios son directorios de usuarios que proporcionan a los usuarios de las aplicaciones opciones para inscribirse e iniciar sesión. Los grupos de identidades permiten conceder a los usuarios acceso a otros servicios de AWS. [6]

Es empleado para la autenticación de los usuarios de las aplicaciones web y móvil de nuestro sistema.

Amazon CloudWatch

Figura 3.9 Logo Amazon Cloudwatch

Amazon CloudWatch es un servicio que monitorea los recursos y las aplicaciones que AWS ejecuta en tiempo real. [7]

Lo usamos durante la fase de desarrollo de las funciones lambda de nuestras APIs con el objetivo de validar las peticiones de los usuarios, así como para registrar las posibles excepciones generadas en el código de estas funciones lambda.

3.2 Lenguajes de programación

C#

Figura 3.10 Logo C#

C# es un lenguaje de programación moderno, basado en objetos y con seguridad de tipos. [8]

Es el lenguaje en el que están desarrolladas las APIs de nuestro sistema.

JavaScript

Figura 3.11 Logo JavaScript

JavaScript es un lenguaje de programación diseñado en un principio para añadir interactividad a las páginas webs y crear aplicaciones web. [9]

Es el lenguaje empleado en el frontend de nuestras aplicaciones (web y móvil) a través de bibliotecas de JavaScript como son React JS y React Native.

3.3 Tecnologías

HTML

Figura 3.12 Logo de HTML5

El Lenguaje de Marcado de Hipertexto (HTML) es el código que se utiliza para estructurar y desplegar una página web y sus contenidos. [10]

Con HTML generamos la estructura de la aplicación web.

CSS

Figura 3.13 Logo de CSS

CSS (Cascading Style Sheets), en español "Hojas de estilo en cascada", es un lenguaje de marcas enfocado a definir, crear y mejorar la presentación de un documento basado en HTML. [11]

Empleamos este lenguaje de estilos en nuestras aplicaciones con fines de diseño.

Transact-SQL

Figura 3.14 Logo Transact-SQL

Transact-SQL (T-SQL) es una extensión al SQL de Microsoft. SQL, que frecuentemente se dice ser un Lenguaje de Búsquedas Estructurado (por sus siglas en inglés), es un lenguaje de cómputo estandarizado, desarrollado originalmente por IBM para realizar búsquedas, alterar y definir bases de datos relacionales utilizando sentencias declarativas. T-SQL expande el estándar de SQL para incluir programación procedimental, variables locales, varias funciones

de soporte para procesamiento de strings, procesamiento de fechas, matemáticas, etc, y cambios a las sentencias DELETE y UPDATE. [12]

Es el lenguaje empleado para los procedimientos almacenados y las funciones de nuestra base de datos.

React JS

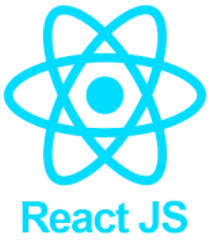

Figura 3.15 Logo React JS

React es una biblioteca de JavaScript declarativa, eficiente y flexible para construir interfaces de usuario. Permite componer IUs complejas de pequeñas y aisladas piezas de código llamadas “componentes”. [13]

Es la biblioteca de JavaScript empleada para el desarrollo de la aplicación web.

React Native

Figura 3.16 Logo React Native

React Native es un framework para aplicaciones móviles de código abierto creado por Facebook. Se usa para desarrollar aplicaciones para Android, iOS, Web y UWP (Windows) y proporciona controles de interfaz de usuario nativa y acceso completo a la plataforma nativa. Trabajar con React Native requiere un conocimiento de los aspectos básicos de JavaScript. [14]

React Native es el framework utilizado para el desarrollo de la aplicación móvil.

GIT

Figura 3.17 Logo Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia. [15]

Git es el sistema de control de versiones adoptado en el proyecto para la gestión del código de nuestras aplicaciones.

3.4 Librerías

Material-UI

Figura 3.18 Logo Material-UI

Material UI es una librería de código abierto para el desarrollo frontend que implementa los componentes gráficos de Material Design de Google.

Esta librería la usamos como apoyo para el diseño de la aplicación web gracias a su colección de componentes predefinidos.

React Native Paper

Figura 3.19 Logo React Native Paper

Es una colección de componentes personalizables y listos para producción de React Native, siguiendo las pautas de diseño de Material Design de Google.

Es empleada para el diseño de la aplicación móvil mediante el uso de sus componentes predefinidos.

React Bootstrap

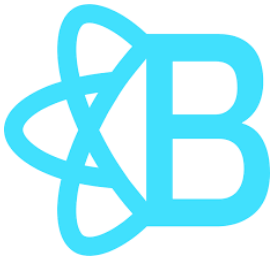

Figura 3.20 Logo React Bootstrap

React Bootstrap es una reimplementación completa de los componentes de Bootstrap usando React. [16]

En nuestro proyecto se utiliza para el diseño de la barra de navegación de la aplicación web.

3.5 Entornos de desarrollo (IDEs)

Visual Studio

Figura 3.21 Logo Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) creado por Microsoft para Windows, Linux y macOS, compatible con una gran cantidad de lenguajes de programación y enfocado fundamentalmente al desarrollo de aplicaciones para la plataforma .NET de Microsoft.

Es el entorno de desarrollo que hemos empleado para la implementación de las APIs de nuestro sistema, puesto que estas están desarrolladas dentro de la plataforma .NET.

SQL Server Management Studio

Figura 3.22 Logo SQL Server Management Studio

SQL Server Management Studio (SSMS) es un entorno integrado para administrar bases de datos de SQL Server, entre otras.

Es el entorno de desarrollo usado en el proyecto para gestionar la base de datos en SQL Server de nuestro sistema, con la creación de tablas, procedimientos almacenados y funciones auxiliares.

3.6 Herramientas

Visual Studio Code

Figura 3.23 Logo Visual Studio Code

Visual Studio Code es un editor de código fuente independiente que se ejecuta en Windows, macOS y Linux. Cuenta con una gran cantidad de extensiones y admite casi cualquier lenguaje de programación. [17]

Es el editor de código empleado en el proyecto para el desarrollo de las aplicaciones web y móvil.

GitHub

Figura 3.24 Logo GitHub

GitHub es una plataforma de hospedaje de código para el control de versiones y la colaboración.

En esta plataforma alojamos los repositorios GIT de nuestros proyectos, esto es, aplicación web, aplicación móvil y servicios web.

Postman

Figura 3.25 Logo Postman

Postman es una herramienta para la documentación y testeo de APIs, permitiendo documentar una API existente, así como organizar nuestras peticiones HTTP en colecciones y módulos. [18]

Ha sido empleado durante el desarrollo del proyecto para el testeo y validación de nuestras APIs.

Swagger

Figura 3.26 Logo Swagger

Swagger es un conjunto potente pero fácil de usar de herramientas de desarrollo de API para equipos e individuos, que permite el desarrollo en todo el ciclo de vida de la API, desde el diseño y la documentación hasta la prueba y la implementación. [19]

Ha sido empleado para la documentación de las APIs de nuestro proyecto, integrándose en las funciones lambda de los servicios web de nuestras aplicaciones.

Crystal Reports

Figura 3.27 Logo Crystal Reports

Crystal Reports es una potente herramienta de diseño y elaboración de informes altamente personalizables desarrollada por SAP.

Ha sido utilizada en la API ClockInReportsAPI para la generación de informes (empleados, roles, histórico de jornadas, etc.) solicitados desde la web, obteniendo la información a mostrar en el informe mediante conexiones a base de datos en base a los filtros recibidos desde la web.

4. Planificación del proyecto

En este apartado se tratará la metodología llevada a cabo en el desarrollo del proyecto, la planificación estimada inicialmente con sus variaciones posteriores, y, finalmente, se abordará el coste presupuestario que supondría nuestro sistema en un hipotético despliegue en producción.

4.1 Metodología

La metodología software empleada en el desarrollo del proyecto es una metodología de desarrollo basada en SCRUM. Se trata de una metodología de desarrollo ágil de carácter colaborativo en la que se va desarrollando el producto progresivamente mediante entregas regulares no superiores a un mes (*Sprint*). Estas entregas deben aportar un valor tangible al cliente.

SCRUM es una metodología que, gracias a la gestión de pequeñas entregas a través de iteraciones, se adapta a los cambios continuos que habitualmente se dan en los proyectos de desarrollo software, en los cuáles los requisitos del sistema varían continuamente durante el desarrollo del proyecto. Los principales elementos que forman parte de SCRUM son los siguientes:

Historias de usuario: Definición informal de una funcionalidad del software centrada en el usuario que lo va a usar.

Scrum Master: Lidera el equipo SCRUM y es el responsable de que se logren los objetivos fijados.

Product Owner: Es el miembro del equipo SCRUM encargado de elaborar las historias de usuario y de establecerles una prioridad en la pila de producto.

Pila de producto: Lista ordenada en orden de prioridad de las historias de usuario establecidas por el *Product Owner*.

Sprint: Cada una de las iteraciones del proyecto, en las que se deben aportar valor real al cliente, incrementando el valor del producto en desarrollo.

El motivo de la elección de esta metodología es que nos permite una gran flexibilidad en el desarrollo del producto, pudiendo añadir nuevas funcionalidades en la pila de producto y adaptando el número de *sprints* a realizar en consecuencia. Si bien SCRUM está diseñado para el trabajo en equipo en grupos de varias personas a través de reuniones diarias de corta duración entre los miembros del equipo, podemos aprovechar algunas de sus características como la gestión de la pila de producto y la entrega continua de software mediante *sprints* para el desarrollo de este proyecto.

4.2 Planificación

A continuación, se detallan la planificación inicial y final del proyecto con el diferencial de horas en cada una de las fases del proyecto.

Fases	**Duración estimada (horas)**	**Tareas** *(nombre y descripción, obligatorio al menos una por fase)*

Estudio previo / Análisis	50	Estudio de servicios de Machine learning de aws, fundamentalmente AWS Rekognition
		Análisis de tecnologías front end disponibles para emplear en la aplicación. Previsiblemente Blazor webassembly o React
		Adquisición de conocimientos más avanzados en el lenguaje *frontend* elegido en la tarea 1.2
		Análisis de sofware (diagramas de casos de uso, prototipo de la app, diseño de tablas de bbdd,etc)
Diseño / Desarollo / Implementación	180	Creación de la base de datos (SQL Server)
		Implementación del diseño *frontend* (lado cliente) de la aplicación
		Implementación del software de backend (mediante AWS lambda) que servirá de enlace entre nuestra aplicación, la base de datos y los servicios de AWS
		Desarrollo de software de gestión de entidades de la aplicación (empresas, empleados, horarios, vacaciones, etc.)
		Desarrollo del software para el análisis de imágenes mediante AWS rekognition y AWS lambda.
Evaluación / Validación / Prueba	20	Pruebas unitarias para el software de backend desarrollado
		Registro de empresas y trabajadores en la aplicación el testeo de la misma
		Comprobación de los resultados mediante el fichaje en la jornada laboral con los usuarios test agregados previamente
Documentación / Presentación	50	Realizar memoria TFT
		Preparar presentación TFT

Tabla 4.1 Planificación Inicial

Fases	**Duración estimada (horas)**	**Tareas** ***(nombre y descripción, obligatorio al menos una por fase)***
Estudio previo / Análisis	~~50~~ 40	Estudio de servicios de Machine learning de aws, fundamentalmente AWS Rekognition
		Análisis de tecnologías front end disponibles para emplear en la aplicación. Previsiblemente Blazor webassembly o React
		Adquisición de conocimientos más avanzados en el lenguaje *frontend* elegido en la tarea 1.2
		Análisis de sofware (diagramas de casos de uso, prototipo de la app, diseño de tablas de bbdd,etc)
Diseño / Desarollo / Implementación	~~180~~ 200	Creación de la base de datos (SQL Server)
		Implementación del diseño *frontend* (lado cliente) de la aplicación
		Implementación del software de backend (mediante AWS lambda) que servirá de enlace entre nuestra aplicación, la base de datos y los servicios de AWS
		Desarrollo de software de gestión de entidades de la aplicación (empresas, empleados, horarios, vacaciones, etc.)
		Desarrollo del software para el análisis de imágenes mediante AWS rekognition y AWS lambda.
Evaluación / Validación / Prueba	~~20~~ 10	Pruebas unitarias para el software de backend desarrollado
		Registro de empresas y trabajadores en la aplicación el testeo de la misma
		Comprobación de los resultados mediante el fichaje en la jornada laboral con los usuarios test agregados previamente

Documentación / Presentación	50	Realizar memoria TFT
		Preparar presentación TFT

Tabla 4.2 Planificación final

Tal y como se muestra en la planificación final, se han dado algunos cambios en la planificación inicial. En lo referente a la primera fase de análisis, se ha reducido en tiempo de desarrollo respecto de la planificación inicial debido a que finalmente se decidió optar por React JS para el desarrollo del frontent en lugar de Blazor WebAssembly, que, puesto que se trata de una tecnología muy reciente, requería de un mayor tiempo de estudio previo.

La fase de desarrollo es la que ha sufrido una mayor alteración respecto al estudio inicial de las tareas, con hasta 20 horas más estimadas. Esto se debe fundamentalmente a que se decidió desarrollar una pequeña aplicación móvil para el fichaje de los usuarios. Adicionalmente, se aumentaron el número de funcionalidades de nuestra pila de producto añadiendo, entre otras, la elaboración de informes para ser visualizados y/o descargados por los administradores en la aplicación web.

La fase de pruebas también ha soportado cambios respecto a la base inicial. En contraposición a lo planificado inicialmente, las pruebas del software de backend no se han llevado a cabo mediante test unitarios. Puesto que las aplicaciones han sido desarrolladas mediante el framework "serverless", todo el *backend* del sistema se sustenta en servicios web (APIs). La validación de estos servicios web se ha llevado a cabo mediante herramientas de testeo de APIs como Postman y Swagger.

4.3 Presupuesto

La estimación de un presupuesto para el desarrollo de cualquier producto es un elemento fundamental del análisis previo. Una mala estimación puede generar sobrecostes, y estos sobrecostes pueden provocar la paralización o cancelación de un proyecto, aún en una fase avanzada de desarrollo.

Por todo esto, en este informe se tratará de hacer una **estimación** del presupuesto para este proyecto, incluyendo tanto su desarrollo como su mantenimiento posterior. Este presupuesto será desglosado en base a varios rangos de usuarios del sistema que hemos desarrollado, estimando los costes del sistema teniendo desde unos pocos usuarios (1-100) hasta varios miles de usuarios. Adicionalmente, se hará un posible plan de precios para los usuarios en caso de un despliegue a producción. Estos presupuestos se dividen en dos grandes grupos:
Cotes fijos: Licencias software de los programas empleados en el desarrollo y salario de los empleados contratados, entre otros.
Costes variables: Recursos empleados en AWS, donde el coste es variable según su uso, así como recursos de índole empresarial como la luz, internet, impuestos, etc.

A continuación, se muestran varias tablas con la estimación presupuestaria de nuestro sistema para varios rangos de usuarios.

Hardware	
Asus zenbook pro 15 M535QE-KJ159	1.149€ (Coste fijo)
McBook Pro 13"	1.318,99€ (Coste fijo)
Software	
Visual Studio 2022	0€
Visual Studio Code	0€

Microsoft SQL Management Studio	0€
Despliegue	
Amazon EC2	150€
Amazon RDS	400€
Amazon API Gateway	0,75€
Amazon Lambda	0€
Amazon Amplify	25 €
Amazon Cognito	60€
Amazon S3	3€
Amazon Rekognition	72€
Personal	
Desarrollador *Full Stack*	35.000€
AWS *Solutions* architect	25.000€
TOTAL	
Total anual sin personal:	3.109,74€
Total anual con personal:	63.109,74€

Tabla 4.3 Presupuesto anual (50-100 usuarios)

Hardware	
Asus zenbook pro 15 M535QE-KJ159	1.149€ (Coste fijo)
McBook Pro 13”	1.318,99€ (Coste fijo)
Software	
Visual Studio 2022	0€
Visual Studio Code	0€
Microsoft SQL Management Studio	0€
Despliegue	
Amazon EC2	150€
Amazon RDS	2.400€
Amazon API Gateway	4€
Amazon Lambda	0€
Amazon Amplify	30 €
Amazon Cognito	600€
Amazon S3	30€
Amazon Rekognition	720€
Personal	
Desarrollador *Full Stack*	35.000€
AWS *Solutions* architect	25.000€
TOTAL	
Total anual sin personal:	6.501,99€
Total anual con personal:	66.501,99€

Tabla 4.4 Presupuesto anual (100-1000 usuarios)

Hardware	
Asus zenbook pro 15 M535QE-KJ159	1.149€ (Coste fijo)
McBook Pro 13”	1.318,99€ (Coste fijo)
Software	
Visual Studio 2022	0€

Visual Studio Code	0€
Microsoft SQL Management Studio	0€
Despliegue	
Amazon EC2	150€
Amazon RDS	3.000€
Amazon API Gateway	40€
Amazon Lambda	5€
Amazon Amplify	35 €
Amazon Cognito	6.000€
Amazon S3	125€
Amazon Rekognition	7.200€
Personal	
Desarrollador *Full Stack*	35.000€
AWS *Solutions* architect	25.000€
TOTAL	
Total anual sin personal:	16.805€
Total anual con personal:	7.6805€

Tabla 4.5 Presupuesto anual (1000-10000 usuarios)

Hardware: Los costes del hardware mostrado en las tablas presupuestarias están fijados en base al precio de dos portátiles actuales de gama media alta con los dos principales sistemas operativos (macOS y Windows), puesto que el proyecto contaría con dos desarrolladores.

Software: En lo referente a las licencias software, tanto Visual Studio Code como SQL Management studio son totalmente gratuitos y libres de uso comercial. En el caso de Visual Studio, existe una limitación a empresas para el uso comercial de su versión más sencilla (Community). Sin embargo, estas limitaciones no afectarían a nuestro proyecto puesto que en los términos de licencia Microsoft se indica que "Se entiende por "empresa" una organización y sus filiales que posean, de forma conjunta, bien (a) más de 250 ordenadores (PC) o usuarios o (b) un millón de dólares (o su equivalente en otras monedas) en ingresos anuales, y "filiales" hace referencia a aquellas entidades que dirigen una organización (por propiedad mayoritaria), están dirigidas por otra organización o comparten la dirección común junto con la misma.". [20]

Despliegue: En este apartado se desglosan los costes de los servicios de AWS empleados en el proyecto. Estos servicios han sido calculados de manera aproximada mediante el uso de la calculadora de precios que nos ofrece AWS, que nos permite configurar una estimación de costos mediante la configuración individual de cada uno de los servicios que nos ofrecen. [21]

Personal: El personal designado para el proyecto sería de dos empleados. Un desarrollador *fullstack* para el desarrollo de las aplicaciones y un arquitecto de soluciones de AWS (certificado) para la gestión de la infraestructura del sistema en la nube. Los salarios han sido estimados en base a la media de salarios ofertados en linkedin para los puestos indicados.

Hay que tener en cuenta que, por las limitaciones del proyecto, no se han tenido en cuenta costes logísticos propios de una empresa como el coste del alquiler de una oficina, el impuesto de inmuebles de esta, el coste de luz e internet, etc. Para la estimación de este presupuesto, supondremos que los dos empleados de la empresa trabajarán en remoto durante la fase de desarrollo y mantenimiento del producto.

En base a lo presupuestado en los diferentes tramos y a los diferentes planes de precios que ofertan otras empresas y que han sido descritos en el apartado de estado del arte de este informe, podemos establecer unos planes de precios de la siguiente manera:

Plan de precios	Características principales incluidas
Plan mensual "base": 1,00€ / usuario	Registros de jornada (entradas y salidas)
	Informes básicos de registro de jornadas
	Gestión de roles
	Gestión de vacaciones
Plan mensual "*premium*": 2,00€ / usuario	Registros de jornada (entradas y salidas), pudiendo agregar observaciones al fichaje
	Restricción de geolocalización en el fichaje
	Informes avanzados (fichajes, roles, empleados, etc.)
	Gestión de roles
	Gestión de vacaciones
	Gestión de lugares de trabajo

Tabla 4.6 Plan de precios del proyecto

5. Desarrollo

Este apartado constituye el elemento fundamental del informe. En él, se detallarán los aspectos de diseño más relevantes, se definirá la estructura de los componentes esenciales del sistema, y se explicará el desarrollo del proyecto en sus distintas fases.

5.1 Análisis

En esta sección se detalla el estudio previo a la implementación del sistema, teniendo en cuenta las funcionalidades requeridas por los clientes. Este estudio incluye la creación de diagramas de casos de uso, así como la elaboración de las historias de usuario, entre otras.

5.1.2 Historias de usuario

Como se comentó previamente en este informe en la descripción de la metodología SCRUM, las historias de usuario son una representación informal de los requisitos funcionales de nuestro sistema, enfocadas desde el punto de vista del usuario.

En esta sección se procede a enumerar las historias de usuario más significativas de nuestro sistema.

ID	HU-1
TÍTULO	Registro de empresa
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como administrador deseo poder registrar mi empresa para gestionar mis empleados
CRITERIOS DE ACEPTACIÓN	1. Se envía un email al usuario con un código de verificación 2. Se muestra la vista con un formulario donde insertar el código de verificación enviado

Tabla 5.1 HU-1 Registro de empresa

ID	HU-2
TÍTULO	Vista de empleados
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como administrador deseo poder ver el listado de empleados para ver visualizar los empelados registrados en la empresa
CRITERIOS DE ACEPTACIÓN	1. Vista con el listado de empleados de la empresa

Tabla 5.2 HU-2 Vista de empleados

ID	HU-3
TÍTULO	Crear empleado
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como administrador deseo poder registrar un empleado para que puedan registrar sus jornadas
CRITERIOS DE ACEPTACIÓN	1. Se envía un email al empleado con una contraseña temporal en la aplicación móvil 2. Se muestra la vista del listado de empleados con un mensaje de éxito de creación de empleado

Tabla 5.3 HU-3 Crear empleado

ID	HU-4
TÍTULO	Crear un rol
PRIODIDAD	Media
DESCRIPCIÓN	Yo como administrador deseo poder crear un rol en mi empresa para asignar un rol a cada empleado
CRITERIOS DE ACEPTACIÓN	1. Se muestra la vista del listado de roles con un mensaje de éxito de creación de rol

Tabla 5.4 HU-4 Crear un rol

ID	HU-5
TÍTULO	Crear un lugar de trabajo
PRIODIDAD	Baja
DESCRIPCIÓN	Yo como administrador deseo poder registrar lugares de trabajo de mi empresa para asignar a cada empleado un lugar de trabajo determinado
CRITERIOS DE ACEPTACIÓN	1. Se muestra la vista del listado de lugares de trabajo con un mensaje de éxito de creación de lugar de trabajo

Tabla 5.5 HU-5 Crear un lugar de trabajo

ID	HU-6
TÍTULO	Visualizar informe
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como administrador deseo poder visualizar informes mi empresa para obtener información relativa a esta
CRITERIOS DE ACEPTACIÓN	1. Se muestra el informe al usuario en un visor de PDF en la vista de informes 2. Se muestra la vista con un formulario donde insertar el código de verificación enviado

Tabla 5.6 HU-6 Visualizar informe

ID	HU-7
TÍTULO	Descargar informe
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como administrador deseo poder descargar informes de mi empresa para almacenar personalmente la información de esta
CRITERIOS DE ACEPTACIÓN	1. Se descarga en el sistema de archivos local del usuario en formato PDF el informe visualizado en ese momento en la web

Tabla 5.7 HU-7 Descargar informe

ID	HU-8
TÍTULO	Fichar entrada
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como empleado deseo poder registrar mi fichaje de entrada de jornada para guardar un inicio de jornada

CRITERIOS DE ACEPTACIÓN	1. Se actualiza el calendario con registros de jornada del empleado con el nuevo registro de entrada

Tabla 5.8 HU-8 Fichar entrada

ID	HU-9
TÍTULO	Fichar salida
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como empleado deseo poder registrar mi fichaje de salida de jornada para guardar un fin de jornada
CRITERIOS DE ACEPTACIÓN	1. Se actualiza el calendario con registros de jornada del empleado con el nuevo registro de salida

Tabla 5.9 HU-9 Fichar salida

ID	HU-10
TÍTULO	Fichaje con reconocimiento facial
PRIODIDAD	Alta
DESCRIPCIÓN	Yo como usuario deseo poder enviar una foto con mi rostro para realizar el fichaje de jornada (entrada o salida)
CRITERIOS DE ACEPTACIÓN	1. Se actualiza el calendario del histórico con el nuevo registro de jornada

Tabla 5.10 HU-10 Fichaje con reconocimiento facial

ID	HU-11
TÍTULO	Ver registros de fichaje
PRIODIDAD	Media
DESCRIPCIÓN	Yo como empleado deseo poder ver mi historial de fichajes en un calendario para comprobar mi histórico de jornadas
CRITERIOS DE ACEPTACIÓN	1. Se muestra un calendario con los registros de entrada y salida de cada día

Tabla 5.11 HU-11 Ver registros de fichaje

ID	HU-12
TÍTULO	Enviar ubicación en el fichaje
PRIODIDAD	Media
DESCRIPCIÓN	Yo como empleado deseo poder enviar mi ubicación actual al realizar un fichaje para certificar mi localización en el momento del fichaje
CRITERIOS DE ACEPTACIÓN	1. Se visualizan la localización del fichaje en el informe de registros de fichaje

Tabla 5.12 HU-12 Enviar ubicación en el fichaje

ID	HU-13
TÍTULO	Enviar observaciones en el fichaje
PRIODIDAD	Baja

DESCRIPCIÓN	Yo como empleado deseo poder enviar una observación al realizar un fichaje para indicar algún comentario sobre este
CRITERIOS DE ACEPTACIÓN	1. Se visualizan las observaciones del fichaje en el informe de registros de fichaje

Tabla 5.13 HU-13 Enviar observaciones en el fichaje

5.1.3 Requisitos

Una de las fases iniciales de cualquier desarrollo software consiste en identificar los requisitos que este sistema software debe satisfacer. Estos requisitos del sistema se clasifican en dos grandes grupos; requisitos funcionales y requisitos no funcionales.

Requisitos no funcionales

Los requisitos no funcionales se definen como aquellos que indican propiedades o características del propio sistema, abstrayéndose de las funcionalidades de este. Entre los requisitos no funcionales de nuestro proyecto destacan los siguientes:

ID	Descripción
NF01	Las aplicaciones del sistema deben ser desplegadas en AWS Amplify
NF02	El sistema deberá disponer de un entorno de desarrollo y otro de producción
NF03	El sistema debe alojar su base de datos a través del servicio AWS RDS *(Relational Database Service)*
NF04	La base de datos del sistema será SQLServer
NF05	El sistema debe contar con dos bases de datos en el servidor (desarrollo y producción)
NF06	El sistema hará uso de AWS Cognito para la autenticación de usuarios
NF07	El sistema contará con dos grupos de usuarios en AWS Cognito. Uno para la aplicación web (administradores) y otro para la aplicación móvil (empleados)
NF08	El sistema empleará AWS Rekognition para el reconocimiento facial de los empleados en el proceso de fichaje
NF09	El sistema hará uso de API Gateway para la implementación de las APIs que comunicarán las aplicaciones con la base de datos y otros servicios de AWS
NF10	El sistema hará uso de AWS Lambda para implementar la lógica de las APIs
NF11	El sistema hará uso de *buckets* S3 para el almacenamiento de los logos de la empresa, imágenes de los lugares de trabajo y de los empleados, y *backups* de la base de datos
NF12	El sistema empleará C# como lenguaje de programación del *backend* implementado en AWS Lambda
NF13	Las aplicaciones deben tener una interfaz de usuario sencilla e intuitiva
NF14	El idioma inicial de la interfaz de usuario de las aplicaciones será en inglés
NF15	La aplicación web debe ser desarrollada con la biblioteca React JS
NF16	La web debe de ser compatible en las versiones más modernas de los principales navegadores de internet (Google Chrome, Safari, Mozilla Firefox, Microsoft Edge, etc.)
NF17	El sistema requerirá de acceso a internet para tratar la información vía servicios web (API)
NF18	El sistema mostrará pantallas de carga durante el tiempo que duren las llamadas a los servicios web

NF19	El sistema se comunicará con la base de datos mediante llamadas a servicios web (API)
NF20	La aplicación móvil debe ser desarrollada para el sistema operativo Android
NF21	La aplicación móvil debe ser desarrollada mediante el *framework* React Native
NF22	La aplicación móvil debe solicitar al usuario permisos de uso de la cámara y ubicación para el fichaje

Tabla 5.14 Requisitos no funcionales

Requisitos funcionales

Los requisitos funcionales se definen como aquellos que indican cómo funciona el softwareEntre los requisitos funcionales de nuestro proyecto destacan los siguientes:

ID	Descripción
F01	El sistema debe permitir registrar una empresa
F02	El sistema debe permitir a los usuarios iniciar sesión
F03	El sistema debe permitir a los usuarios cerrar sesión
F04	El sistema debe permitir a los usuarios restablecer su contraseña
F05	El sistema debe permitir a los administradores editar los datos de la empresa
F06	El sistema debe permitir a los administradores crear, ver, editar y eliminar empleados
F07	El sistema creará un usuario para la aplicación móvil por cada empleado creado
F08	El sistema debe permitir a los administradores crear, ver, editar y eliminar roles
F09	El sistema debe permitir a los administradores crear, ver, editar y eliminar lugares de trabajo
F10	El sistema debe permitir a los administradores visualizar y descargar informes de la empresa (roles, empleados, lugares de trabajo, etc.) en formato PDF
F11	El sistema debe permitir a los administradores ver el resumen con los empleados
F12	El sistema debe permitir a los administradores crear nuevos administradores
F13	El sistema debe permitir a los administradores ver el número de empleados actuales en período de vacaciones, así como los empleados en horario de trabajo o fuera de él
F14	El sistema debe permitir a los administradores ver las últimas entradas y salidas de jornada de los empleados
F15	El sistema debe permitir a los empleados registrar su inicio de jornada
F16	El sistema debe permitir a los empleados registrar su fin de jornada
F17	El sistema debe permitir a los empleados enviar una observación (texto) en su registro de jornada (inicio o fin)
F18	El sistema debe mostrar un calendario con el historial de jornadas del empleado
F19	El sistema debe mostrar los períodos vacacionales del empleado
F20	El sistema debe permitir al empleado tomar una foto en el proceso de registro de la jornada
F21	El sistema debe forzar al usuario a cambiar su contraseña cuando inicia sesión por primera vez tras haber sido dado de alta por un administrador

Tabla 5.15 Requisitos funcionales

5.1.4 Diagrama de casos de uso

El diagrama de casos de uso representa la interacción entre los actores o agentes del sistema con el sistema en sí mismo

A continuación, se muestran los diagramas de uso de nuestro sistema. Puesto que contamos con dos aplicaciones, cada una de ellas con un actor o agente diferente, se ha elaborado un diagrama de casos de uso para cada una de ellas:

Diagrama de casos de uso de la aplicación web

Figura 5.1 Diagrama de casos de uso aplicación web

Diagrama de casos de uso de la aplicación móvil

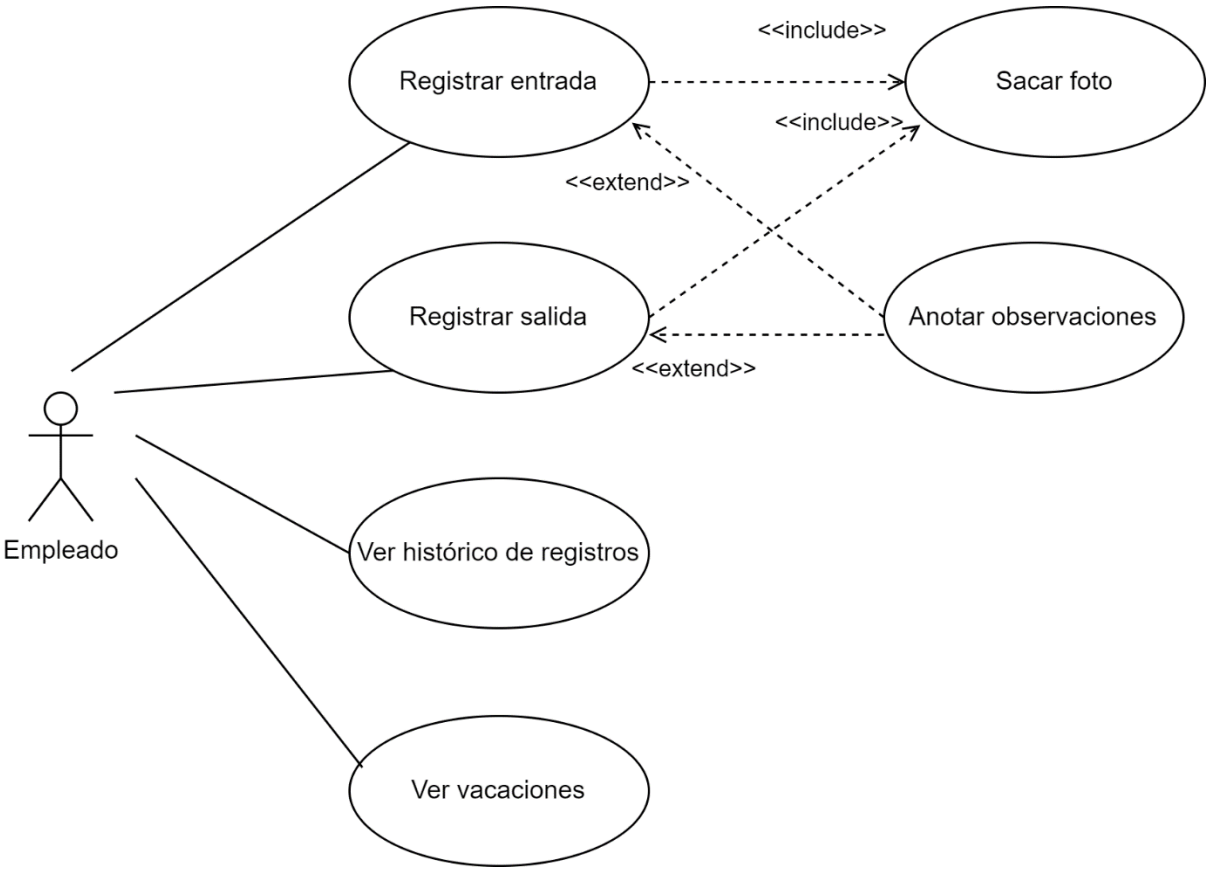

Figura 5.2 Diagrama de casos de uso aplicación móvil

5.1.5 Especificación de casos de uso

La especificación de casos de uso es un procedimiento que detalla los casos de uso. En este procedimiento se describe el caso de uso, indicando además sus actores involucrados, así como las precondiciones, postcondiciones, posibles extensiones y excepciones del caso de uso.

A continuación, se especifican algunos de los casos de usos más relevantes de nuestro sistema:

CASO DE USO	**1**	**Registrar Empresa**
Descripción	El administrador de la empresa registra su empresa en la web	
Actores	Administrador de la empresa	
Precondiciones		
Flujo normal	**Paso**	**Acción**
	1	El usuario hace click en "*Register your company*"
	2	El sistema muestra el formulario de registro de la empresa

CASO DE USO	1	Registrar Empresa	
	3	El usuario rellena el formulario de registro	
	4	El usuario hace click en "*Register*"	
Postcondiciones	Se registra la empresa con los datos enviados		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones		No se rellenan todos los campos obligatorios del formulario	

Tabla 5.16 Caso de uso de registrar empresa

CASO DE USO	2	Crear empleado
Descripción	El administrador de la empresa crea un empleado en la app web	
Actores	Administrador de la empresa	
Precondiciones	El usuario ha iniciado sesión en la aplicación web	
Flujo normal	**Paso**	**Acción**
	1	El usuario hace click en "*Manage Employees"* desde el menú "*Employees"*
	2	El sistema muestra el listado de empleados con una cabecera con filtros para su búsqueda
	3	El usuario hace click en el botón "+" en la esquina superior derecha
	4	El sistema muestra un formulario de registro de empleado
	5	El usuario rellena el formulario del nuevo empleado
	6	El usuario hace click en "*Create employee"*
	7	El sistema muestra el resumen de los empleados y un *header* con la notificación del registro satisfactorio del empleado
Postcondiciones	Se registra el empleado y se le envía un mail con su contraseña temporal en la app móvil	
Variaciones	**Paso**	**Acción**
	1	El usuario hace click en "*Roster Summary*" desde el menú "*Employees"*
	2	El sistema muestra el listado de empleados con una cabecera con filtros para su búsqueda
	3	El usuario hace click en el botón "+" en la esquina superior derecha
	4	El sistema muestra un formulario de registro de empleado

CASO DE USO	2	Crear empleado	
	5	El usuario rellena el formulario del nuevo empleado	
	6	El usuario hace click en "*Create employee"*	
	7	El sistema muestra el resumen de los empleados y un *header* con la notificación del registro satisfactorio del empleado	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones		El usuario ya existe en el sistema	

Tabla 5.17 Caso de uso de crear empleado

CASO DE USO	3	Crear usuario administrador	
Descripción	El administrador de la empresa registra a otros usuarios con permisos de administración		
Actores	Administrador de la empresa		
Precondiciones	El usuario ha iniciado sesión en la app web		
Flujo normal	**Paso**	**Acción**	
	1	El usuario hace click en "*Users*" desde el menú "*Company*"	
	2	El sistema muestra el listado de usuarios de la empresa (administradores y empleados)	
	3	El usuario hace click en el botón "+" en la esquina superior derecha	
	4	El sistema muestra al usuario un *popup* con los campos email y dni para el registro del usuario administrador	
	5	El usuario hace click en "*Save"*	
	6	El sistema muestra el listado de usuarios y un *header* con la notificación de adición satisfactoria del usuario	
Postcondiciones	Se registra el usuario y se le envía un mail con su contraseña temporal		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones		El usuario ya existe en el sistema	

Tabla 5.18 Caso de uso de crear usuario administrador

CASO DE USO	4	Generar Informe
Descripción	El administrador de la empresa visualiza un informe	
Actores	Administrador de la empresa	
Precondiciones	El usuario ha iniciado sesión en la app web	

CASO DE USO	4	Generar Informe	
Flujo normal	**Paso**	**Acción**	
	1	El usuario hace click en "*Reports*" desde el menú "*Company*"	
	2	El sistema muestra un combo con los tipos de informes disponible y los botones desactivados "*Load Report*" y "*Download*"	
	3	El usuario selecciona un tipo de informe	
	4	El sistema muestra al usuario los filtros asociados al tipo de informe seleccionado	
	5	El usuario aplica los filtros del informe	
	6	El usuario hace click en "*Load Report"*	
	7	El sistema muestra al usuario el informe en formato PDF	
Postcondiciones	El usuario visualiza el informe solicitado		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	1	El usuario desea descargar el informe generado	CU-5: Descargar informe
Excepciones			

Tabla 5.19 Caso de uso de generar informe

CASO DE USO	5	Descargar informe	
Descripción	El administrador de la empresa descarga un informe generado en la web		
Actores	Administrador de la empresa		
Precondiciones	El usuario ha iniciado sesión en la app web		
	El usuario ha generado un informe		
Flujo normal	**Paso**	**Acción**	
	1	El usuario hace click en "*Reports*" desde el menú "*Company*"	
	2	El sistema muestra un combo con los tipos de informes disponible y los botones desactivados "Load *Report*" y "*Download*"	
	3	El usuario selecciona un tipo de informe	
	4	El sistema muestra al usuario los filtros asociados al tipo de informe seleccionado	
	5	El usuario aplica los filtros del informe	
	6	El usuario hace click en "*Load Report"*	

CASO DE USO	5	Descargar informe	
	7	El sistema muestra al usuario el informe en formato PDF	
	8	El usuario hace click en "*Download*"	
	9	El sistema descarga el informe en formato PDF en el equipo del usuario	
Postcondiciones	El usuario descarga el informe en formato PDF en su equipo		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones			

Tabla 5.20 Caso de uso de descargar informe

CASO DE USO	6	Registrar entrada	
Descripción	El empleado o trabajador registra su inicio de jornada laboral		
Actores	Empleado		
Precondiciones	El usuario ha iniciado sesión en la app móvil		
Flujo normal	**Paso**	**Acción**	
	1	El usuario hace click en "*Clock In*"	
	2	El sistema muestra un modal para agregar una observación opcional al fichaje con dos botones ("Close" o "Continue")	
	3	El usuario hace click en "Continue"	
	5	El sistema le muestra una pantalla con la cámara	
	6	El usuario toma una autofoto haciendo click en el botón con el icono de la cámara	
	7	El sistema muestra la foto tomada	
	8	El usuario hace click en "*Send photo*"	
	9	El sistema retorna al usuario a la pantalla principal con el estado actual	
Postcondiciones	Se registra el inicio de jornada del usuario		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones		La foto tomada no corresponde con la cara registrada asignada al usuario	

Tabla 5.21 Caso de uso de registrar entrada

5.2 Diseño

En este apartado se detallará el diseño de la interfaz de usuario de las aplicaciones de nuestro sistema, así como de la base de datos.

5.2.1 Diseño de la interfaz de usuario

La interfaz de usuario es el medio gráfico mediante el cual nuestras aplicaciones se comunican con el usuario. Debe ser intuitiva, agradable a la vista y carente de elementos innecesarios que sobrecarguen el enfoque del usuario.

En las aplicaciones del proyecto (web y móvil), se ha optado por un diseño minimalista, fundamentalmente por los siguientes motivos:
Aplicación web: Funciona a modo de panel de administración, por lo que se requiere que el total de acciones posibles en la web tengan una estructura clara y concisa para no inducir a errores al administrador de la empresa.
Aplicación móvil: Aplicación con unas pocas funcionalidades (registrar fichaje, ver histórico y vacaciones), que será utilizada por los usuarios como mínimo 2 veces diarias para el fichaje, por lo que debe ser intuitiva y con un catálogo de menús muy restringido.

5.2.2 Gama de colores

Se ha empleado la misma elección de colores en ambas aplicaciones debido a que forman parte del mismo sistema. Dado que la idea fundamental del proyecto recae en el uso de servicios en la nube de AWS, se ha decidido emplear la misma gama de colores que podemos ver en la consola de administración de AWS, esto es, una paleta de colores derivada del naranja y azul oscuro. A continuación, se muestra la gama de colores empleada con su código en hexadecimal y su uso en la aplicación.

COLOR	CÓDIGO (HEX)	MOTIVO
	#FF9900 (Primario)	Texto de los títulos, menús y botones y algunos elementos de diseño (bordes, *loading spinners*, etc.)
	#232F3E (Primario)	Barra de navegación de la web, cabecera de la app móvil, color de fondo de los botones y otros contenedores
	#FEBD69 (Secundario)	Textos de títulos de cabeceras secundarias
	#37475A (Secundario)	Cabeceras de algunas páginas y tablas y combobox de tipos de informe
	#666666	Deshabilitado de los botones de la pantalla de informes
	#008000	Barra de estado de fichaje diario completo (app móvil)
	#FF0000	Barra de estado de fichaje bloqueado (vacaciones, ubicación incorrecta) (app móvil)

	#FFFFFF	Color de fondo y algunos textos
	#000000	Color de textos

Tabla 5.22 Gama de colores

5.2.3 Pantallas de la aplicación web

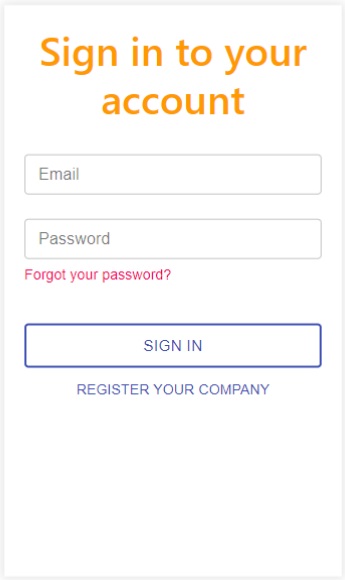

Figura 5.3 Pantalla principal de la web (Login)

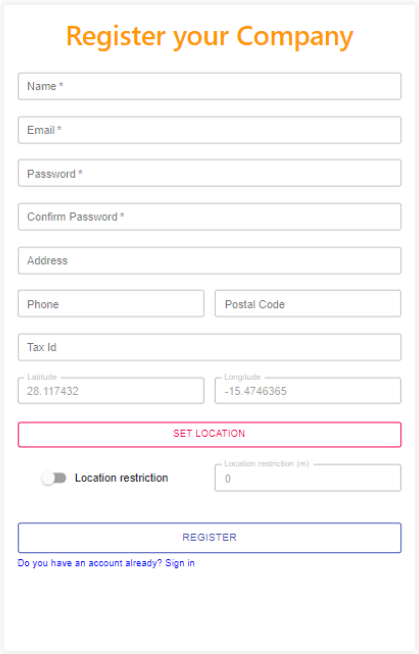

Figura 5.4 Registro de una empresa (RegisterCompany)

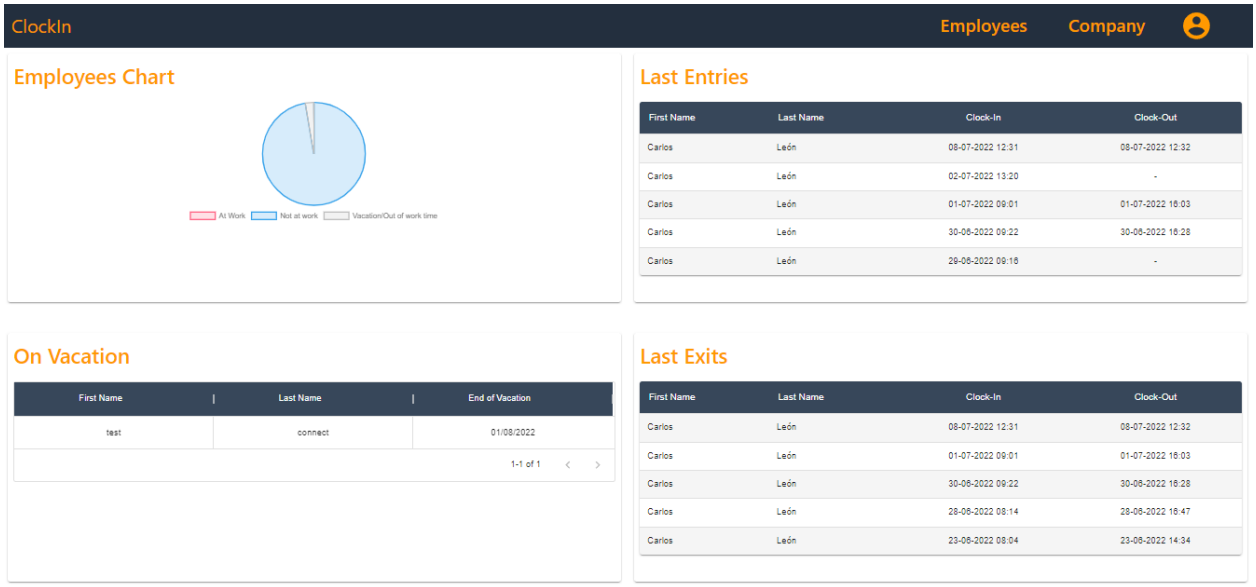

Figura 5.5 Dashboard (Home)

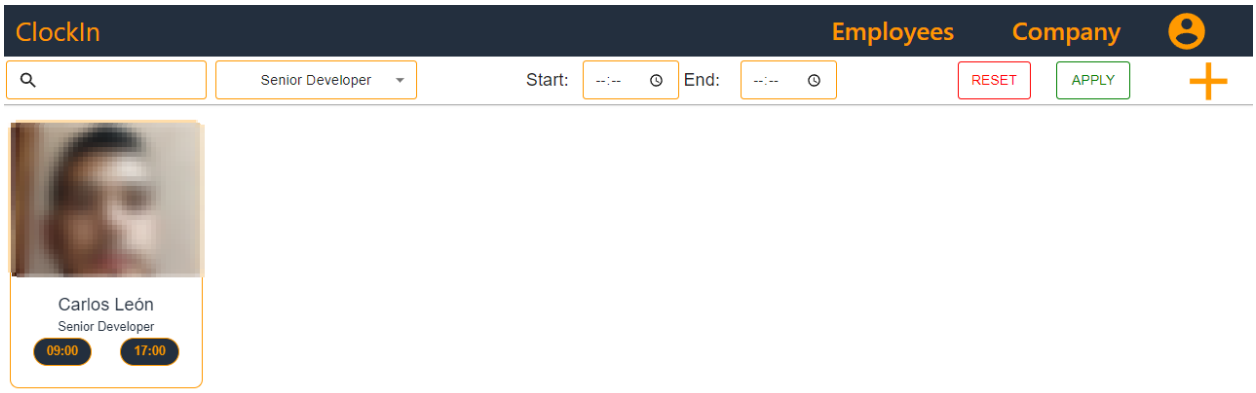

Figura 5.6 Gestión de empleados (ManageEmployees)

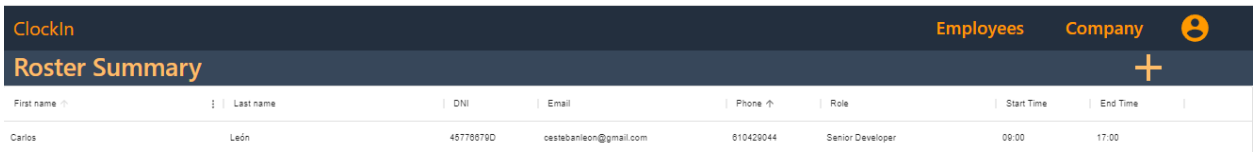

Figura 5.7 Resumen de empleados (RosterSummary)

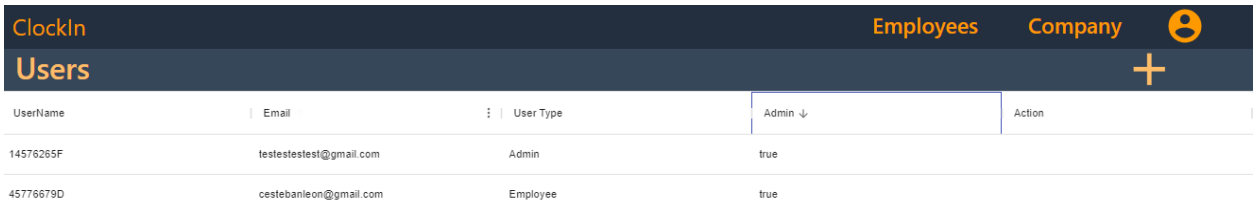

Figura 5.8 Usuarios (Users)

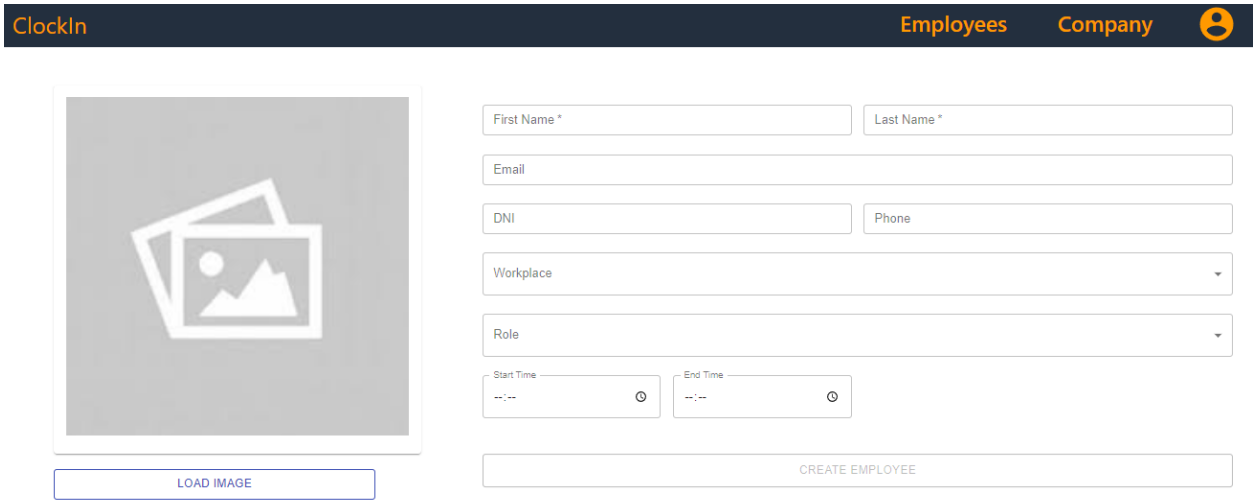

Figura 5.9 Creación de empleado

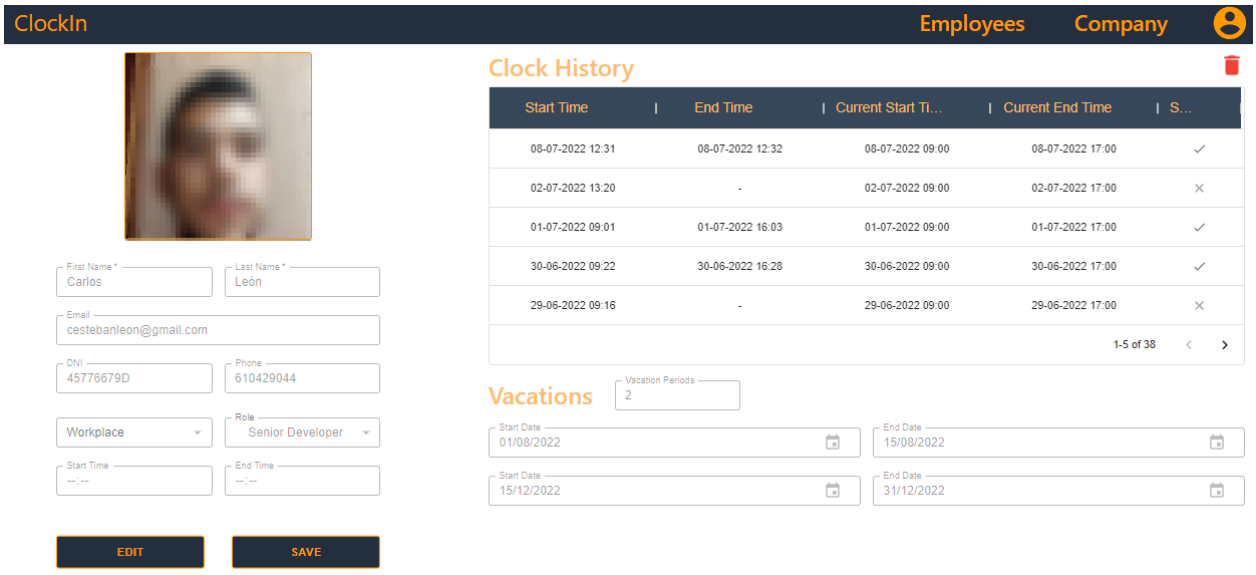

Figura 5.10 Edición de empleado (Employee/{dni})

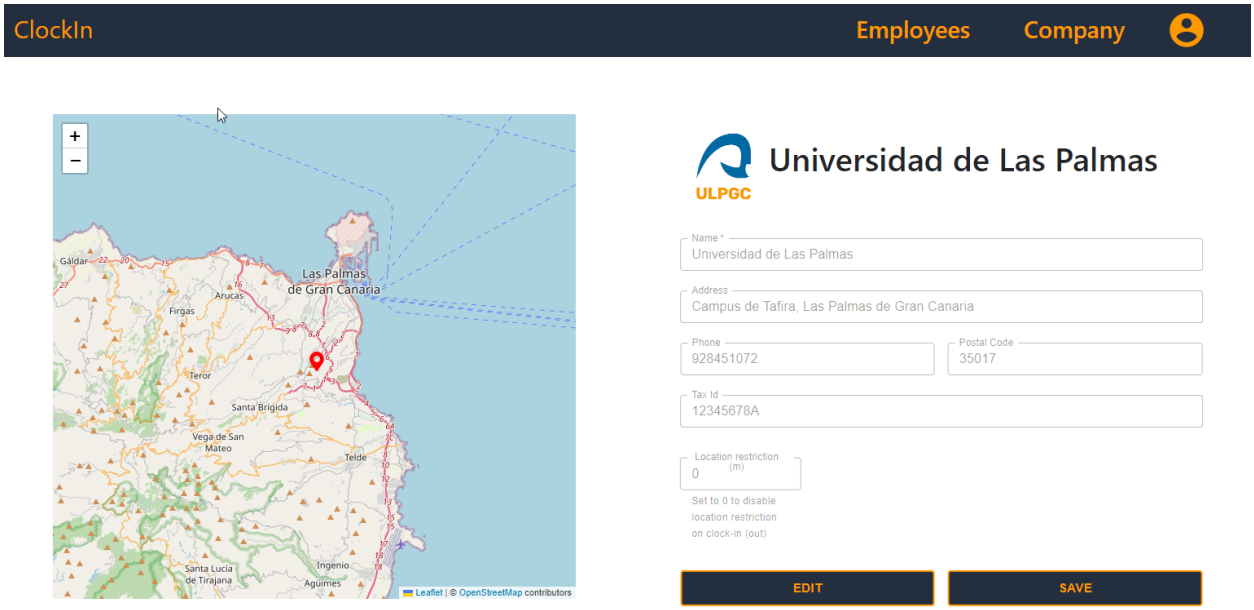

Figura 5.11 Edición de la empresa (MyCompany)

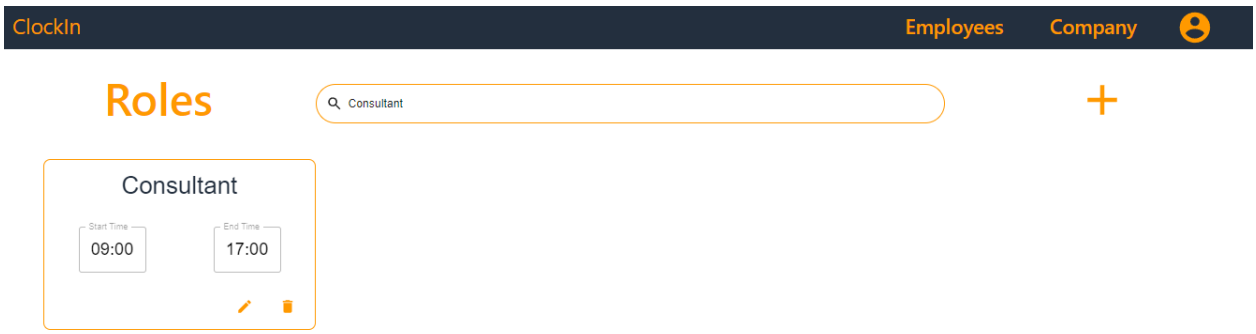

Figura 5.12 Administración de roles (Roles)

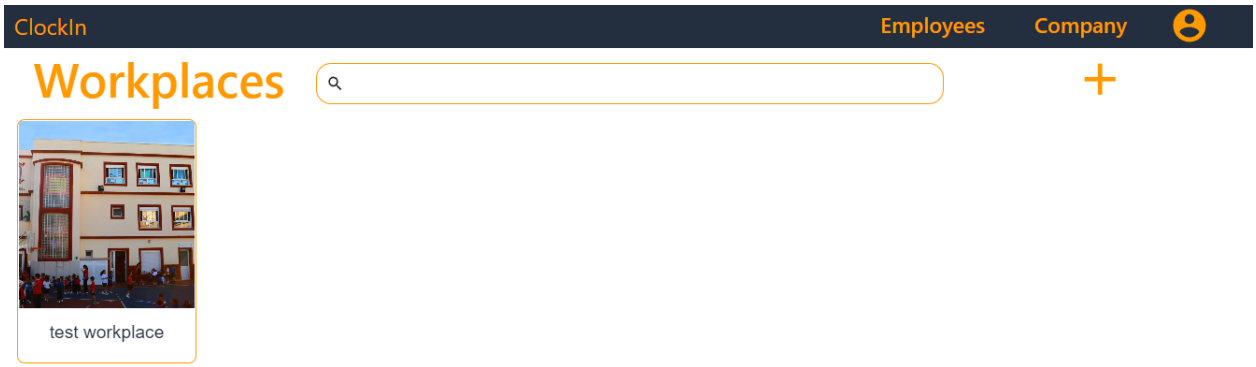

Figura 5.13 Administración de lugares de trabajo (Workplaces)

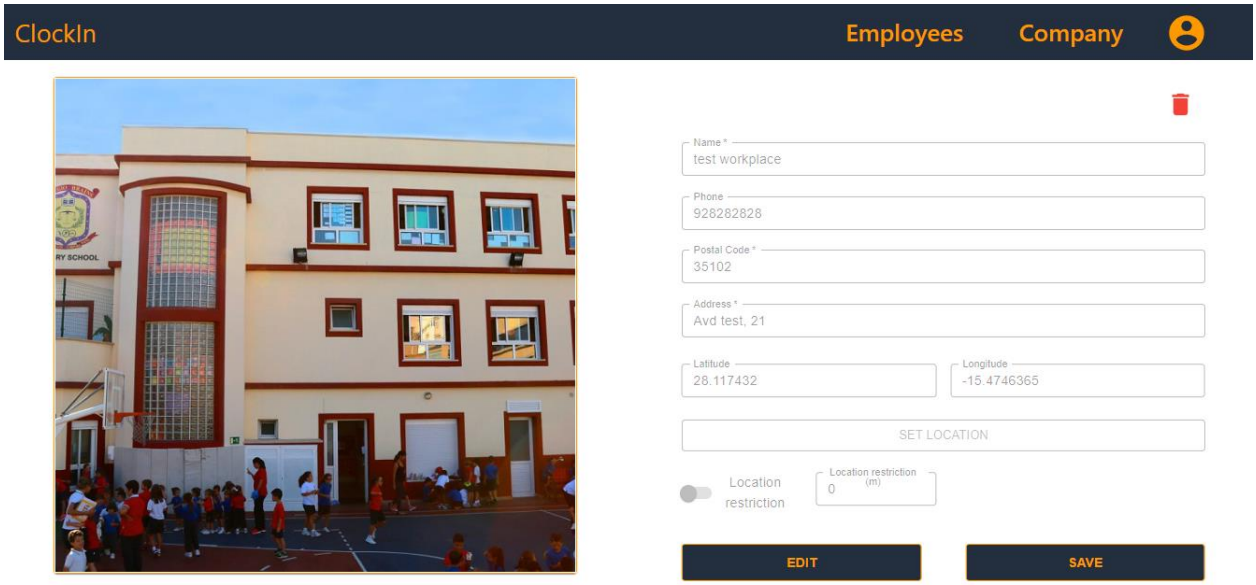

Figura 5.14 Edición de lugar de trabajo (Workplace/{id})

Figura 5.15 Informes (Reports)

5.2.4 Pantallas de la aplicación móvil

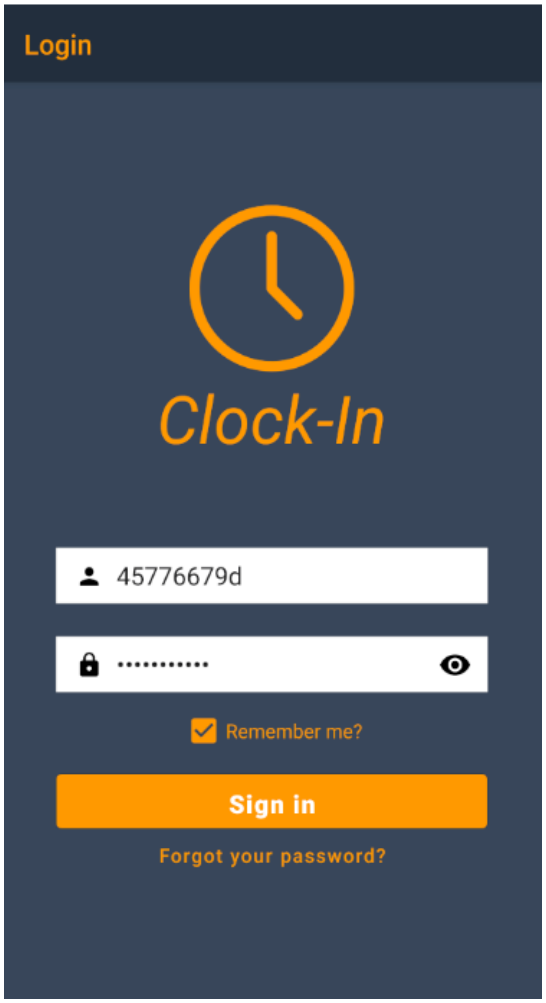

Figura 5.16 Login

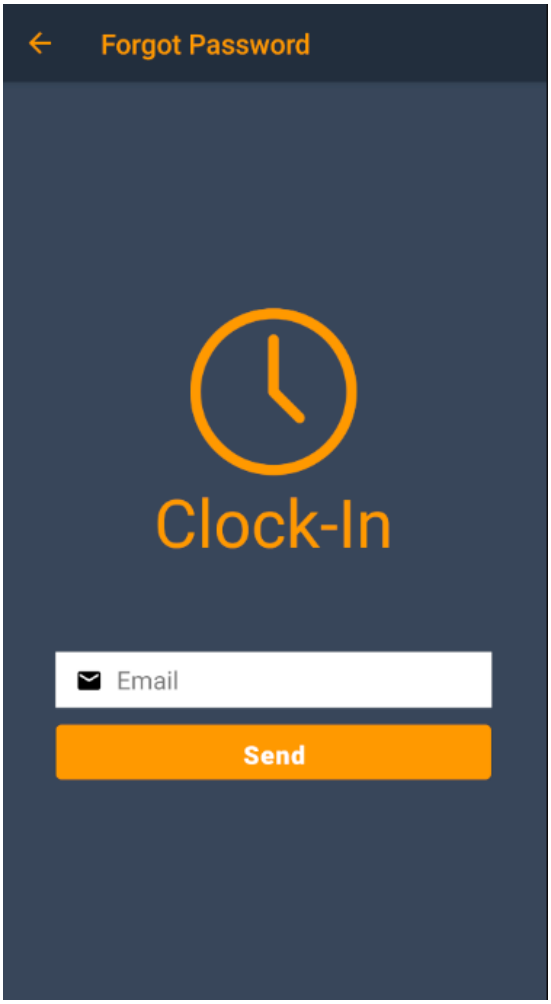

Figura 5.17 Forgot Password

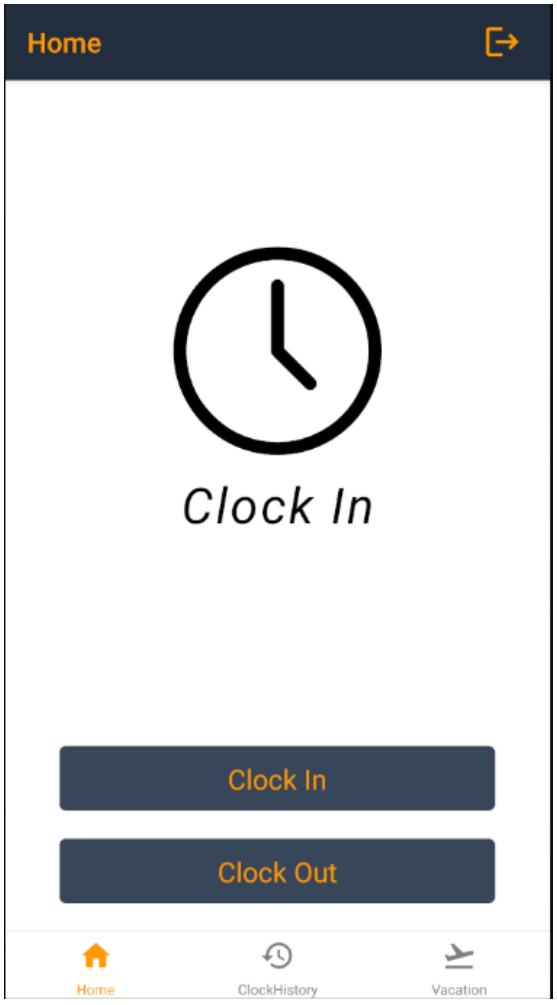

Figura 5.18 Home

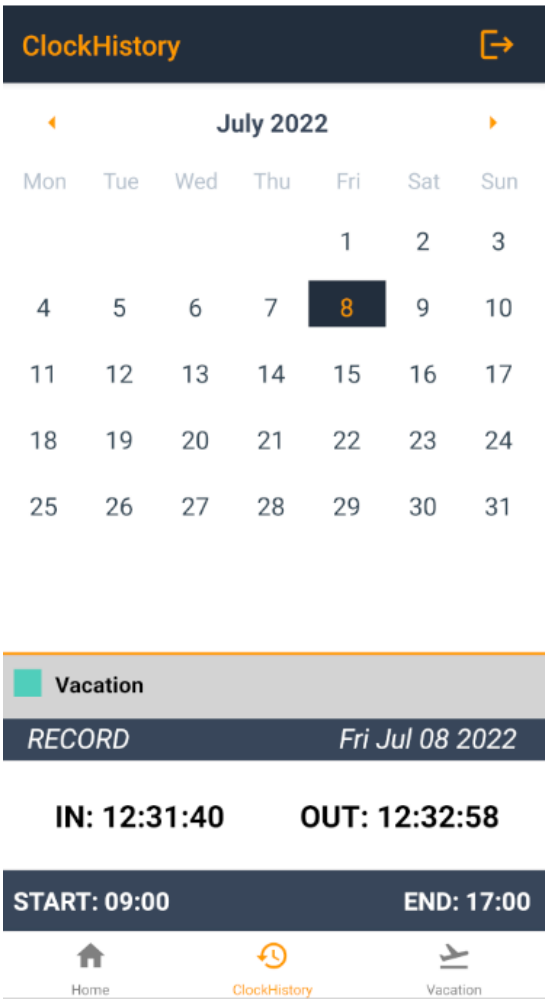

Figura 5.19 Clock History

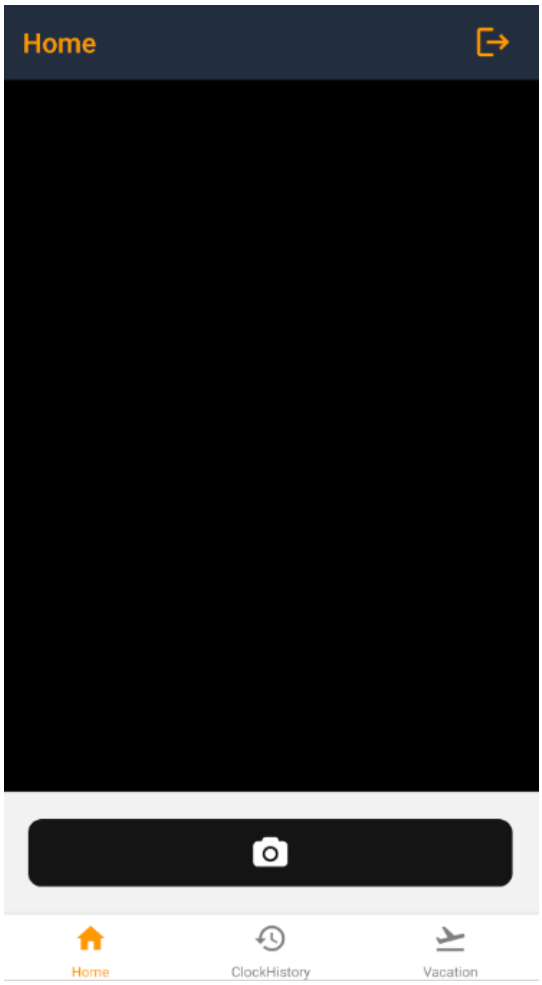

Figura 5.20 Clock Picture

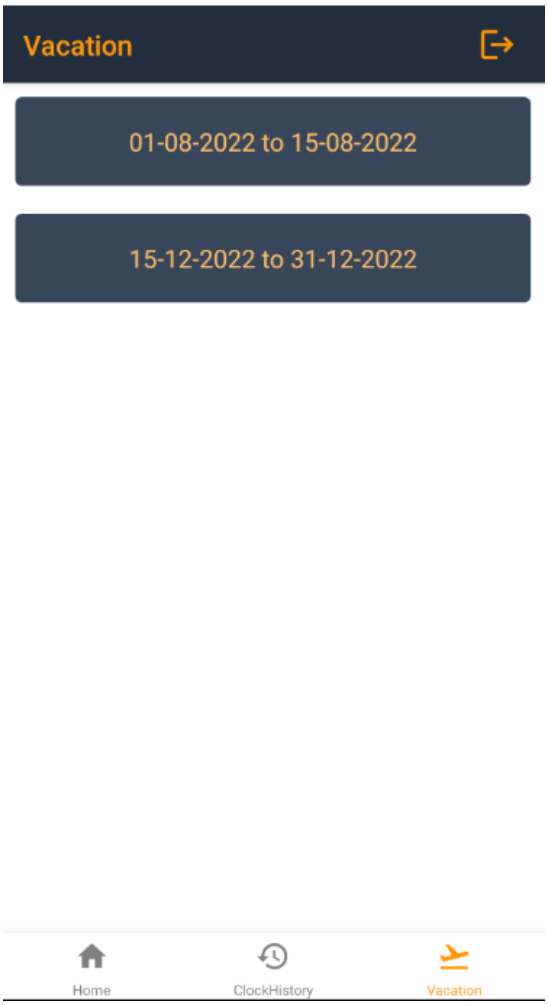

Figura 5.21 Vacation

5.2.5 Diseño software

Con relación al diseño software de nuestro sistema, la principal característica es que se trata de un sistema con una arquitectura *serverless* o sin servidor. Esto quiere decir que se usa un software externo a la propia aplicación o aplicaciones cliente para hacer las labores de *backend* del sistema, de tal modo que la lógica de negocio y la comunicación entre las aplicaciones cliente (web y móvil) y la base de datos las lleva a cabo este software externo.

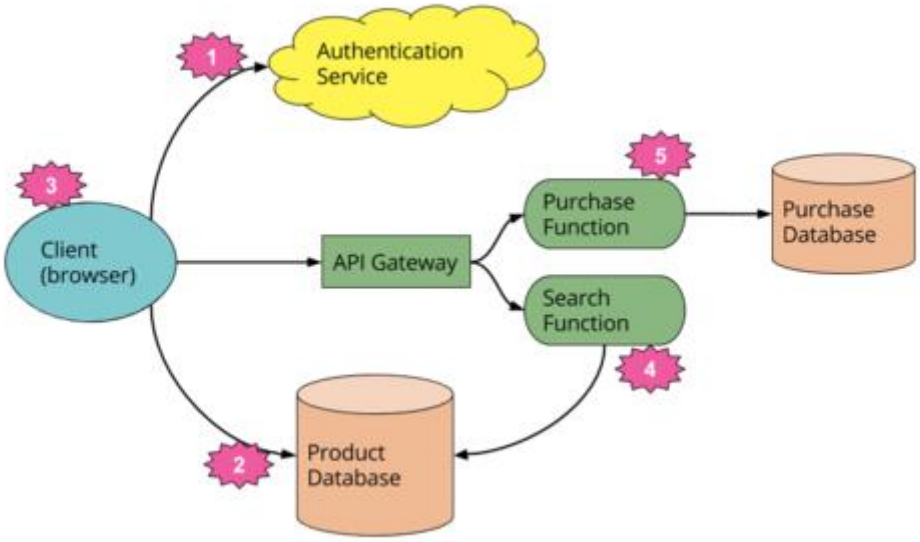

Figura 5.22 Serverless architecture simplified [20]

Se trata de un paradigma frecuentemente empleado en sistemas como el que nos ocupa, en el cual las aplicaciones se despliegan en la nube, y estas a su vez consumen de otros servicios de la nube (autenticación, bases de datos, etc.).

En ambas aplicaciones (web y móvil) hacemos usos de estos servicios externos (autenticación, BBDD, reconocimiento facial, etc.) a la propia aplicación mediante llamadas a APIs desplegadas en AWS y cuya estructura e implementación serán descritas posteriormente en este informe. Más concretamente, hemos desplegado tres APIs para el *backend* de nuestro sistema, que se definen a continuación:

ClockInAPI: API para la aplicación web.

- Establece la comunicación entre la aplicación web y la base de datos para el tratamiento de los datos.
- Hace de puente para el proceso de autenticación mediante el uso del servicio en la nube AWS Cognito.
- Gestiona el tratamiento de imágenes en la web (almacenamiento y subida) mediante el uso del servicio en la nube del servicio Amazon S3 *(Amazon Simple Storage Service)*, generando URLs prefirmadas para que desde la aplicación cliente se pueda acceder a este repositorio S3 tanto para almacenar como para obtener imágenes.

ClockInConnectAPI: API para la aplicación móvil.

- Establece la comunicación entre la aplicación móvil y la base de datos para el tratamiento de los datos.
- Hace de puente para el proceso de autenticación mediante el uso del servicio en la nube AWS Cognito.

- Gestiona el tratamiento del fichaje en la jornada laboral mediante el uso del servicio en la nube Amazon Rekognition en conjunción con el servicio Amazon S3, comprobando que la foto enviada por el usuario corresponde con su imagen asignada en la empresa.

ClockInReportsAPI: API para la generación de informes.

- Establece la comunicación entre la aplicación web y la base de datos para el tratamiento de los datos de los informes en base a los filtros recibidos del usuario.

Estas APIs, a excepción de la API destinada a la generación de informes por motivos que se explicarán más adelante en este informe, están integradas en la nube de AWS a través del servicio API Gateway, que se encarga de gestionar las peticiones HTTP de las aplicaciones. El código de estas APIs está desplegado en AWS Lambda.

Un ejemplo del flujo de nuestro sistema "serverless" sería el siguiente:

1. El usuario realiza una acción en la web (Por ejemplo: Crear un rol). Esta acción implica una llamada a nuestro servicio web, con el *endpoint* correspondiente para esa acción.

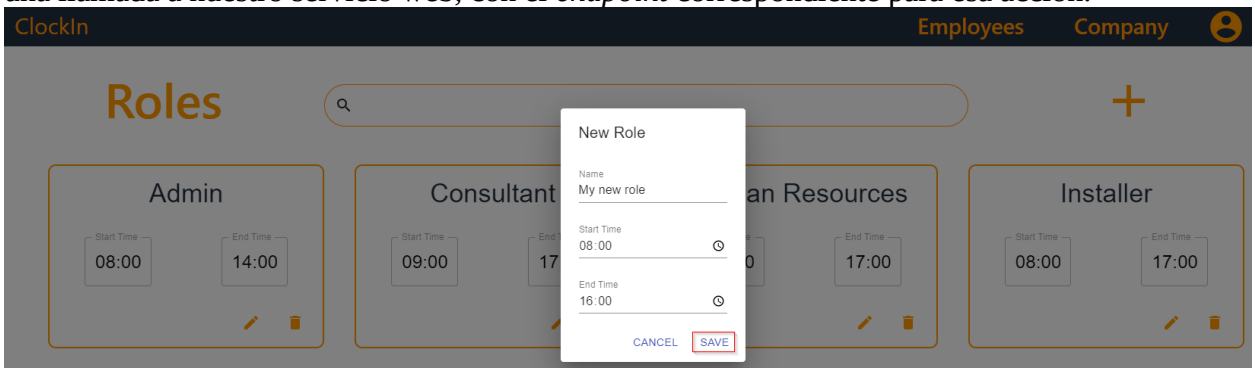

Figura 5.23 Creación de un rol en la web

```
export async function createRole(role, companyId) {
    const requestOptions = {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(role),
    };
    const response = await fetch(
        `${API_URL}/Role?companyId=${companyId}`,
        requestOptions
    );
    const data = await response.json();
    return data;
}
```

2. El servicio web (API Gateway) desencadena la llamada a una función *lambda* hacia el controlador y el método correspondiente de la función en base a la url de la petición.

```
[HttpPost]
public bool CreateRole(Role role, int companyId)
{
    try
    {
        return _roleService.CreateRole(role, companyId,
_connectionString);
    }
```

```
    catch (Exception ex)
    {

LambdaLogger.Log($"RoleController::CreateRole::Exception={ex}");
        return false;
    }
}
```

3. La función *lambda* ejecuta la lógica correspondiente. En el ejemplo que nos ocupa, realizar una conexión con la base de datos para insertar el rol en la tabla “role”.

```
public bool CreateRole(Role role, int companyId, string
connectionString)
{
    try
    {
        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand("usp_CreateRole",
connection);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@company_id",
companyId);
            command.Parameters.AddWithValue("@name", role.Name);
            command.Parameters.AddWithValue("@start_time",
Utils.ConvertToTimeSpan(role.StartTime));
            command.Parameters.AddWithValue("@end_time",
Utils.ConvertToTimeSpan(role.EndTime));
            return command.ExecuteNonQuery() > 0;
        }
    }
    catch (Exception ex)
    {

LambdaLogger.Log($"RoleService::CreateRole::Exception={ex}");
        return false;
    }
}
```

4. El servicio web retorna al usuario (*response*) la respuesta obtenida de la función *lambda.*

5.3 Implementación y despliegue

En este apartado se describe el desarrollo llevado a cabo para la implementación de los principales casos de uso de nuestro sistema, así como la estructura de este.

5.3.1 Estructura del sistema

En este apartado vamos a desglosar la estructura de todos los componentes que conforman nuestro sistema, esto es:

- Estructura de directorios de la aplicación web
- Estructura de directorios de la aplicación móvil
- Estructura de la API de la aplicación web
- Estructura de la API de la aplicación móvil
- Estructura de la API de generación de informes
- Estructura de la base de datos
- Estructura de los servicios en la nube (AWS)

Estructura de la aplicación web

Directorio	Funcionalidad
/amplify	Contiene ficheros *.json* de configuración de AWS Amplify
/node_modules	Contiene todas las dependencias de nuestro proyecto
/public	Contiene el punto de entrada de la aplicación "*index.html*" así como el icono del header "*favicon.ico*"
/src	Contiene todos los ficheros JS y CSS empleados en la aplicación
/src/api	Contiene un archivo *.js* con las funciones que gestionan las llamadas a la API
/src/components	Contiene componentes usados en diferentes páginas dentro de la web
/src/helpers	Contiene ficheros *.js* con funciones de ayuda en la aplicación como el formateo de fechas, entre otras
/src/images	Contiene imágenes empleadas en diferentes puntos de la web
/src/pages	Contiene las páginas que conforman la web

Tabla 5.23 Estructura de la aplicación web

Estructura de la aplicación móvil

Directorio	Funcionalidad
/.expo /.expo-shared	Contiene ficheros de configuración de Expo

/amplify	Contiene ficheros *.json* de configuración de AWS Amplify
/node_modules	Contiene todas las dependencias de nuestro proyecto
/assets	Contiene imágenes usadas en el proyecto, como el icono del header de la aplicación o el icono global de la aplicación
/src	Contiene todos los ficheros JS y CSS empleados en la aplicación
/src/api	Contiene un archivo *.js* con las funciones que gestionan las llamadas a la API
/src/components	Contiene componentes usados en diferentes páginas dentro de la aplicación
/src/helpers	Contiene ficheros *.js* con funciones de ayuda en la aplicación como el formateo de fechas, entre otras
/src/routes	Contiene las rutas de la aplicación, así como el fichero *tabNavigator.js*, que establece el diseño y la lógica del menú de la aplicación
/src/pages	Contiene las pantallas que conforman la aplicación

Tabla 5.24 Estructura de la aplicación móvil

Estructura de la API de la aplicación web/móvil

Dado que las APIs de ambas aplicaciones han sido implementadas con el servicio de AWS API Gateway mediante la integración de funciones de Amazon Lambda, se ha decidido condensar el análisis de la estructura de ambas APIs en este único apartado. La estructura de directorios en ambas APIs es idéntica, estructurada en la siguiente tabla:

Directorio	Funcionalidad
/Controllers	Gestiona las peticiones HTTP del cliente y devuelve la respuesta al cliente
/Helpers	Contiene clases con funciones y modelos de ayuda (Ejemplo: Conversión de imágenes de *array* de *bytes* a *string* en *base64*)
/Models	Contiene las clases que representan el modelo de nuestra aplicación
/Services	Contiene clases que implementan la lógica de los servicios web, incluyendo la comunicación con otros servicios de la nube y con la base de datos.

Tabla 5.25 Estructura de la API (web y móvil)

Estructura de la API de generación de informes

Previamente en este informe, indicamos que esta API, a diferencia de los servicios web desplegados para las aplicaciones web y móvil, no estaba implementada mediante la integración API Gateway + Amazon Lambda.

El motivo es que, para la generación de informes, utilizamos la herramienta Crystal Reports, desarrollada por SAP. Esta herramienta no tiene soporte para el framework .NET Core, que es a su vez el único framework del entorno .NET soportado por Amazon Lambda. Debido a estos problemas de compatibilidad, se decidió hacer uso de otro servicio que nos proporciona AWS: Elastic Beanstalk. Se trata de un servicio que nos permite desplegar una aplicación sin tener que preocuparnos por la infraestructura que lo soporta. Algunas de las aplicaciones que nos permite desplegar son las siguientes:

- Aplicaciones web en java, ruby o php, entre otros, desplegadas en un servidor Apache.
- Aplicaciones NodeJS.
- **Aplicaciones en .NET en IIS Server.**

Es esta última aplicación resaltada la que nos interesa, puesto que IIS Server (servidor web para Windows) sí tiene compatibilidad para aplicaciones web desarrolladas en .NET Framework. De modo que de esta manera también podemos tener nuestra API de informes desplegada en la nube, cumpliendo así el objetivo de que todo el backend de nuestro sistema esté desplegado en AWS.

Directorio	Funcionalidad
/Controllers	Gestiona las peticiones HTTP del cliente y devuelve la respuesta al cliente.
/Utils	Contiene clases con funciones y modelos de ayuda (Ejemplo: Conversión de imágenes de array de bytes a string en base64)
/Models	Contiene las clases que representan el modelo de nuestra aplicación
/Datasets	Contiene las tablas con los datos a representar en los informes
/Reports	Contiene los ficheros "*.rpt*" que representan el diseño de los informes

Tabla 5.26 Estructura de la API de generación de informes

Estructura de la base de datos

Para la base de datos del sistema, se ha optado por el sistema gestor de base de datos (SGBD) Microsoft SQL Server (MSSQL). Este servidor de base de datos ha sido desplegado en una instancia de AWS RDS (Relational Database Server), con dos bases de datos en creadas en esta instancia:

ClockInDev: Base de datos de desarrollo empleada para pruebas.

ClockIn: Base de datos de producción.

Durante el desarrollo de la base de datos, se han empleado los siguientes elementos de esta:

Tablas: Definen el modelo de nuestra aplicación, con sus campos y relaciones. La estructura de estas con sus relaciones se muestra en la Figura 5.14 - Diagrama entidad-relación (ER). A continuación, se muestra una imagen con la estructura de una tabla de nuestro sistema, esto es, sus campos, claves e índices.

Campos (Columns): son los atributos de las tablas, tienen un nombre y un tipo. Pueden o no ser clave de la tabla, así como tener la propiedad de ser valor nulo.

Claves (Keys): identifican registros únicos en la tabla (clave primaria PK) o relaciones con otras tablas de la base de datos (clave foránea FK).

Índices (Indexes): estructuras de datos que nos permiten agilizar las búsquedas. Por defecto, las tablas generan un índice por clave primaria al ser creadas.

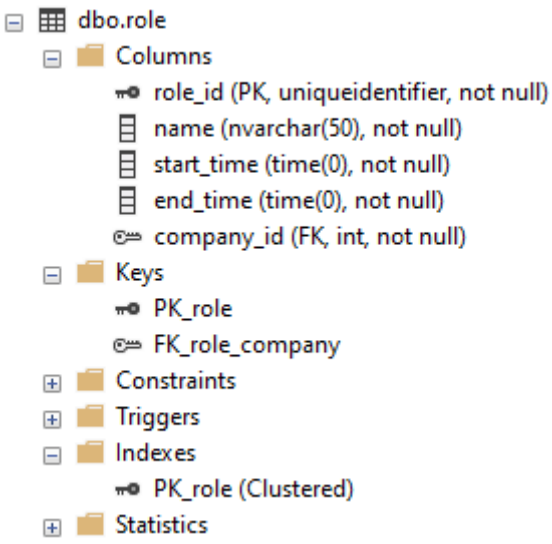

Figura 5.24 Estructura tabla "role"

Procedimientos almacenados: Funciones en lenguaje *SQL*. Contienen parámetros de entrada y devuelven un valor entero (nº de registros modificados) o el resultado de una consulta (*query*). A continuación, se muestra una imagen con el código de un procedimiento almacenado de usuario de nuestro sistema.

```
ALTER PROCEDURE [dbo].[usp_CreateRole]
    @company_id INT,
    @name NVARCHAR(50),
    @start_time TIME(0),
    @end_time TIME(0)
AS
BEGIN
    INSERT INTO [role] (company_id,[name],start_time,end_time)
    VALUES (@company_id,@name,@start_time,@end_time)
END
```

Funciones escalares: Funciones en lenguaje *SQL* invocadas desde procedimientos almacenados. Realizan funciones muy concretas en nuestro sistema. A continuación, se muestra una imagen con la función "Haversine" en *SQL*, que usamos para calcular la distancia entre dos coordenadas.

```
ALTER FUNCTION [dbo].[udf_Haversine](@lat1 float, @long1 float, @lat2
float, @long2 float) RETURNS float
BEGIN
    DECLARE @dlon float, @dlat float,
            @rlat1 float, @rlat2 float, @rlong1 float, @rlong2 float,
            @a float, @c float, @R float, @d float, @DtoR float

    SELECT
        @DtoR = PI() / 180,
        @R = 6371
```

```
    SELECT
        @rlat1 = @lat1 * @DtoR,
        @rlong1 = @long1 * @DtoR,
        @rlat2 = @lat2 * @DtoR,
        @rlong2 = @long2 * @DtoR

    SELECT
        @dlat = @rlat1 - @rlat2,
        @dlon = @rlong1 - @rlong2

    SELECT @a = SIN(@dlat / 2) * SIN(@dlat / 2) +
                SIN(@dlon / 2) * SIN(@dlon / 2) * COS(@rlat2) *
COS(@rlat1)

    SELECT @c = 2 * atn2(sqrt(@a), sqrt(1 - @a))
    SELECT @d = @R * @c

RETURN @d
END
```

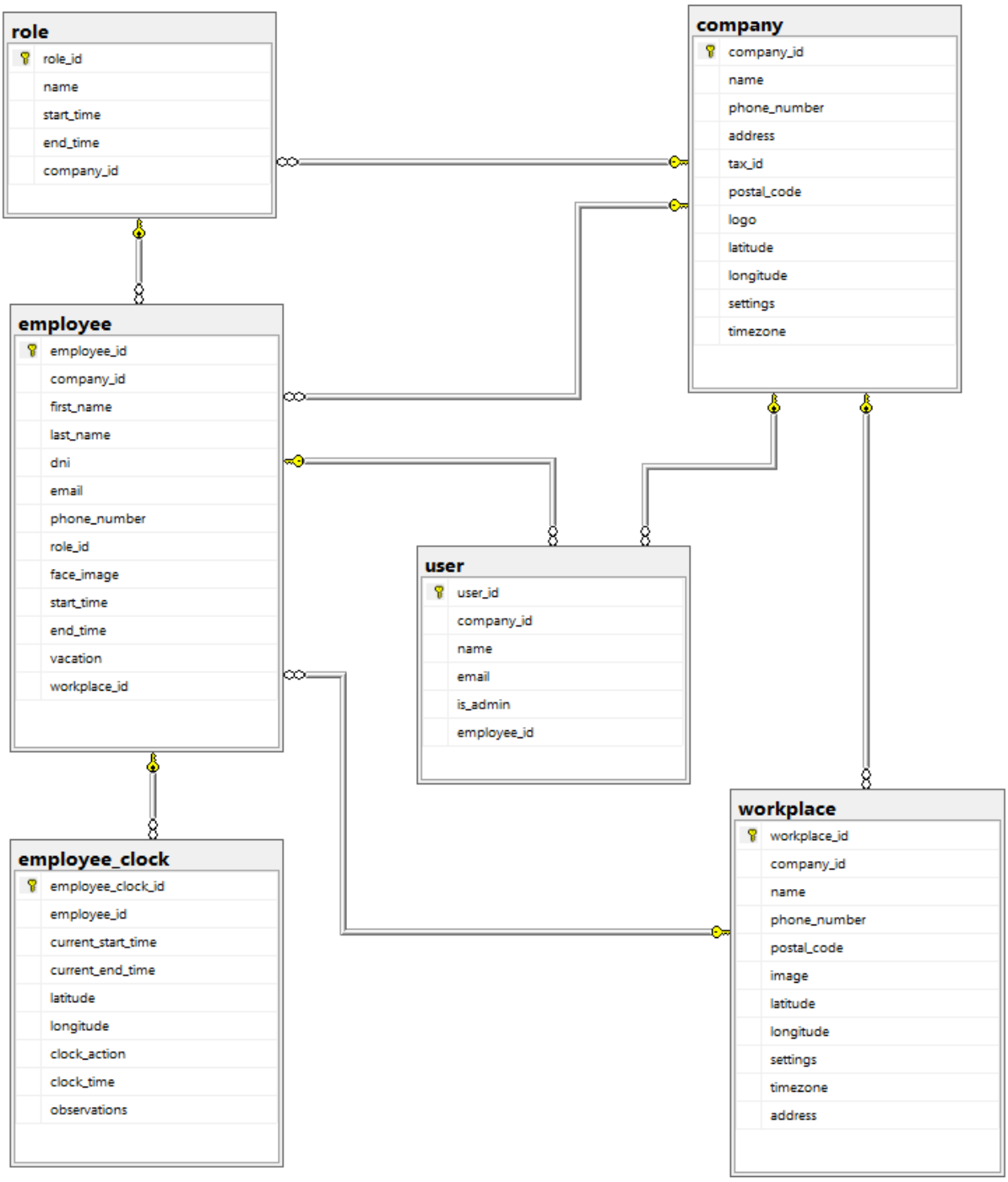

Figura 5.25 Diagrama entidad-relación (ER)

Estructura de los servicios en la nube (AWS)

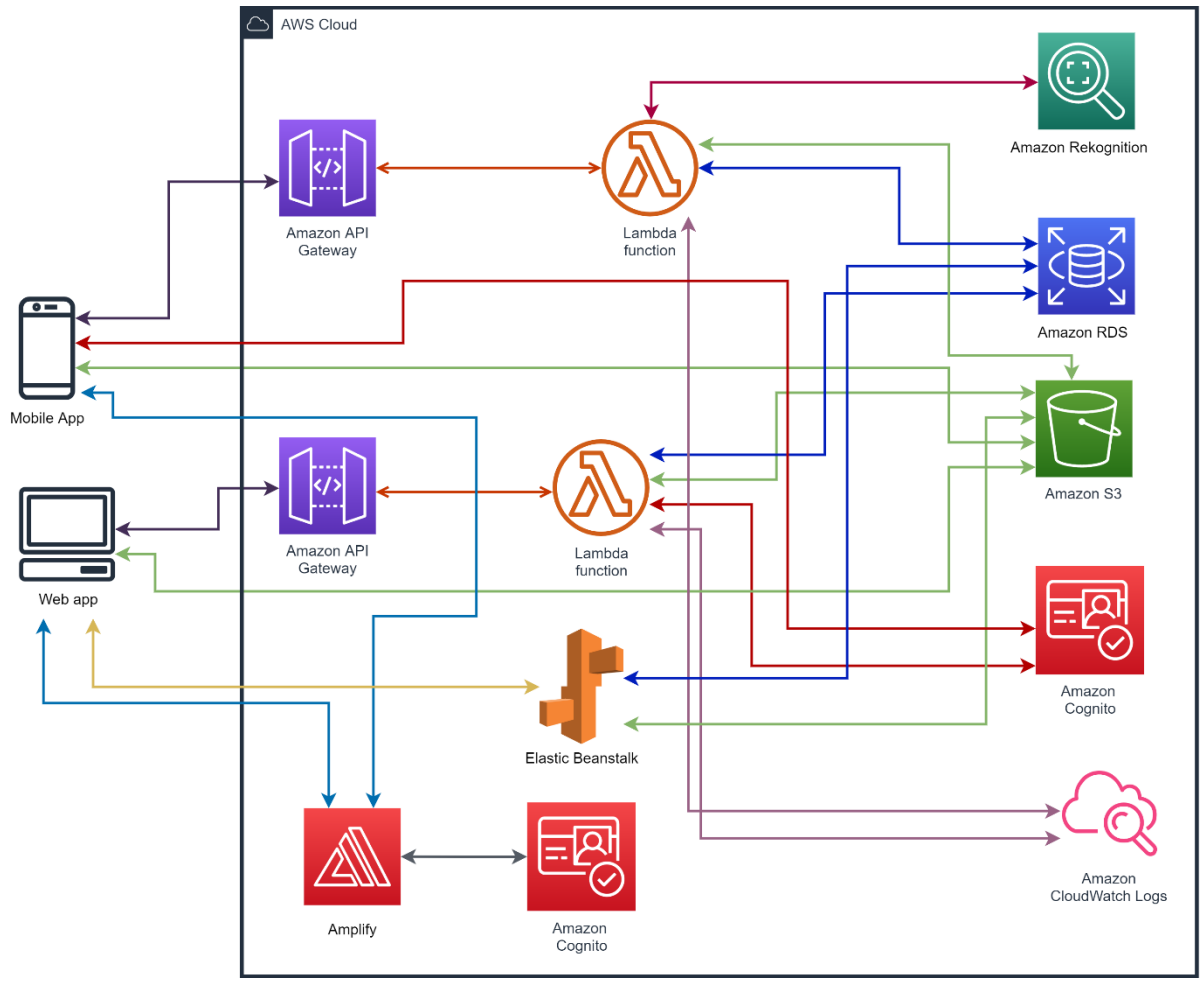

Figura 5.26 Esquema arquitectura AWS en nuestro sistema

A continuación, vamos a explicar todos los enlaces del esquema mostrado en la Figura 3.29 "Esquema arquitectura AWS en nuestro sistema". Nos centraremos en cómo interaccionan estos servicios entre sí y en qué aportan a nuestro sistema, dejando las definiciones y características de cada servicio para el apartado de recursos.

Enlace App móvil - Amplify - Cognito: La aplicación móvil ha sido desplegada en AWS Amplify. La razón fundamental de esta decisión es que Amplify cuenta con una fácil integración con el servicio de autenticación de usuarios AWS Cognito, de modo que las acciones relacionadas con la autenticación de usuarios se llevan a cabo desde la app cliente mediante el uso del *SDK* de Amplify, sin necesidad de hacer llamadas adicionales a un servicio web externo a la aplicación.

Enlace App web - Amplify – Cognito: La web también ha sido configurada en AWS Amplify. En este caso, al motivo del uso de AWS Cognito como proveedor de servicios de autenticación, se le suman la integración con Git y el *hosting* de la aplicación. De modo que para desplegar o actualizar nuestra web no tenemos más que enlazar una rama de nuestro repositorio git a la aplicación en Amplify, y hacer una subida a dicha rama en remoto.

Enlaces App móvil/App web - API Gateway: Ambas aplicaciones cuentan con su propia API para la comunicación con la base de datos y con otros recursos de AWS.

Enlace App móvil - Amazon S3*:* Esta interacción se debe a la subida a Amazon S3 de la foto del empleado en el fichaje.

Enlace App web - Amazon S3: Para la subida y descarga de todas las imágenes entre web y Amazon S3. (empleados, logo de empresa, lugares de trabajo…)

Enlace App web - Amazon ElasticBeanstalk: Para la generación de informes en la web, llamando a la API de informes alojada en AWS ElasticBeanstalk.

Enlaces API Gateway - Amazon Lambda: Cuando API Gateway recibe las peticiones HTTP desde la app cliente, esta actúa como desencadenador invocando una función lambda para que gestione la petición recibida. La creación, actualización, obtención y borrado de entidades (roles, empleados, etc.) así como la lógica asociada al proceso de fichaje mediante reconocimiento facial se llevan a cabo en estas funciones lambda.

Enlaces Amazon Lambda - Amazon S3: Las funciones lambda de nuestro sistema se encargan de generar lo que se conoce como *PresignedUrl* (URL prefirmada) de Amazon S3, que funciona a modo de puntero para la subida o descarga de una imagen. Esto es debido a que Amazon Lambda tiene una limitación en cuanto a los datos que esta recibe de 6 MB. De modo que, para evitar posibles peticiones a la API que sobrepasen este límite, el proceso que se lleva a cabo es el de generar en base a la información recibida por la API una URL para que el usuario pueda subir o descargar una imagen de Amazon S3 directamente desde la aplicación cliente. Esta información necesaria para generar la URL no es más que el nombre del repositorio o *bucket* en S3, la *key* o nombre de la imagen, el tipo de petición de la URL (subida - PUT, descarga - GET) y el tiempo de expiración. A continuación, se muestra el código encargado de generar estas URLs.

```
public static string GeneratePreSignedUrl(string fileName, string
contentType, string bucket, bool isUpload = false)
{
    try
    {
        string accessKey =
Environment.GetEnvironmentVariable("AccessKey");
        string secretKey =
Environment.GetEnvironmentVariable("SecretKey");

        AmazonS3Client s3Client = new AmazonS3Client(accessKey,
secretKey, RegionEndpoint.EUWest1);

        GetPreSignedUrlRequest getPreSignedUrlRequest = new
GetPreSignedUrlRequest
        {
            BucketName = bucket,
            Key = fileName,
            Verb = isUpload ? HttpVerb.PUT : HttpVerb.GET,
            Expires = DateTime.UtcNow.AddMinutes(15),
        };

        string url = s3Client.GetPreSignedURL(getPreSignedUrlRequest);
        return url;
    }
    catch (Exception ex)
```

```
    {
        LambdaLogger.Log($"GeneratePreSignedUrl::Exception={ex}");
        return string.Empty;
    }
}
```

Enlaces Amazon Lambda - Amazon RDS: Este enlace representa la conexión entre los servicios web implementados en funciones lambda y la base de datos.

Enlace Amazon Lambda - Amazon Rekognition: La lógica del proceso de reconocimiento facial durante el fichaje se lleva a cabo mediante llamadas al servicio Amazon Rekognition desde la función lambda.

Enlace Amazon Lambda - Cognito: En el proceso de creación de un empleado, no sólo se crea al empleado la base de datos, sino que se le crea un usuario en el grupo de usuarios de Cognito de la aplicación móvil. De modo que al usuario le llega a su email una contraseña temporal que deberá modificar en su primer inicio de sesión en la app móvil.

Enlace Amazon Lambda - Amazon CloudWatch Logs: Desde las funciones lambda se generan mensajes de log en CloudWatch para pruebas y control de excepciones.

5.3.2 Creación de la base de datos

El primer desarrollo llevado a cabo fue el de la creación de la base de datos para nuestras aplicaciones. Para ello, se ha hecho uso del servicio en la nube de AWS RDS (Relational Database Server) para la creación de un servidor de SQL Server en la nube a través de la consola de administración de AWS.

El siguiente paso tras la creación del servidor SQL Server es conectarnos a la base de datos a través de un entorno de desarrollo. En este caso, hemos usado Microsoft SQL Server Management Studio. Una vez conectados a nuestro servidor, se crean dos bases de datos. Una base de datos de desarrollo (ClockInDev) y una base de datos de producción (ClockIn).

En este momento se crean las tablas fundamentales del sistema (empleados, empresa y usuarios), así como los procedimientos almacenados que serán ejecutados para realizar inserciones, actualizaciones, borrados y lecturas.

Cabe destacar que tanto la aplicación web como la aplicación móvil comparten las mismas bases de datos.

5.3.3 Creación y despliegue de la aplicación web

La siguiente tarea implementada es la de la creación de la web, desarrollada a partir de la librería ReactJS, y que funciona a modo de panel de administrador para los administradores de las empresas.

Para ello, nos hemos apoyado en el servicio Amplify de AWS. Se trata de un servicio que nos proporciona una serie de herramientas tanto frontend como backend para una rápida creación y despliegue de nuestra web, gracias a tres elementos fundamentales:

- *Hosting* de la web.
- Integración continua con un repositorio git.
- Integración con el servicio de AWS Cognito para la autenticación de los usuarios.

Los pasos realizados para lograr este objetivo de creación y despliegue de la web son los siguientes:

1. Creación del proyecto que conformará el *frontend* de nuestra web en React JS desde la consola con el comando "***npx create-react-app ClockIn"***.
2. Con la aplicación ya creada en un directorio local de nuestro equipo, el siguiente paso es integrar la app con Amplify. Para ello, han de seguirse una serie de instrucciones descritas en la documentación de Amplify en AWS.
    a. Instalación del CLI (*Command Line Interface*) de Amplify desde la consola con el comando "**npm install -g @aws-amplify/cli**".
    b. Configuración de Amplify en la que especificamos las claves de nuestra cuenta de AWS, así como la región donde vamos a publicar nuestro proyecto, a través del comando "**amplify configure**".
    c. Integración de Amplify con nuestra app con el comando "**amplify init**", seguido de un procedimiento de configuración que nos va apareciendo en consola para configurar nuestro entorno y en el que indicamos el nombre del proyecto en Amplify así como el framework empleado, entre otras opciones. Tras este paso, ya queda inicializado nuestro proyecto en Amplify.
    d. Integración de Amplify con el servicio de autenticación Cognito mediante el comando "**amplify add auth**", seguido de una breve configuración por pasos en la que se indica el modo de autenticación (email, nombre de usuario, etc.) de los usuarios, entre otras.
    e. Publicación de la configuración del *backend* de Amplify (al que hemos agregado un servicio de autenticación en el paso anterior) mediante el comando "**amplify push**".
    f. Finalmente, se integra el *frontend* de Amplify con el repositorio git de nuestro proyecto alojado en la plataforma GitHub. Este paso se ha llevado a cabo desde la consola de AWS.
3. Una vez ya hemos desplegada la aplicación en Amplify y añadido el servicio de autenticación de Cognito, sólo nos falta integrar el *frontend* de nuestro proyecto en Amplify con un repositorio git, de modo que la integración sea continua, permitiéndonos publicar nuestra aplicación en Amplify mediante subidas al repositorio remoto de git. Este último paso se ha realizado desde la consola de administración de AWS.

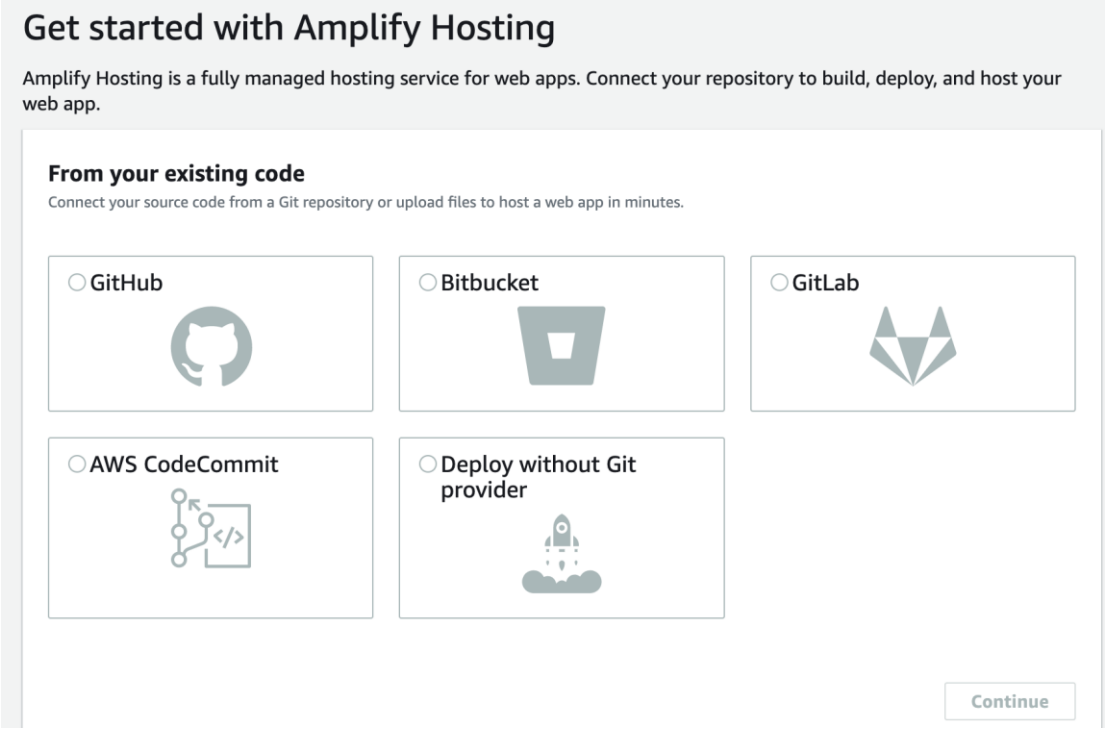

Figura 5.27 Integración del frontend de Amplify con un repositorio git

Tras el desarrollo de esta tarea de creación y despliegue de la web, ya contamos con una versión funcional del producto principal de nuestro sistema, cumpliendo así con la premisa fundamental de la metodología incremental de crear un producto funcional en las fases iniciales del proyecto.

5.3.4 Creación y despliegue de la aplicación móvil

Con la web de uso dedicado a los administradores ya creada, el siguiente paso era el de elaborar una aplicación móvil muy sencilla para los empleados. Esta app móvil estaría destinada a realizar los registros de jornada, así como para ver información básica como los períodos vacacionales asignados o un calendario con el histórico de registros de jornada.

Para ello, nos hemos apoyado del framework Expo de React Native, que nos permite la creación y publicación de una app de React Native rápidamente. Los pasos llevados a cabo para la creación de la aplicación móvil son los siguientes:

1. Instalación de la CLI de Expo mediante el comando "**npm install -g expo-cli**", necesaria para ejecutar los comandos de Expo que nos permitirán generar la app.
2. Creación de la aplicación en local con el comando "**npx create-expo-app my-app**".

Con estos dos sencillos comandos ya tenemos creada la aplicación móvil con react native. Cabe destacar que esta aplicación no ha sido publicada en ninguna de las tiendas oficiales de aplicaciones móviles como Google Play o App Store.

Para generar un fichero de instalación de android (.apk), basta con ejecutar el comando de consola "expo build:android". Tras esto seleccionamos la opción apk y se nos muestra una url que nos redirige a la web de expo con el proceso de compilado de la aplicación en curso. Cuando este ha finalizado, ya podemos descargar la apk e instalarla en cualquier dispositivo android.

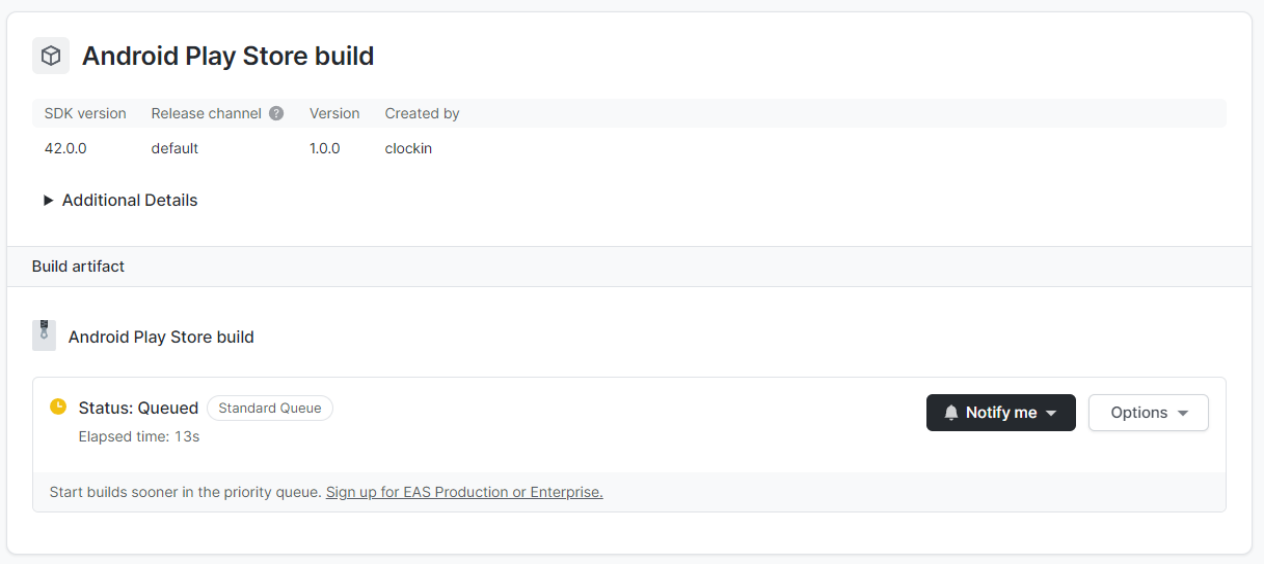

Figura 5.28 Generación apk a través de expo

El último paso en la configuración de nuestra aplicación móvil previo al desarrollo de esta consiste en integrar la aplicación con Amplify, al igual que hicimos con la aplicación web. El proceso de integración con Amplify es exactamente igual que el descrito en el apartado 4.2.

5.3.5 Creación de las APIs de las aplicaciones

Con la web ya desplegada y funcional, el siguiente paso es el de crear el backend necesario para gestionar las entidades de la web y dotarla de mayor funcionalidad.

Para ello, hemos usado la plantilla del conjunto de herramientas de AWS para Visual Studio "AWS Serverless Application". [21]

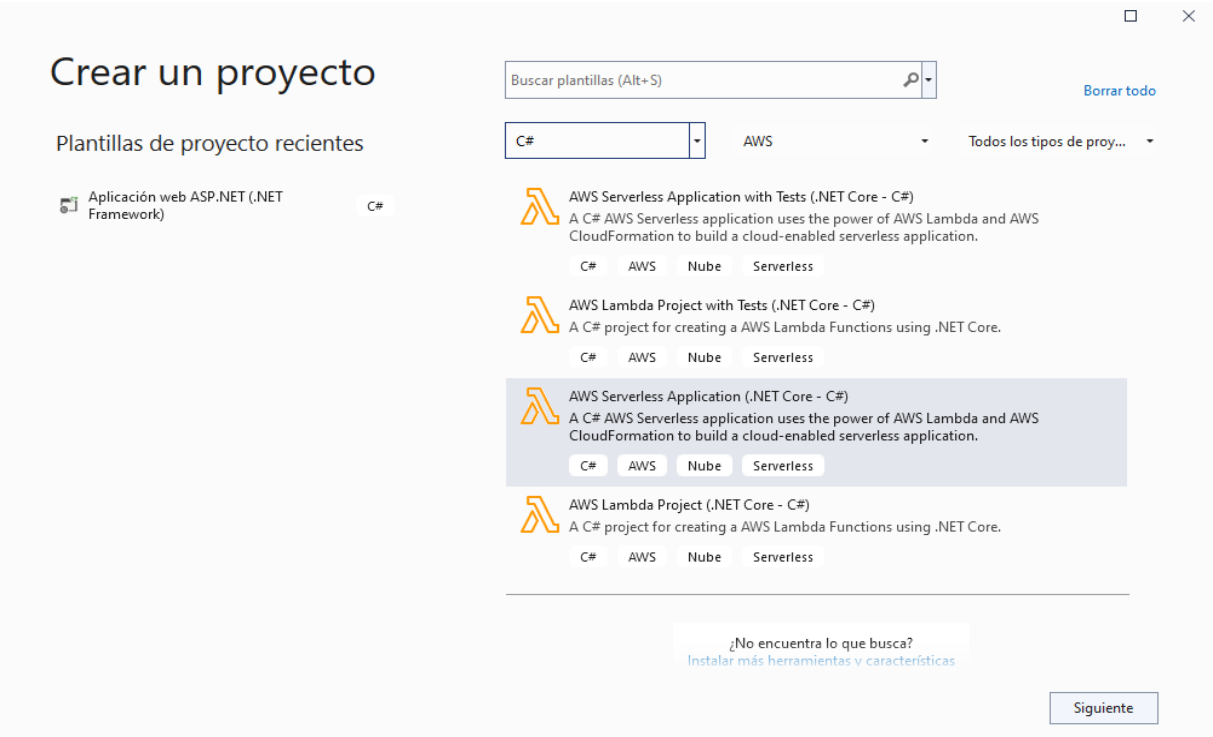

Figura 5.29 Creación AWS Serverless Application (.NET Core- C#) en Visual Studio 2022

Este *template* nos permite generar una instancia de API Gateway de AWS integrada con una función lambda. De modo que en la primera publicación que hagamos de este proyecto a AWS, se creará la arquitectura en la nube de esta aplicación, esto es, una función lambda con una instancia de API Gateway de desencadenador. De modo que, al invocarse un método de la API, se ejecutará el controlador correspondiente en la función lambda.

Adicionalmente, este proyecto cuenta con un fichero de configuración "Startup.cs" en la raíz del proyecto, en el que habilitamos el CORS (Cross-origin resource sharing) para las peticiones desde las aplicaciones cliente, establecemos la configuración de la herramienta swagger para la validación de nuestras APIs y establecemos las dependencias necesarias.

Esta clase cuenta con un método "ConfigureServices" que es ejecutado en cada llamada a la función lambda, y que recibe como parámetro la interfaz "IServiceCollection". Esta interfaz representa el contenedor de dependencias que nos proporciona el proyecto, de modo que sobre esta colección agregamos los servicios que vayamos a utilizar en la lambda y que serán invocados por los controladores (puntos de acceso a las APIs). Estos servicios representan la lógica de la API y son los que se encargan de realizar conexiones con la base de datos y de invocar a otros servicios de AWS.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy(name: MyAllowSpecificOrigins,
            builder =>
            {
             builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod();
            });
    });

    services.AddControllers();
    services.AddSingleton<ICompanyService,CompanyService>();
    services.AddSingleton<IUserService, UserService>();
    services.AddSingleton<IRoleService, RoleService>();
    services.AddSingleton<IEmployeeService, EmployeeService>();
    services.AddSingleton<IAuthService, AuthService>();
    services.AddSingleton<IWorkplaceService, WorkplaceService>();

    services.AddSwaggerGen(swagger =>
    {
        swagger.SwaggerDoc("v1", new OpenApiInfo { Title = "ClockIn API"
        });
        var xmlFile =
        $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
        var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
        swagger.IncludeXmlComments(xmlPath);
    });
}
```

5.3.6 Desarrollo del software de gestión de entidades del sistema

Con la infraestructura de AWS desplegada y nuestras aplicaciones ya creadas, el siguiente paso es implementar los casos de uso que aún no han sido desarrollados, desarrollando conjuntamente el *frontend* y el *backend* de nuestro sistema.

Puesto que contamos con una amplia variedad de entidades (empresas, empleados, roles, lugares de trabajo, informes, usuarios…), vamos a centrarnos en este apartado en los que consideramos los casos de uso más representativos de nuestro proyecto, en el orden en que estos han sido implementados.

Creación de empresa

El primer punto en el desarrollo del proyecto era implementar la creación de empresas, puesto que es la principal entidad de nuestro sistema, en torno a la que se gestionan el resto de las entidades (empleados, roles, etc.).

El primer paso consiste en realizar un formulario en la web incluyendo la información de la empresa que vamos a almacenar en nuestra base de datos. Esto es, nombre, CIF, dirección, localización, email y contraseña del usuario administrador, teléfono, código postal y la posibilidad de establecer un radio de distancia para el control del fichaje con geolocalización.

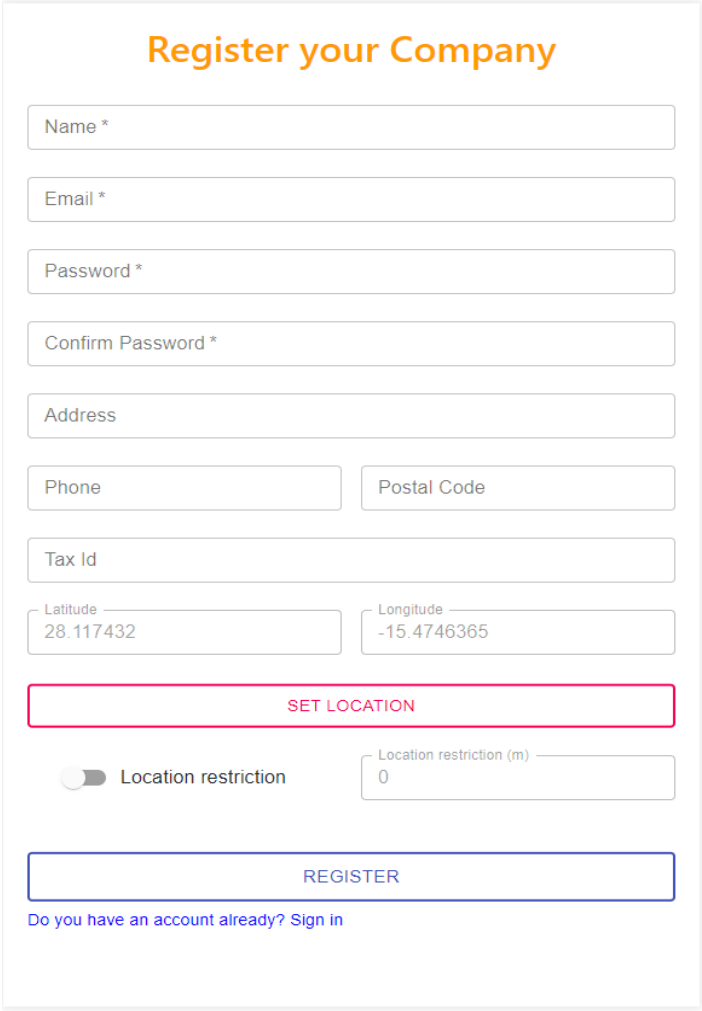

Figura 5.30 Formulario de registro de una empresa

Cuando el usuario hace click en registrar, se realiza una llamada a Cognito a través del componente Auth que nos proporciona Amplify para registrar el usuario en el grupo de usuarios de Cognito asociado a nuestra aplicación.

```
async function handleSubmit() {
    try {
```

```
      setOpen(true);
      const formValidationSucceded = validateForm();
      if (formValidationSucceded) {
        const adminUserSignedUpSuccesfully = await Auth.signUp(
          companyInfo.email,
          companyInfo.confirmPassword
        );
        if (adminUserSignedUpSuccesfully) {
          setCompanyInfo((prevState) => ({
            ...prevState,
            userId: adminUserSignedUpSuccesfully.userSub,
            userName: companyInfo.email,
          }));
        }
        setOpen(false);
      } else {
        setOpen(false);
      }
    } catch (error) {
      console.log("Register company error: ", error);
      setOpen(false);
    }
}
```

Una vez el usuario ha sido registrado en Cognito, se realiza una llamada a nuestro servicio web, que tratará la información creando la empresa en base de datos, así como el identificador del usuario en la tabla de usuarios.

El siguiente paso consiste en implementar el código de backend alojado en las funciones lambda de la API para realizar las conexiones a la base de datos y a Cognito. Para ello, lo primero es crear el modelo equivalente a la petición recibida.

```
public class CompanyRequest
{
    public int? Id { get; set; }
    public string Name { get; set; }
    public string PhoneNumber { get; set; }
    public string Address { get; set; }
    public string TaxId { get; set; }
    public string PostalCode { get; set; }
    public double? Latitude { get; set; }
    public double? Longitude { get; set; }
    public string Logo { get; set; }
    public double[] Position { get; set; }
    public List<CompanySettings> Settings { get; set; }
    public Guid UserId { get; set; }
    public string UserName { get; set; }
}
```

Con el modelo creado, el siguiente paso consiste en crear el punto de entrada de la petición HTTP de la aplicación cliente en nuestra API. Esto se consigue creando un controlador con el método correspondiente, en este caso “Company” con un método POST.

```
[Route("api/[controller]")]
[ApiController]
public class CompanyController : ControllerBase
{
    private readonly ICompanyService _companyService;
    private readonly IConfiguration _configuration;
    private readonly string _connectionString;
```

```
    public CompanyController(ICompanyService companyService,
IConfiguration configuration)
    {
        _companyService = companyService;
        _configuration = configuration;
        _connectionString =
Environment.GetEnvironmentVariable("ClockInDB");
    }

    [HttpPost]
    public bool CreateCompany(CompanyRequest companyRequest)
    {
        try
        {
            return _companyService.CreateCompany(companyRequest,
_connectionString);
        }
        catch (Exception ex)
        {

LambdaLogger.Log($"CompanyController::CreateCompany::Exception={ex}");
            return false;
        }
    }
```

La conexión con la base de datos no se implementa en el controlador, sino en una capa intermedia o capa de datos, a la que hemos llamado “Services”,

A continuación, se muestra el código de nuestro servicio asociado a la entidad empresa encargado de realizar una conexión con la base de datos para guardar la información de la empresa a registrar, así como el código del procedimiento almacenado (SQL) invocado desde este servicio.

```
public bool CreateCompany(CompanyRequest companyRequest, string
connectionString)
{
    try
    {
        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand("usp_CreateCompany",
connection);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@name",
companyRequest.Name);
            command.Parameters.AddWithValue("@phone_number",
companyRequest.PhoneNumber);
            command.Parameters.AddWithValue("@address",
companyRequest.Address);
            command.Parameters.AddWithValue("@tax_id",
companyRequest.TaxId);
            command.Parameters.AddWithValue("@postal_code",
companyRequest.PostalCode);
            command.Parameters.AddWithValue("@latitude",
companyRequest.Position[0]);
            command.Parameters.AddWithValue("@longitude",
companyRequest.Position[1]);
```

```
            command.Parameters.AddWithValue("@settings",
JsonConvert.SerializeObject(companyRequest.Settings));
            command.Parameters.AddWithValue("@user_id",
companyRequest.UserId);
            command.Parameters.AddWithValue("@user_name",
companyRequest.UserName);
            return command.ExecuteNonQuery() > 0;
        }
    }
    catch (Exception ex)
    {

LambdaLogger.Log($"CompanyService::CreateCompany::Exception={ex}");
        return false;
    }
}
```

```
ALTER PROCEDURE [dbo].[usp_CreateCompany]
    @name NVARCHAR(50),
    @phone_number NVARCHAR(50) = NULL,
    @address NVARCHAR(50) = NULL,
    @tax_id NVARCHAR(50) = NULL,
    @postal_code NVARCHAR(10) = NULL,
    @latitude FLOAT = NULL,
    @longitude FLOAT = NULL,
    @settings NVARCHAR(MAX) = NULL,
    @logo NVARCHAR(MAX) = NULL,
    @user_id UNIQUEIDENTIFIER,
    @user_name NVARCHAR(50) = NULL
AS
BEGIN
    INSERT INTO
company([name],phone_number,[address],latitude,longitude,tax_id,postal_co
de,logo,settings)
    VALUES
(@name,@phone_number,@address,@latitude,@longitude,@tax_id,@postal_code,@
logo,@settings)

    INSERT INTO [user]([user_id],company_id,[name],email,is_admin,
employee_id)
    VALUES (@user_id,SCOPE_IDENTITY(),@user_name,@user_name,1,NULL)
END
```

Creación de empleado

Con el desarrollo de creación de empresas ya finalizado, el siguiente paso lógico consiste en crear empleados para la empresa. El proceso llevado a cabo es muy similar al de la creación de empresas.

En primer lugar, elaboramos el formulario de creación del empleado en la web. Este formulario debe incluir su nombre y apellidos, email, dni, teléfono, horario de trabajo o rol asignado y una foto con su rostro.

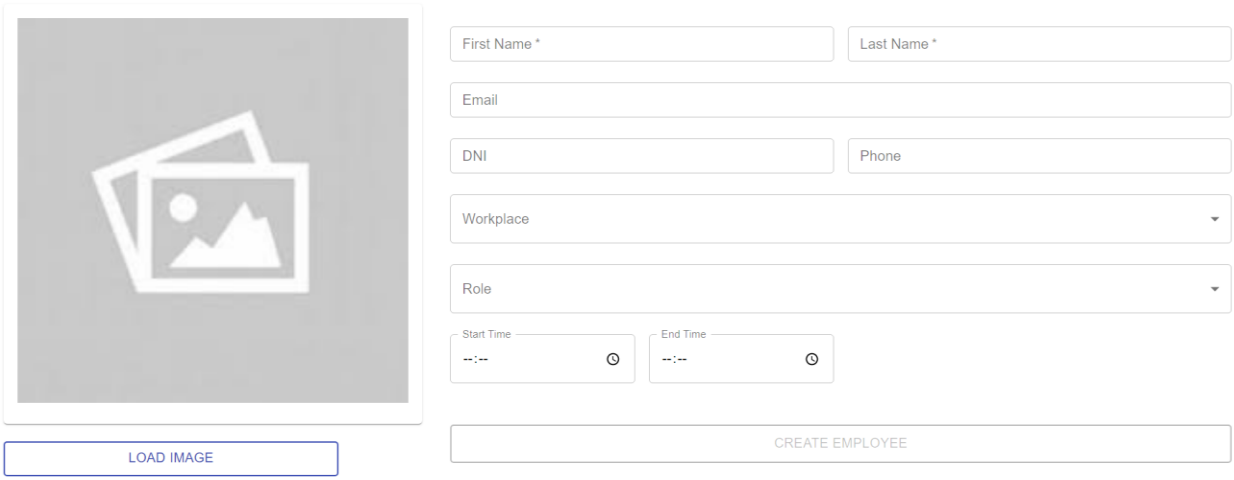

Figura 5.31 Formulario de creación de un empleado

Cuando el usuario hace click en crear empleado, se realiza una llamada a la API enviando la información del empleado, a excepción de la imagen, con el objetivo de reducir el tamaño de la petición, que está limitado a 6MB en AWS Lambda. Esta llamada a la API retorna una URL de S3 a modo de puntero sobre la que la aplicación cliente realiza una petición HTTP para la subida de la imagen.

```
export async function uploadEmployeeImageToS3(S3presignedUrl, employee) {
    const image = await fetch(employee.faceImage).then((res) =>
res.blob());
    const requestOptions = {
        method: "PUT",
        body: image,
    };

    const response = await fetch(S3presignedUrl, requestOptions);
    return response;
}
```

A continuación, si la imagen es subida a S3 correctamente, se vuelve a llamar a la API para crear el empleado con su información asociada.

Este procedimiento en la API tiene varias funciones. Inicialmente, debe agregar el usuario en el pool de usuarios de “ClockInConnect”, esto es, el grupo de usuarios de la aplicación móvil, ya que se trata de la aplicación que van a usar los empleados y donde únicamente deben estar registrados como usuarios.

```
public async Task<bool> AddEmployeeToClockInConnectUserPoolAsync(Employee
employee)
{
    try
    {
        string accessKey =
Environment.GetEnvironmentVariable("AccessKey");
        string secretKey =
Environment.GetEnvironmentVariable("SecretKey");
        AmazonCognitoIdentityProviderClient client = new
AmazonCognitoIdentityProviderClient(accessKey, secretKey,
RegionEndpoint.EUWest1);
        AdminCreateUserRequest adminCreateUserRequest = new
AdminCreateUserRequest()
```

```
        {
            Username = employee.Dni,
            DesiredDeliveryMediums = new List<string>() { "EMAIL" },
            UserAttributes = new List<AttributeType>()
            {
                new AttributeType()
                {
                    Name = "email",
                    Value = employee.Email
                }
            },
            UserPoolId =
Environment.GetEnvironmentVariable("ClockInConnectUserPoolId"),
            TemporaryPassword =  Utils.CreateRandomPassword(8)
        };
        AdminCreateUserResponse adminCreateUserResponse = await
client.AdminCreateUserAsync(adminCreateUserRequest);
        return adminCreateUserResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
    }
    catch (Exception ex)
    {

LambdaLogger.Log($"AuthService::AddEmployeeToClockInConnectUserPoolAsync:
:Exception:{ex}");
        return false;
    }
}
```

En el código mostrado se muestra este proceso de creación de un usuario asociado al empleado recibido. En este proceso se realiza una llamada al servicio en la nube de Cognito mediante el SDK de AWS Cognito con el método “AdminCreateUserRequest”. Este método permite a los administradores del grupo de usuarios crear usuarios programáticamente, de modo que los empleados no tienen que registrarse personalmente desde la aplicación móvil, sino que se les creará un usuario automáticamente cuando el administrador de su empresa cree el empleado.

Adicionalmente, Cognito se encarga de enviar un correo al usuario con la contraseña aleatoria temporal que hemos establecido, de modo que el usuario será forzado a cambiar la contraseña en su primer inicio de sesión en la aplicación móvil.

Si el usuario ha sido creado satisfactoriamente en cognito, se procede a llamar al servicio de empleados para la inserción del empleado en base de datos, de igual manera que hacemos cuando creamos una empresa o cualquier otra entidad del sistema.

Finalmente, la API devuelve como respuesta una URL de S3 prefirmada para que sea la aplicación cliente la que haga la subida de la imagen a S3.

5.3.7 Desarrollo del software de gestión del fichaje laboral mediante reconocimiento facial

Este último apartado dentro del bloque de desarrollo refleja el principal caso de uso de nuestra aplicación, por lo que se le dedica un apartado completo en este informe.

Como comentamos en el apartado 4.5.2 “Creación de empleado”, durante el proceso de creación de un empleado es obligatoria la adición de una imagen con el rostro del empleado.

Esta imagen será comparada con la imagen tomada por el usuario con su aplicación móvil al fichar (entrada o salida) en la jornada, de modo que sólo se registrará un fichaje satisfactoriamente si las imágenes corresponden a la misma persona.

El primer paso en el desarrollo de esta funcionalidad es el de enviar desde la aplicación móvil una foto tomada por el empleado de su rostro, así como la información básica del mismo para poder identificarlo en base de datos e ir a buscar su imagen registrada en el sistema. Para ello, desde la aplicación cliente se genera la *key* de la imagen del fichaje de jornada a partir del identificador de la empresa,el nombre del empleado y la fecha actual.

A continuación, se realiza una petición a la API enviando como parámetro esta *key* de la imagen para obtener la URL de un bucket S3 auxiliar (clockinfaceimages-temp) que servirá de puntero para enviar la foto. Una vez se recibe la URL de S3, se sube la imagen haciendo una petición a dicha URL con el contenido de la imagen en el cuerpo de la petición.

Finalmente, sólo queda llamar a la API indicando el usuario y la información relativa al fichaje que esta realice el proceso de reconocimiento facial y registre el fichaje en caso de un reconocimiento exitoso. El formato de la petición a la API contiene los siguientes campos en el cuerpo de la petición:
UserId: identificador del usuario en el sistema.
ImageKey: nombre de la imagen del registro de jornada en S3.
ClockTime: fecha y hora del registro de jornada.
Observations: Texto con observaciones del usuario en el fichaje (opcional).
Latitude/Longitude: Localización del dispositivo en el que se realiza el fichaje.

Con la parte *frontend* ya definida e implementada, sólo queda el proceso que se llevará a cabo en la API y que consta de dos tareas principales:

1. Hacer uso de Amazon Rekognition para el registro del fichaje comparando la imagen recibida en la petición HTTP con la imagen con la que el empleado ha sido registrado en el sistema.
2. Almacenar en base de datos el registro de jornada en caso de que el reconocimiento facial sea exitoso.

A continuación, se muestra el código de la API encargado de llevar a cabo el proceso de reconocimiento facial.

```
public async Task<bool> FaceRekognizer(string sourceImage, string
targetImage, string mainBucket, string tempBucket)
{
    try
    {
        string accessKey =
Environment.GetEnvironmentVariable("AccessKey");
        string secretKey =
Environment.GetEnvironmentVariable("SecretKey");
        AmazonRekognitionClient amazonRekognitionClient = new
AmazonRekognitionClient(accessKey, secretKey, RegionEndpoint.EUWest1);
        CompareFacesRequest compareFacesRequest = new
CompareFacesRequest()
        {
            SimilarityThreshold = 98f,
            SourceImage = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = tempBucket,
                    Name = sourceImage
```

```
                }
            },
            TargetImage = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = mainBucket,
                    Name = targetImage
                }
            }
        };
        CompareFacesResponse compareFacesResponse = await
amazonRekognitionClient.CompareFacesAsync(compareFacesRequest);
        bool facesMatch = compareFacesResponse.FaceMatches.Count > 0;
        if(facesMatch) LambdaLogger.Log($"Face confidence:
{compareFacesResponse.FaceMatches[0].Similarity}");
        return facesMatch;
    }
    catch (Exception ex)
    {

LambdaLogger.Log($"EmployeeService::FaceRekognizer::Exception={ex}");
        return false;
    }
}
```

Como se muestra en este código, el proceso de reconocimiento facial FaceRekognizer hace una llamada al método CompareFaces del SDK de Amazon Rekognition, que compara dos imágenes con rostros. Entre los parámetros de este método, indicamos la imagen fuente y destino que vamos a comparar, en este caso, la imagen del fichaje del empleado con la imagen de este registrada en el sistema. Además, incluimos el umbral de similitud de las imágenes que estamos comparando (SimilarityTreshold), en este caso establecemos un umbral de 98/100, para asegurarnos de que ambas caras son realmente de la misma persona y que el registro de jornada no ha sido falseado.

El método CompareFaces nos devuelve entre los atributos del objeto el campo FaceMatches. Este campo nos indica el número de caras del parámetro *source* que coinciden con la imagen marcada como *target*. Esto se debe a que tenemos la posibilidad de enviar una lista de imágenes a comparar en lugar de una única imagen como es nuestro caso.

En base a esto último, usamos la propiedad *Count* de las listas en C# para retornar como reconocimiento facial satisfactorio aquel cuyo número de caras coincidentes sea superior a 0.

Si el reconocimiento facial del registro de jornada resulta exitoso, procedemos a almacenar la información del fichaje en base de datos y a guardar la imagen del fichaje en nuestro bucket S3 de producción (clockinfaceimages) en el que almacenamos las imágenes de los fichajes de los empleados.

Debido a que la imagen ya había sido subida a S3 por parte del usuario al inicio de este proceso, la subida al bucket de producción consiste en hacer una copia del objeto del bucket auxiliar en el bucket de producción.

```
public async Task<bool> SaveEmployeeClockImageOnS3(string imageKey,
string clockInBucket, string clockInTempBucket, DateTime clockTime)
{
    string accessKey = Environment.GetEnvironmentVariable("AccessKey");
    string secretKey = Environment.GetEnvironmentVariable("SecretKey");
```

```
    AmazonS3Client s3Client = new AmazonS3Client(accessKey, secretKey,
RegionEndpoint.EUWest1);
    try
    {
        CopyObjectRequest copyObjectRequest = new CopyObjectRequest()
        {
            SourceBucket = clockInTempBucket,
            SourceKey = imageKey,
            DestinationBucket = clockInBucket,
            DestinationKey = imageKey,
            MetadataDirective = S3MetadataDirective.REPLACE
        };
        CopyObjectResponse copyObjectResponse =  await
s3Client.CopyObjectAsync(copyObjectRequest);
        if (copyObjectResponse.HttpStatusCode != HttpStatusCode.OK)
return false;
        bool imageDeletedFromTempBucketSuccessfully = await
Utils.DeleteS3Image(clockInTempBucket, imageKey);
        if(!imageDeletedFromTempBucketSuccessfully)
LambdaLogger.Log($"EmployeeService::SaveEmployeeClockImageOnS3 -
Image={imageKey} could not be deleted from temp bucket");
        return true;
    }
    catch (Exception ex)
    {

LambdaLogger.Log($"EmployeeService::SaveEmployeeClockImageOnS3:Exception=
{ex}");
        return false;
    }
}
```

Tras este proceso llevado a cabo en la API, el servicio web retorna al usuario si el fichaje ha resultado o no exitoso. Cuando el usuario recibe la respuesta, se retorna a la pantalla principal de aplicación móvil (Home) con el mensaje recibido de la API.

6. Conclusiones

Al inicio del proyecto establecimos el objetivo principal de elaborar un sistema informático capaz de gestionar de una manera sencilla el control horario de la jornada laboral de los trabajadores con reconocimiento facial mediante el uso de Amazon Web Services.

Podemos afirmar que este objetivo ha sido cumplido, pues el sistema desarrollado presenta como características fundamentales su usabilidad y su arquitectura "serverless" basada en el uso de servicios en la nube. Los principales casos de uso, como la creación de empresas y empleados, así como el registro de fichajes de los empleados mediante reconocimiento facial han sido implementados satisfactoriamente. Además, han surgido casos de uso durante el desarrollo del proyecto como la visualización y descarga de informes y estos también se han implementado satisfactoriamente.

Si bien todas estas características (usabilidad del sistema, reconocimiento facial en el fichaje de los trabajadores, etc.) establecidas en el documento TFT01 han sido implementadas satisfactoriamente, durante el desarrollo del proyecto han aparecido nuevos casos de uso, de los cuáles algunos no fueron finalmente implementados, y que resultan interesantes para la elaboración de un producto final más completo.

A continuación, se enumeran algunas de estas características (ordenadas de mayor a menor importancia a criterio personal) a implementar para un hipotético despliegue del sistema en producción:

1. Posibilidad de cargar la información (empleados, roles, etc.) por parte de las empresas de forma masiva (vía excel).
2. Personalización de los informes.
3. Programación de informes para su envío vía correo electrónico.
4. Calendario interactivo en el que los administradores puedan establecer las festividades, así como adjuntar notas.
5. Funcionalidad de la aplicación en modo *offline* mediante el uso de una base de datos local a la aplicación.
6. Notificaciones tanto a administradores como a empleados (Por ejemplo: notificar a un empleado que no ha cerrado su jornada en la hora prevista).
7. Gestión de pausas en la jornada laboral.

A nivel personal, valoro especialmente la capacidad que he tenido de desarrollar toda la arquitectura del sistema usando AWS, pues si bien tenía conocimientos previos de uso de algunos servicios de AWS, hasta ahora no había tenido la oportunidad de implementar desde cero un sistema basado en la nube. Las mayores dificultades han surgido en el desarrollo del *frontend* de las aplicaciones, campo en el que no tenía demasiada experiencia y que no es especialmente de mi agrado. Sin embargo, he podido lidiar con la frustración tan común en el desarrollo de software que aparece cuando algunos problemas parecen imposibles de resolver, encontrando alternativas a la resolución de estos problemas. En definitiva, he ganado bastante experiencia con este proyecto en el desarrollo de aplicaciones usando servicios de AWS. Con ello, veo posible enfrentarme a proyectos más difíciles con arquitecturas más complejas en desarrollos futuros, y seguir avanzando en mis conocimientos de *backend* y de la nube.

Anexo I Competencias

IS01: "Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software."

Justificación: Durante el transcurso de realización del proyecto, se ha desarrollado un software compuesto por dos aplicaciones: una aplicación web para administradores de la empresa y una app móvil para los empleados. Previo a este desarrollo software, se ha creado una infraestructura en la nube de AWS con los servicios necesarios para implementar este sistema (autenticación, base de datos, *hosting* web, etc.) Adicionalmente, se ha llevado un mantenimiento continuo del software de dichas aplicaciones, así como de la infraestructura en la nube para adecuarse a los nuevos casos de uso que han ido apareciendo durante el desarrollo del proyecto.

IS03: "Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones."

Justificación: Al comienzo del desarrollo del TFT, se consultó a compañeros de la empresa en la que trabajo sobre las herramientas de las que disponíamos para el control de la jornada laboral. En el momento de iniciar este proyecto, el control horario se estaba realizando en papel, indicando las horas de entrada y salida de cada jornada, así como las horas totales del día. Con el constante cambio en el tiempo de trabajo diario de una semana a otra, debido a las horas extraordinarias fruto de los plazos de finalización de los proyectos que estábamos realizando, el control de estas horas extraordinarias se volvía una quimera. Especialmente en aquellos empleados a tiempo parcial que tenían una jornada flexible. Se hacía indispensable el poder contar con un sistema digital donde poder registrar la entrada y la salida, así como visualizar los registros de fichajes anteriores. Es por esto por lo que el poder contar con una aplicación que nos permitiese lograr este objetivo fue recibido unánimemente como el camino a seguir.

IS04: "Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales."

Justificación: Al comienzo del desarrollo del TFT, se identificó el problema existente de muchas empresas que carecían de un sistema informático para el control horario de la jornada laboral de sus empleados, con los consiguientes problemas que esto acarrea cuando se producían inspecciones laborales y no se tenía un control preciso de esta información. Es por ello por lo que se ha desarrollado una solución a través de un sistema de dos aplicaciones tanto para administrar a los empleados como para el fichaje de estos. Este desarrollo ha constado de varias partes:

- Diseño y desarrollo de la interfaz de usuario de las aplicaciones web (administradores) y móvil (empleados)
- Implementación de la lógica de negocio (*backend*) a través de servicios web consumidos por las aplicaciones para su enlace con la base de datos.
- Documentación de los servicios web (APIs) a través de la herramienta Swagger, que será descrita posteriormente en este informe.

- Verificación del software implementado en los servicios web mediante el uso de herramientas de prueba de APIs como Postman y Swagger, nombrado previamente. Ambas herramientas nos permiten testear mediante peticiones HTTP nuestros servicios web, que en última instancia son los que manejan la lógica y los datos de nuestro sistema. Posteriormente en este informe ambas herramientas serán descritas más en profundidad.

IS05: "Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse."

Justificación: Tanto al inicio como durante el desarrollo del proyecto, se ha hecho uso de numerosos recursos en la nube de AWS. Estos recursos tienen un coste tanto por el tipo de recurso como por su tiempo de uso. Se ha realizado una monitorización continua de estos costes a través de la interfaz de AWS Cost Explorer, que nos permite ver el coste actual y proyectado de nuestros servicios, con un desglose muy detallado de estos. Adicionalmente, se ha hecho uso de AWS Billing Alarm, que nos notifica a nuestro email/teléfono móvil cuando se supera una cantidad umbral fijada.

Anexo II Normativa y legislación

A la hora de desarrollar cualquier proyecto software, es fundamental conocer la legislación vigente. Un mal conocimiento de esta normativa o legislación puede acarrear sanciones económicas o incluso delitos penales.

En la aplicación se almacena y trata información sensible de los potenciales usuarios tales como: imagen de los empleados, DNI, dirección, teléfono móvil, email, etc. A continuación, se nombran las principales normativas a las que se adecuaría nuestro proyecto en caso de un hipotético despliegue a producción.

Reglamento general de protección de datos (RGPD)

"El Reglamento General de Protección de Datos (RGPD) es el reglamento europeo relativo a la protección de las personas físicas en lo que respecta al tratamiento de sus datos personales y a la libre circulación de estos datos. Entró en vigor el 24 de mayo de 2016 y fue de aplicación el 25 de mayo de 2018, dos años durante los cuales las empresas, las organizaciones, los organismos y las instituciones se fueron adaptando para su cumplimiento. Es una normativa a nivel de la Unión Europea, por lo que cualquier empresa de la unión, o aquellas empresas que tengan negocios en la Unión Europea, que manejen información personal de cualquier tipo, deberán acogerse a ella. Las multas por el no cumplimiento del RGPD pueden llegar a los 20 millones de euros.

En España, el RGPD dejó obsoleta la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) de 1999, siendo sustituida el 6 de diciembre de 2018 por la Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales, acorde con el RGPD". [22]

Ley orgánica de protección de datos personales y garantía de los derechos digitales (LOPD-GDD)

"La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD) es una ley orgánica aprobada por las Cortes Generales de España que tiene por objeto adaptar el Derecho interno español al Reglamento General de Protección de Datos. Esta ley orgánica deroga a la anterior Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (aunque se mantiene vigente para la regulación de ciertas actividades)". [23]

"Esta ley entró en vigor el 7 de diciembre de 2018". [24]

Real Decreto-ley 8/2019, de 8 de marzo, de medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo

"Real Decreto-ley dispuesto el 8 de marzo de 2019, anunciado en el Boletín oficial del Estado (BOE) el 12 de marzo de 2019. Establece medidas de control social y lucha frente a la precariedad laboral, centrándose en el control horario de la jornada laboral.

Asimismo, el artículo 10 de este decreto modifica el artículo 34 del estatuto de trabajadores, incluyendo un nuevo apartado (9) que obliga a las empresas a realizar "un registro diario de

jornada, que deberá incluir el horario concreto de inicio y finalización de la jornada de trabajo de cada persona trabajadora" sin perjuicio de la flexibilidad horaria establecida en el Estatuto de los Trabajadores, pero sin concretar la forma de realizar dicho registro. La empresa tendrá que conservar los registros durante cuatro años. Incumplir esta obligación será una infracción grave que supone una multa de 206 a 6.250 euros". [25]

Anexo III Manual de usuario

Requisitos previos

Los requisitos para el uso de las aplicaciones del sistema son:

- Navegador web (Chrome, Mozilla, Safari, Edge, etc.) con conexión a internet.
- Dispositivo móvil Android

Registro de empresa (aplicación web)

El primer paso a realizar es la creación de una empresa donde agregar nuestros empleados. En la página principal de la aplicación web aparecerá en el margen derecho un formulario de inicio de sesión, con un botón en su parte inferior con el texto “Register your company”.

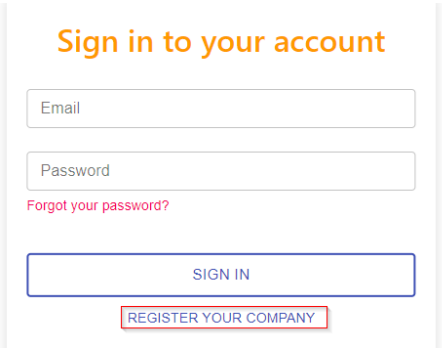

Figura 9.1 Formulario de inicio de sesión

Al pulsar en el enlace, se mostrará un formulario de creación de empresa, el usuario debe rellenarlo y hacer click en “Register”.

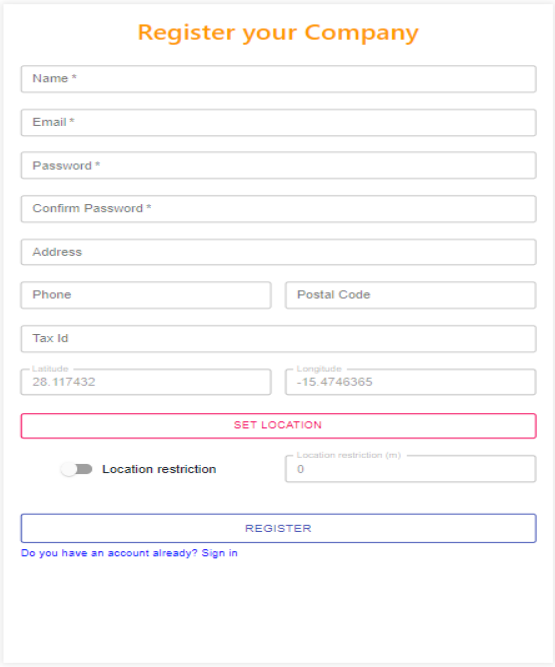

Figura 9.2 Formulario de registro de empresa

Inicio de sesión (aplicación web)

Tras crear la empresa, la web muestra al usuario una vista en la que introducir el código de verificación que el usuario ha recibido al email puesto en el formulario de creación de empresa.

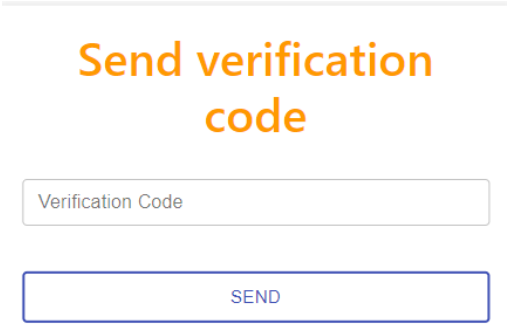

Figura 9.3 Enviar código de verificación de usuario

Con esto, el usuario puede iniciar sesión rellenando el formulario mostrado en la figura 9.1.

Creación de lugar de trabajo (aplicación web)

Una vez iniciado sesión como usuario en administrador, podemos comenzar por crear un lugar de trabajo que asignar a nuestros empleados. Para ello, navegamos a través del menú superior *Company – Workplaces.*

En la vista de lugares de trabajo, debemos hacer click en el botón con el icono de un “+” en la esquina superior derecha.

Figura 9.4 Agregar un lugar de trabajo (1)

Tras pulsar sobre el “+”, se nos aparecerá una nueva vista con un formulario de registro de lugar de trabajo, el usuario debe rellenarlo y hacer click en “Create Workplace”.

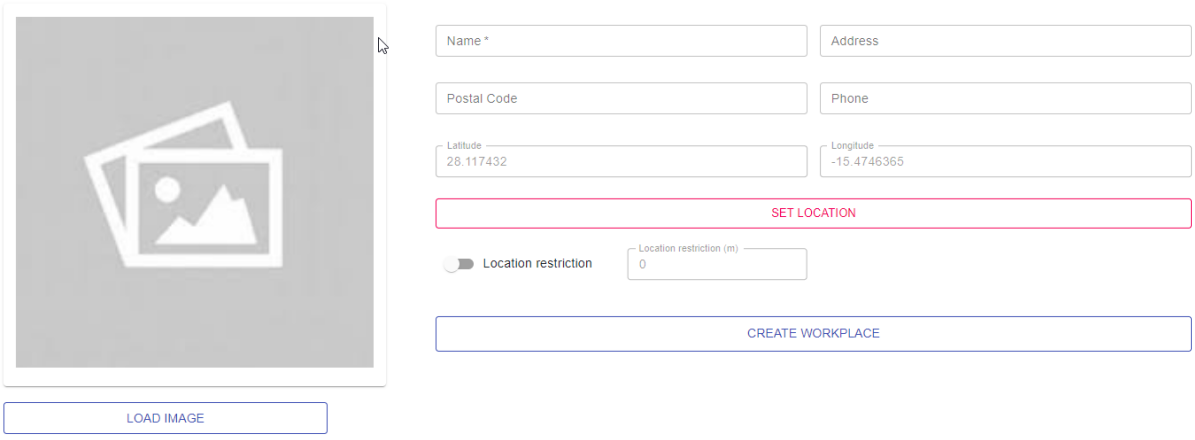

Figura 9.5 Agregar un lugar de trabajo (2)

Creación de rol (aplicación web)

Otra de las funcionalidades principales es la de crear roles para nuestros empleados. Para ello, navegamos a través del menú superior *Company – Role.*

En la vista de roles, debemos hacer click en el botón con el icono de un "+" en la esquina superior derecha.

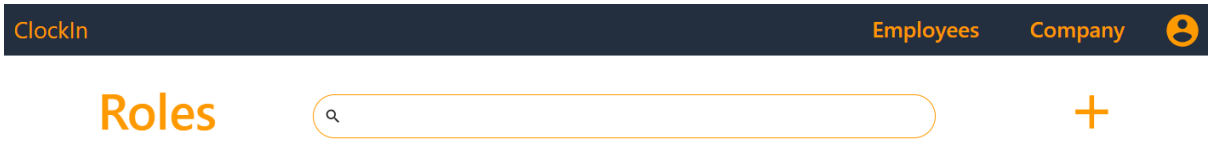

Figura 9.6 Crear un rol (1)

Tras pulsar sobre el "+", se nos mostrará un *popup* con un formulario de creación de rol, en el que indicamos su nombre y las horas de entrada y salida de jornada. El usuario debe rellenarlo y hacer click en "Save".

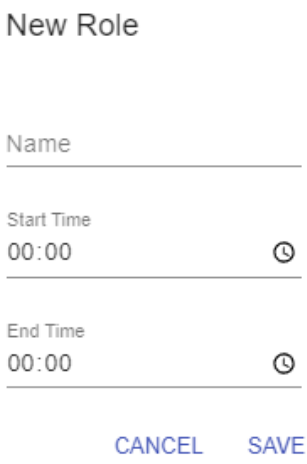

Figura 9.7 Crear un rol (2)

Creación de empleado (aplicación web)

A continuación, se detalla cómo crear un empleado, para que estos puedan usar la aplicación y registrar sus fichajes.

En cualquiera de las vistas del menú de empleados (*Roster Summary* y *Manage Employees*), debemos hacer click en el botón con el icono de un "+" en la esquina superior derecha.

Figura 9.8 Crear un empleado (1)

Tras pulsar sobre el "+", se nos mostrará un formulario de creación de un empleado. En él, debemos cargar una imagen con el rostro del empleado y rellenar su información personal,

pudiéndole agregar un rol y un lugar de trabajo. El usuario debe rellenar este formulario y hacer click en “Create employee”.

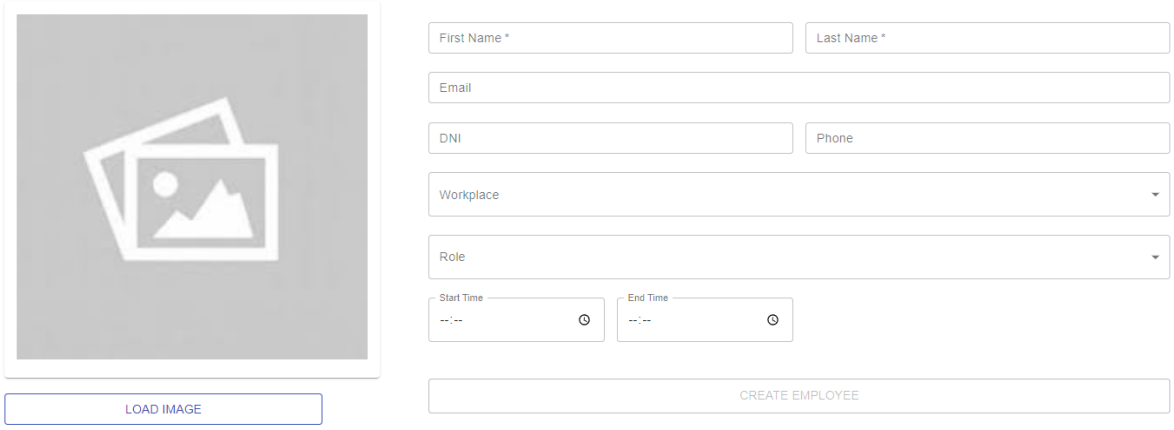

Figura 9.9 Crear un empleado (2)

Tras esto, el empleado recibirá su usuario (dni) y su contraseña temporal al email indicado por el administrador en su formulario de registro.

Inicio de sesión (aplicación móvil)

Tras los pasos realizados previamente por el administrador en la aplicación web, ya el empleado puede usar la aplicación móvil y registrar sus fichajes diarios. Para ello, debe instalar y ejecutar la aplicación móvil en su dispositivo.

A continuación, debe iniciar sesión con su usuario y su contraseña temporal que habrá recibido a su email. Al introducir estos datos, se le mostrará un nuevo formulario solicitándole un cambio de contraseña. Una vez realizado, ya puede iniciar sesión de un modo normal.

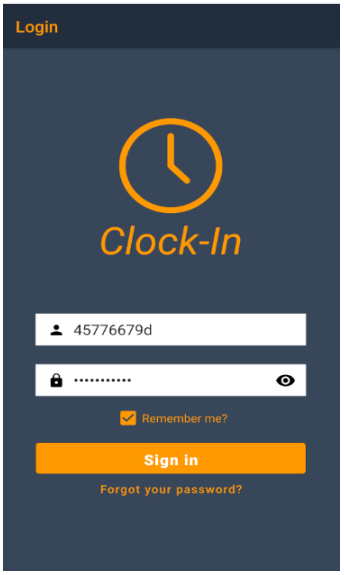

Figura 9.10 Inicio de sesión de empleado

Registrar entrada de jornada (aplicación móvil)

En esta sección se detalla el proceso de fichaje del empleado en la aplicación móvil. Tras el inicio de sesión, se le muestra al usuario la vista principal de la aplicación, con dos botones para los registros de jornada (entrada y salida).

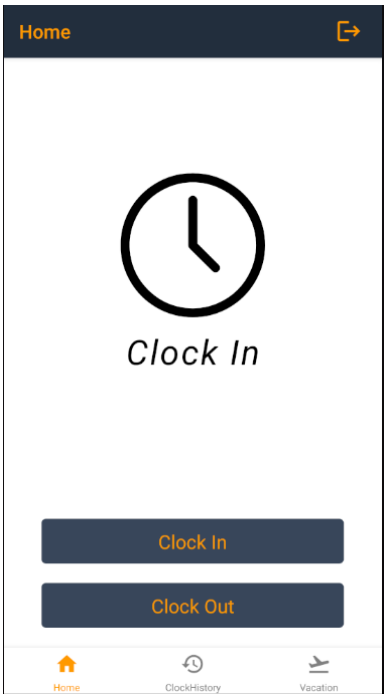

Figura 9.11 Registro de entrada del empleado (1)

El usuario debe hacer click en el botón con el texto “Clock In” para fichar su entrada. Tras esto, se le mostrará una ventana con un cuadro de texto (opcional) para indicar alguna observación en el fichaje. El usuario debe hacer click en “Continue” para continuar con el proceso de fichaje.

A continuación, la aplicación requerirá al usuario permisos de ubicación y de cámara para poder realizar el fichaje, el usuario debe aceptarlos para proseguir.

Una vez aceptados estos permisos, se abrirá la cámara del dispositivo en la aplicación para que el usuario tome una foto de su rostro, haciendo click en el botón con el icono de una cámara.

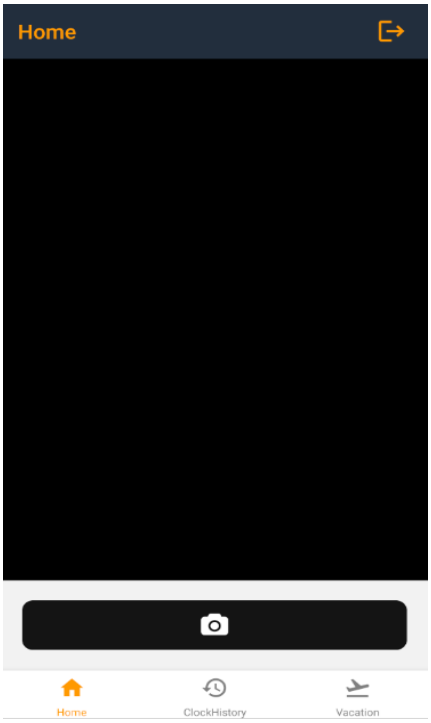

Figura 9.12 Registro de entrada del empleado (2)

Ya capturada la imagen, se le muestra al usuario la foto capturada, pudiendo cancelar o finalizar el proceso de fichaje haciendo click en el botón con el texto “Send photo”. Finalmente, tras un proceso que dura unos segundos en los que se le muestra al usuario una pantalla de carga, la aplicación retorna al usuario a la vista principal. Para comprobar que el fichaje ha sido exitoso, el usuario puede navegar a la vista “Clock History” haciendo click en el menú inferior de la aplicación en dicha vista.

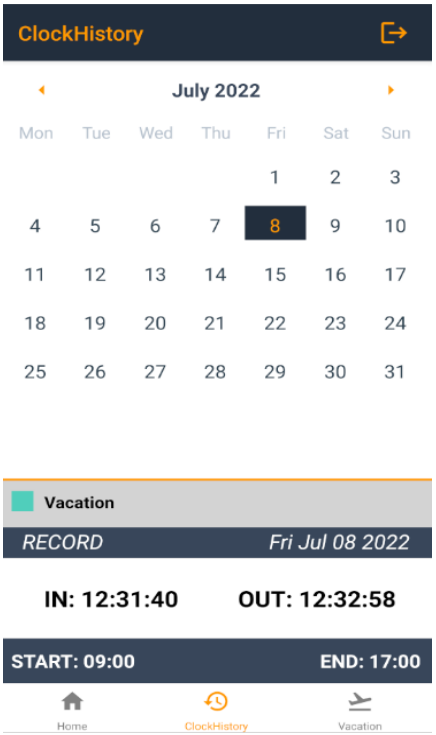

Figura 9.13 Registro de entrada del empleado (3)

En esta vista se muestra un calendario con el histórico de fichajes del empleado.

Descargar informe de registros de jornada (aplicación web)

Para obtener la información del histórico de fichajes de nuestros empleados, podemos generar y descargar un informe en la web.

Para ello, el usuario debe iniciar sesión en la web y acceder a la vista de informes a través del menú *Company – Reports*. Ya en la vista de informes, se observa un desplegable con los informes disponibles en el sistema.

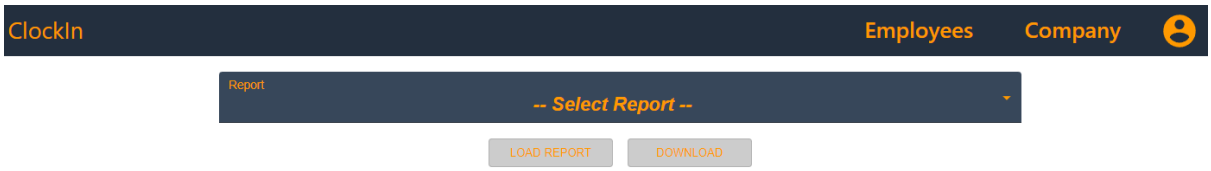

Figura 9.14 Descargar informe de registros de jornada (1)

El usuario deberá expandir el desplegable y seleccionar el informe “Employee Records”. Tras ello, se le mostrarán los filtros asociados a este informe en la parte inferior del desplegable de informes.

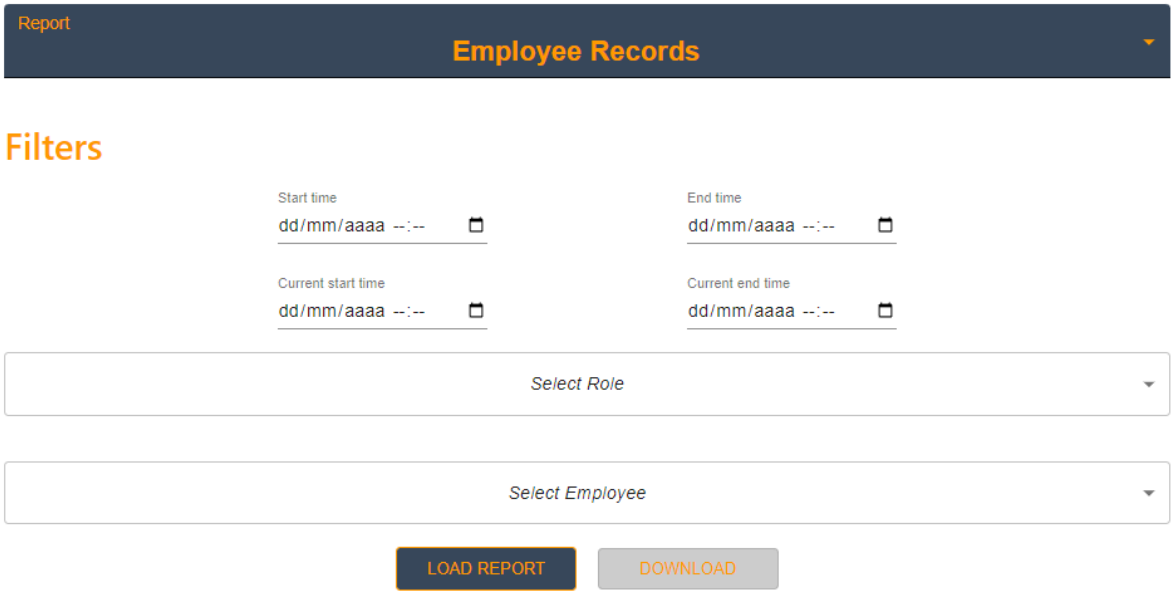

Figura 9.15 Descargar informe de registros de jornada (2)

Tras establecer los filtros deseados (todos opcionales), el usuario debe hacer click en el botón con el texto “Load Report” para generar el informe.

Employees Clock Records

Name	Role	Date	Time	Type	Location
Carlos Esteban León Díaz	Estudiante	12/07/2022	22:26	IN	
Carlos Esteban León Díaz	Estudiante	12/07/2022	22:58	OUT	
Carlos Esteban León Díaz	Estudiante	12/07/2022	23:31	OUT	
Carlos Esteban León Díaz	Estudiante	13/07/2022	09:33	IN	
Carlos Esteban León Díaz	Estudiante	14/07/2022	13:42	IN	
Carlos Esteban León Díaz	Estudiante	15/07/2022	10:02	IN	
Carlos Esteban León Díaz	Estudiante	17/07/2022	17:27	IN	
Observations:	Esto es una prueba para el envío de observaciones en el fichaje de jornada laboral				
Carlos Esteban León Díaz	Estudiante	17/07/2022	17:33	OUT	
Carlos Esteban León Díaz	Estudiante	20/07/2022	09:54	IN	

Figura 9.16 Descargar informe de registros de jornada (3)

Una vez generado el informe, el usuario debe hacer click en el botón con el texto “Download”, adyacente al de generación del informe, para descargar en su equipo el informe en formato PDF.

Bibliografía

[1] A. Olcese, 24 Mayo 2022. [En línea]. Available: https://www.elmundo.es/economia/macroeconomia/2022/05/24/6287b5a9fc6c83ef7e8b45e5.html.

[2] Boe, 8 Marzo 2019. [En línea]. Available: https://www.boe.es/eli/es/rdl/2019/03/08/8.

[3] Amazon, «docs.aws.amazon,» [En línea]. Available: https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html. [Último acceso: 07 2022].

[4] Amazon, «docs.aws.amazon,» [En línea]. Available: https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/Welcome.html. [Último acceso: Julio 2022].

[5] «hava,» [En línea]. Available: https://www.hava.io/blog/what-is-aws-elastic-beanstalk. [Último acceso: Julio 2022].

[6] Amazon, «docs.aws.com,» [En línea]. Available: https://docs.aws.amazon.com/es_es/cognito/latest/developerguide/what-is-amazon-cognito.html. [Último acceso: Julio 2022].

[7] Amazon, «docs.aws.amazon,» [En línea]. Available: https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html. [Último acceso: Julio 2022].

[8] Microsoft, «docs.microsoft,» 8 Julio 2022. [En línea]. Available: https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/.

[9] «arimetrics,» [En línea]. Available: https://www.arimetrics.com/glosario-digital/javascript. [Último acceso: Julio 2022].

[10] Mozilla, «developer.mozilla,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics. [Último acceso: Julio 2022].

[11] A. Robledano, «openwebinars,» 26 Junio 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-css/.

[12] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Transact-SQL. [Último acceso: Julio 2022].

[13] «reactjs,» [En línea]. Available: https://es.reactjs.org/tutorial/tutorial.html. [Último acceso: Julio 2022].

[14] Microsoft, «docs.microsoft,» 20 Abril 2022. [En línea]. Available: https://docs.microsoft.com/es-es/windows/dev-environment/javascript/react-native-for-windows.

[15] «git-scm,» [En línea]. Available: https://git-scm.com/. [Último acceso: Julio 2022].

[16] M. H. P. V. Stephen J. Collings, «runebook,» 2014. [En línea]. Available: https://runebook.dev/es/docs/react_bootstrap/getting-started/why-react-bootstrap/index#:~:text=Why%20React%2DBootstrap%3F,React%2DBootstrap%20es%20una%20reimplementaci%C3%B3n%20completa%20de%20los%20componentes%20de,tiene%20todo%20lo%20que%20necesita.

[17] Microsoft, «visualstudio.microsoft,» [En línea]. Available: https://visualstudio.microsoft.com/es/. [Último acceso: Julio 2022].

[18] Postman, «postman,» [En línea]. Available: https://www.postman.com/product/what-is-postman/. [Último acceso: Julio 2022].

[19] Swagger, «swagger,» [En línea]. Available: https://swagger.io/about/. [Último acceso: Julio 2022].

[20] Microsoft, «visualstudio.microsoft,» 2022. [En línea]. Available: https://visualstudio.microsoft.com/es/license-terms/vs2022-ga-community/.

[21] Amazon, [En línea]. Available: https://calculator.aws/#/. [Último acceso: Julio 2022].

[22] M. Roberts, «martinfowler,» 22 Mayo 2018. [En línea]. Available: https://martinfowler.com/articles/serverless.html.

[23] Amazon, «aws.amazon,» [En línea]. Available: https://aws.amazon.com/es/visualstudio/. [Último acceso: Julio 2022].

[24] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Reglamento_General_de_Protecci%C3%B3n_de_Datos. [Último acceso: Julio 2022].

[25] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_Personales_y_garant%C3%ADa_de_los_derechos_digitales. [Último acceso: 07 2022].

[26] Boe, 6 Diciembre 2018. [En línea]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673.

[27] «noticias jurídicas,» 12 Marzo 2019. [En línea]. Available: https://noticias.juridicas.com/actualidad/noticias/13772-las-15-novedades-del-rdl-8-2019-de-medidas-urgentes-de-proteccion-social-y-de-lucha-contra-la-precariedad-laboral-en-la-jornada-de-trabajo/.