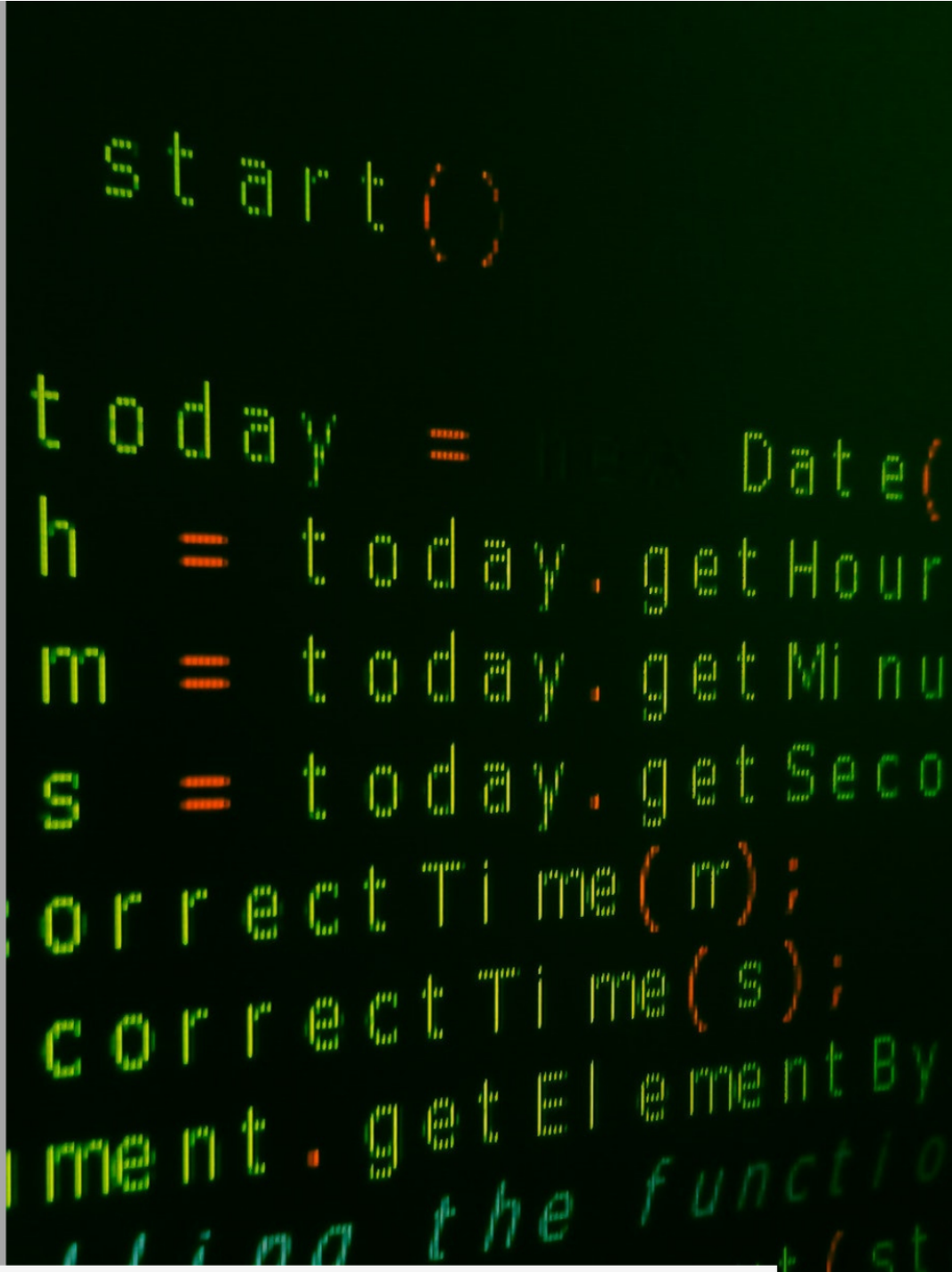
Florida Universitària

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport



Memoria de Proyecto Final de Grado

Desarrollo de Aplicaciones Multiplataforma

Eduardo Calvo Feltrer - Curso 2022-2023

Check-In App

Aplicación de fichada en el entorno laboral

ÍNDICE DE CONTENIDOS:

ÍNDICE DE FIGURAS:

1. RESUMEN DEL PROYECTO:

El proyecto ha consistido en la creación de una aplicación multiplataforma para dispositivos móviles mediante el framework *React Native,* así como de su integración a un ERP de *Microsoft Dynamics Axapta 2012.*

Fig. 1. Logo *React Native*.

Fig. 2. Logo *Microsoft Dynamics*.

La aplicación desarrollada consta de un sistema de fichaje de entrada y salida de la jornada laboral como función principal, y otras dos funciones que permiten visualizar el historial de registros y el horario del usuario. El objetivo de la aplicación es permitir a los empleados de una empresa registrar de manera sencilla y eficiente su jornada laboral y poder acceder a su historial y horario de trabajo de forma clara y organizada. La aplicación ha sido desarrollada con el framework *React Native* a través de la plataforma online *Snack Expo*, que permite emular un dispositivo móvil con la aplicación en desarrollo instalada para poder ver su funcionamiento a medida que se van añadiendo funcionalidades sin la necesidad de instalar ningún programa, ya que todo se realiza desde el navegador web.

Para su funcionamiento más allá del interfaz visual ha sido necesaria la creación de un esquema de datos organizado en tablas que agrupan toda la información necesaria, una base de datos donde definir dicho esquema, y la creación de métodos para gestionar solicitudes e inserciones de información en dichas tablas.

Dichos métodos han sido realizados, en primer lugar, en un entorno de pruebas al margen del ERP para poder probar las funcionalidades de la aplicación a medida que se ha ido desarrollando, y, posteriormente han sido replicados en el lenguaje que emplea *Microsoft Dynamics Axapta 2012* (X++) junto a la lógica necesaria y los permisos necesarios para su funcionamiento. Estos métodos han sido diseñados

para operar de manera eficiente y segura, garantizando que la información se almacene correctamente y esté disponible cuando se necesite.

La base de datos ha sido generada en primera instancia para el entorno de pruebas en *MariaDB* a través de *HeidiSQL*, una herramienta de escritorio que permite consultar y confeccionar tablas y relaciones entre tablas dentro de una base de datos definida en un sistema local o remoto, así como probar sentencias SQL para asegurar que devuelven el *output* deseado. El servidor con dicha base de datos de prueba se ha ubicado en una máquina virtual de *Amazon Web Services* (AWS) a la que tenemos acceso como alumnos de Florida.

En la máquina virtual, además de la base de datos, ha sido necesario instalar *Xampp* para poder ejecutar MySQL gracias a sus herramientas de gestión de base de datos, y *Eclipse,* un entorno de desarrollo integrado (IDE) multi-lenguaje del que han sido utilizadas sus herramientas para programación de aplicaciones Java, además del paquete *Java Runtime Environment* (JRE) instalado en las dependencias del IDE para posibilitar la ejecución del código.

Con estas herramientas ha sido desarrollada una aplicación Java que en ejecución crea un servidor multihilo en la máquina virtual conectado a la base de datos, capaz de gestionar peticiones que entren a la dirección IP de la máquina virtual y en función de unas determinadas condiciones que las peticiones deben cumplir, extraer la información que dichas peticiones solicitan de la base de datos, con el formato adecuado para su lectura desde la aplicación.

Para someter a prueba estos intercambios de información también ha sido utilizada la herramienta de escritorio *Postman*, desde la que se han introducido las URLs que, tras confirmar su correcto funcionamiento, se implementarían en los *fetch* que realiza la aplicación para interactuar con la base de datos.

Tecnologías utilizadas:

Fig. 3. Logo Eclipse IDE. Fig. 4. Logo Xampp. Fig. 5. Logo Postman

Fig. 6. Logo HeidiSQL. Fig. 7. Logo Maria DB. Fig. 8. Logo Expo.

2. JUSTIFICACIÓN Y OBJETIVOS DEL PROYECTO:

El objetivo del proyecto ha sido combinar competencias adquiridas durante el transcurso del grado superior en *Desarrollo de Aplicaciones Multiplataforma*, las cuales han sido el desarrollo de aplicaciones tanto en *React Native* como en *Java,* la creación de una máquina virtual alojada en un servidor remoto y la utilización de un servidor multihilo para procesar solicitudes HTTP; con competencias adquiridas durante el transcurso de las prácticas, como el conocimiento y manipulación del entorno de *Microsoft Dynamics Axapta 2012* y su lenguaje de código propio (X++) con el fin de aprender a sacarle partido a la amplia variedad de funciones que esta última tiene, a través de la creación de una aplicación con posible cabida real dentro del marco de una empresa, aprendiendo también a elaborar un *roadmap* de desarrollo que se ajuste a los plazos propuestos subdivididos en pequeños plazos autoimpuestos con metas concretas, realistas y alcanzables acorde con nuestro tiempo y recursos.

La parte relativa a la aplicación y la base de datos de prueba ha sido desarrollada por mí mismo en solitario bajo la supervisión periódica de Roberto Sanz Requena, tutor asignado para la monitorización del PFC mientras que para la implementación con *Dynamics* he contado con la supervisión y consejo del responsable de mis prácticas en la empresa: Enrique Navarro Fabiá.

El producto final de este proyecto es una aplicación móvil que pretende ser una plantilla configurable en lo que respecta a los colores y las imágenes y logotipos empresariales que aparecen en ella, para su posterior exportación a un paquete instalable en un teléfono móvil, y un paquete de instalación (archivo .xpo) para *Microsoft Dynamics Axapta 2012* que contenga las tablas, relaciones, servicios web y clases necesarias para que la conectividad con la aplicación sea posible.

Dicho producto estaría destinado principalmente a empresas cuya actividad se desarrolla en una oficina y en teletrabajo, conjuntamente. La aplicación en sí misma únicamente permite iniciar sesión, entrar y salir de la jornada laboral y consultar el historial de registros y el horario asignado al usuario. Estos datos podrían ser luego consultados desde Dynamics implementando los formularios correspondientes que consuman las tablas que emplea la aplicación móvil para su funcionamiento.

3. DESARROLLO DEL PROYECTO:

Dado que este proyecto está más enfocado al desarrollo de competencias propias de una empresa como es en este caso el uso del sistema de *Microsoft Dynamics Axapta 2012,* y no lo está tanto en desarrollar un producto con proyección en el mercado, la búsqueda de antecedentes dentro del mismo nicho ha sido, principalmente, la de extraer ideas de varios ejemplos existentes, más que la realización de un estudio de mercado extenso y profundo.

Sin embargo, algo que ha llamado mi atención ha sido que no he encontrado un amplio catálogo de aplicaciones de este tipo vinculadas a un ERP. La práctica más habitual es la de aplicaciones de fichada de entrada y salida con una base de datos propia (de la empresa desarrolladora/distribuidora) No he encontrado ninguna aplicación que se venda con el extra de poder estar conectada a la base de datos del ERP de la empresa.

Aunque no existe competencia (oportunidad), no existe por un motivo y es la sensibilidad de los datos que las empresas alojan dentro de sus sistemas, y lo poco seguro que sería conceder a ciegas los permisos necesarios para que una aplicación de terceros pueda extraer y escribir información en su base de datos.

Por ello, sería todo un desafío lanzar al mercado la aplicación como elemento individual para que empresas adquieran el servicio e instalen el paquete en su ERP. En su lugar, este desarrollo puede servir para incorporarlo como elemento extra (tras muchas más horas de desarrollo y perfeccionamiento) dentro del servicio prestado al adquirir *Microsoft Dynamics Axapta 2012* a través de una consultoría informática de implantación de ERPs como es el caso de *Emiral.*

3.1. Metodología de trabajo.

Para llevar a cabo el proyecto, se ha aplicado una metodología de trabajo combinada entre *SCRUM* y un roadmap inicial. Este roadmap partió de una lista de requisitos mínimos proporcionados por el supervisor de las prácticas y fue ampliada por mí a una lista de mayor tamaño, incluyendo los elementos necesarios para cumplir con los requisitos. Las tareas se dividieron por días en base al tiempo disponible para poder compaginar el desarrollo del proyecto con el proceso de formación.

Durante todo el proceso el planteamiento ha sido el siguiente: Con la lista de tareas o roadmap confeccionado, consultar cuál era la siguiente función de la app a desarrollar. Comenzar diseñando desde la aplicación java ubicada en la máquina virtual del servidor remoto un nuevo caso que nos devuelva la información deseada de las tablas de la base de datos, o bien introduzca en ellas el contenido que le especifiquemos.

Paralelamente, el tutor del proyecto final de ciclo, el profesor Don Roberto Sanz Requena, supervisó el progreso de los proyectos en grupos mediante Sprints bisemanales. Durante este proceso, el profesor proporcionó comentarios constructivos sobre el progreso del proyecto, sugiriendo cambios y mejoras enfocadas en consolidar un producto final de alta calidad. Además, en cada reunión, se preguntó a los estudiantes sobre sus planes para el siguiente Sprint, lo que permitió asegurarse de que el proyecto avanzara en la dirección correcta y se cumplieran los objetivos establecidos.

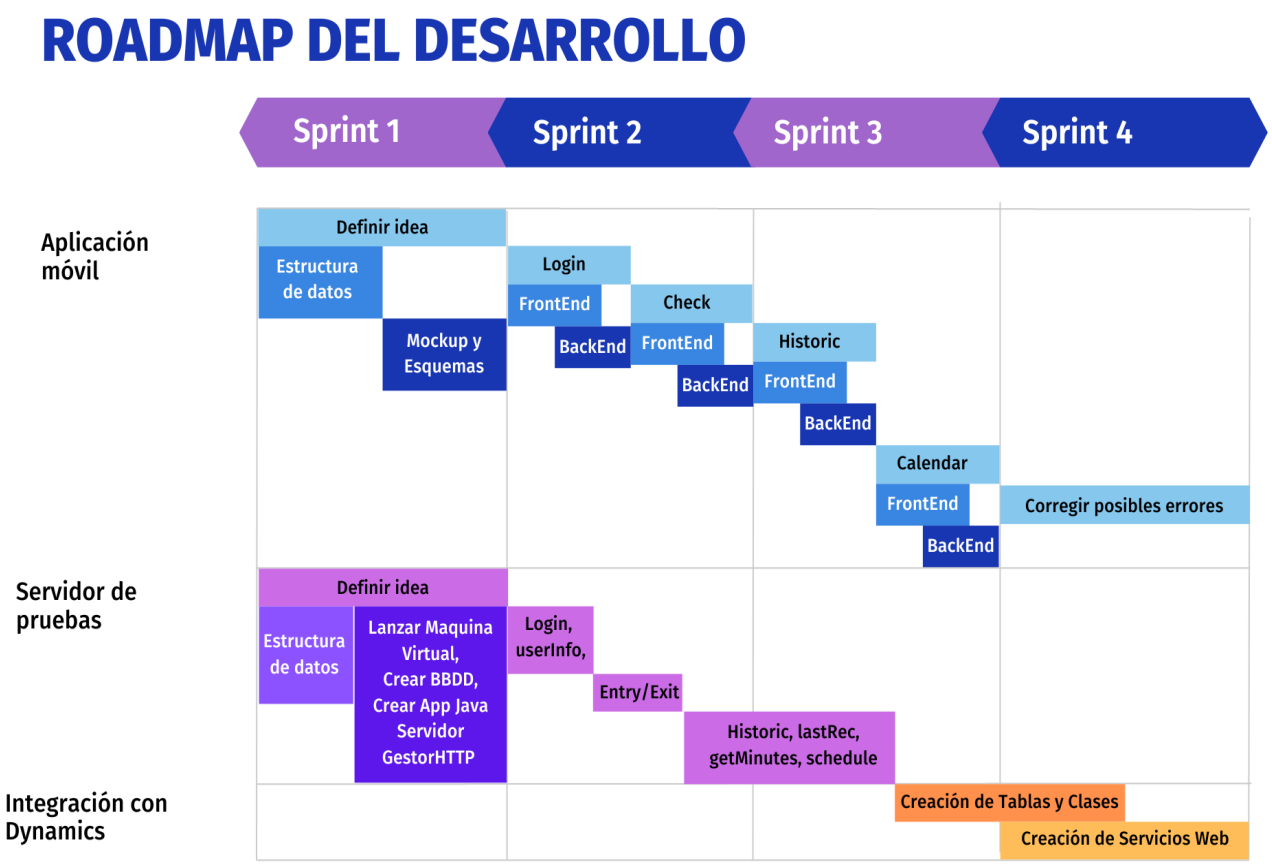


Fig. 9. Esquema del roadmap seguido durante el desarrollo.

3.2. Base de datos.

Puesto que en el momento de comenzar a desarrollar la aplicación mi formación con respecto al entorno de Microsoft Dynamics no era lo suficiente extensa como para ser capaz de desarrollar servicios web capaces de extraer información de tablas del sistema, opté por crear en una máquina virtual en remoto una base de datos de pruebas que además de ayudar a testear la aplicación servivía de guía para crear las tablas necesarias en Dynamics. Por ello, las tablas empleadas en ambos casos son las mismas. A continuación las explicaremos y más adelante discutiremos sus distintas implementaciones:

(rellenas con datos de ejemplo para ilustrarlas mejor)

- checks

CheckID	Username	Type	Date	Time
1	Edcafe	Entry	2023-05-04	08:00:00

En esta tabla se almacenan todos los registros de todos los usuarios. Cuenta con un campo CheckID que ordena los registros con un AutoIncrement (cada registro nuevo adquiere el siguiente valor consecutivo). El campo Type almacena si se trata de una entrada o de una salida, y los campos Date y Time registran cuándo se ha efectuado el registro.

- registeredemployee

EmployeeID	Username	Pwd	GroupID	Office	Status
0001	Edcafe	1234	1	0001	Y

En esta tabla se almacenan los datos de los usuarios registrados en el sistema de check. EmployeeID relacionará estos datos con los de la tabla de empleados del ERP. Username es el nombre que se utiliza en la aplicación para iniciar sesión, y Pwd es la contraseña. GroupID indica a qué grupo de horarios pertenece el usuario (ver siguiente tabla). Office almacena a qué oficina pertenece el usuario y status permite activar o desactivar al usuario desde el sistema (por enfermedad, vacaciones.. etc) sin necesidad de darlo de baja de la base de datos

- hoursgroups

GroupID	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	8:30-18:00	8:30-18:00	8:30-18:00	8:30-18:00	8:30-18:00	free	free

Esta tabla almacena los grupos de horarios y su correspondiente horario, mediante su relación con la anterior tabla se puede asignar un grupo de horario distinto a cada empleado de manera que se puede tanto individualizar como utilizar el mismo horario para varios empleados si fuese necesario.

- employeehoursperday

Username	Date	TotalTime
Edcafe	2023-05-04	480

Esta tabla simplemente almacena la cantidad de horas en minutos realizadas por un empleado en un día concreto. Ya que el planteamiento de los registros de entrada y salida actual permite varias entradas y salidas en un mismo día, esta distribución permite acceder a esta información de manera más clara y ordenada.

3.2.1. Entorno de pruebas.

Como se mencionaba anteriormente, a modo de entorno de pruebas, se creó una base de datos en *MySQL* mediante el asistente *HeidiSQL* para posteriormente gestionar solicitudes entrantes a la IP de la máquina virtual mediante la creación de una aplicación *Java* de escritorio que extraiga e inserte la información pertinente en las tablas de la base de datos Esta aplicación *Java* consta de dos clases: Clase Servidor y Clase Gestor HTTP.

- Servidor. La clase servidor crea un servidor multihilo. Cada vez que se intercepta una petición llama a la clase Gestor HTTP pasándole la URL por parámetro.
- Gestor HTTP. Esta clase se encarga de gestionar las peticiones. En primer lugar las analiza y determina si se trata de una petición tipo GET o POST. Dado que la aplicación no utiliza peticiones tipo PUT ni DELETE no ha sido necesario configurar dichas entradas. Cuando el gestor recibe una petición GET, intenta dividir el string de la URL recibida cortando por el carácter “=”, para separar la solicitud del usuario que la ha realizado. (desde la aplicación móvil, a la hora de confeccionar la URL a la que se va a hacer fetch, a esta se le asigna la función y el usuario, separados por “=”) una vez la aplicación Java dispone de una acción y un usuario, entra en un switch, donde cada case es una de las distintas funciones programadas para el GET. Estas funciones son: login, userInfo, historic, lastRec, getMinutes y schedule.

- login: La app envía en la URL usuario y contraseña separados por un punto y coma “;” el gestor realiza un SELECT a la tabla y compara usuario y contraseña con los proporcionados. Si coinciden devuelve “OK”. La app al recibir “OK”, completa el inicio de sesión.

- userInfo: Si la app envía en la URL el código userInfo=[usuario] (siendo usuario el nombre del usuario cuestión), el gestor realiza un SELECT a la tabla registeredemployee que devuelve todos los campos (excepto la contraseña)

- historic: Si la app envía en la URL el código historic=[usuario] (siendo usuario el nombre del usuario cuestión), el gestor realiza un SELECT a la tabla checks que devuelve todos los registros de dicho usuario.

- lastRec: Si la app envía en la URL el código lastRec=[usuario] (siendo usuario el nombre del usuario cuestión), el gestor realiza un SELECT a la tabla checks que devuelve toda la información del último registro de ese usuario.

- getMinutes: Si la app envía en la URL el codigo getMinutes=[usuario];[fecha] (siendo usuario el nombre del usuario cuestión, y la fecha el día que se desea consultar en el formato ‘yyyy-mm-dd’), el gestor realiza un SELECT a la tabla employeehoursperday que devuelve toda la información del registro que cumpla esos parámetros (usuario y día).

- schedule: Si la app envía en la URL el código schedule=[groupID] (siendo groupID el iD del grupo al que pertenece el usuario, el cual se obtiene en la app a través de userInfo), el gestor realiza un SELECT a la tabla schedule que devuelve toda la información del grupo de horarios solicitado.

Con respecto a las peticiones POST, existen 3 dentro de la aplicación Java, y se encargan de insertar información en las tablas de la base de datos. Al recibir una petición POST, lo que analiza la aplicación es el denominado *body,* ya que en los tres casos se emplea la misma URL (/enviaDatos) En el body, separado por comas “,” se introduce desde la app la siguiente información: acción, usuario, información y fecha
Las peticiones POST son las siguientes:

- entry: Si el POST incluye todos los campos necesarios (acción, usuario, información y fecha) siendo acción “entry” y siendo información un string vacío (“”), el gestor realiza un INSERT en la tabla del tipo entry del usuario en la fecha seleccionada.

- exit: Si el POST incluye todos los campos necesarios (acción, usuario, información y fecha) siendo acción “exit” siendo información un string vacío (“”), el gestor realiza un INSERT en la tabla del tipo entry del usuario en la fecha seleccionada.

- updateMinutes: Si el POST incluye todos los campos necesarios (acción, usuario, información y fecha) siendo acción “updateMinutes” siendo información la cantidad de minutos, el gestor realiza un UPDATE en la tabla de employeehoursperday en el que suma la nueva cantidad a la cantidad existente de minutos en la tabla en el día seleccionado.

3.2.2. Entorno de Microsoft Dynamics.

3.3. Aplicación móvil.

3.3.1. Interfaz, navegación y funciones de cada pantalla de la aplicación.

- **Login:** (Ver Figura nº10) La pantalla que recibe al usuario al iniciar la app. En ella se solicitan el nombre de usuario y la contraseña. Ya que los usuarios se crearán desde Dynamics no viene incluida la opción de registrarse.

- **Check**: (Ver Figuras nº11, 12 y 13)La pantalla principal. Será la primera pantalla que visualiza el usuario tras iniciar sesión. Si el usuario ha accedido a otra pantalla, puede volver a esta mediante la barra inferior de navegación, pulsando sobre el primer icono con el dibujo de un *checkmark*. En ella se encuentra el botón que sirve para entrar y salir de la jornada laboral. La acción que realiza el botón depende del estado actual del usuario. Es decir, si el usuario ha marcado una entrada, la acción del botón será marcar salida y viceversa. Para realizar la entrada se realizan varias comprobaciones.

    - Que el usuario se encuentre dentro de su turno asignado. Esto incluye tanto el horario del día, como que el día en cuestión sea o no laborable o festivo.
    - Que el último registro sea una salida.
        - Aunque la acción del botón depende del estado del usuario, esta comprobación sirve de contramedida ante posibles errores a la hora de solicitar a la base de datos dicho estado.
    - Que la última salida haya sido hace más de 3 minutos.
        - Esta comprobación activa una función especial que en lugar de marcar la entrada, elimina la última salida, de forma que se sigue contabilizando desde la entrada anterior. Diseñada para evitar entradas y salidas innecesarias ante un error por parte del usuario.
    - Que no haya entradas de días anteriores sin su correspondiente salida.
        - En este caso se informará al usuario de que no es posible realizar una nueva entrada hasta que actualice el estado de dicha entrada sin salida, abriendo un diálogo en el que si el usuario pulsa continuar, se añadirá una salida en dicho día a la hora a la que figura su salida en el horario que tiene asignado.

- **Historic**: (Ver Figuras nº14, 15 y 16) La segunda pantalla. El usuario accede a ella desde la barra inferior de navegación, pulsando el icono con las tres líneas horizontales. Desde esta pantalla el usuario puede consultar su registro completo de entradas y salidas, con la posibilidad de filtrar los resultados por día, a través de un modal de selección en el que el usuario deberá pulsar en un calendario el día que quiere consultar. Al filtrar un día aparecerá también un componente de texto que informará de las horas realizadas ese día en concreto. Los registros se muestran en un componente *flatlist*, que independientemente de cuántos registros existan se mostrarán en la pantalla en orden con posibilidad de hacer “scroll” hacia abajo para visualizar resultados situados más abajo en la lista. Este componente dispone también de una función “pull to refresh” que, como su nombre en inglés explica,

arrastrando la lista desde el primer resultado hacia abajo (como intentando visualizar resultados por encima del primero) aparecerá un icono que le indicará al usuario la acción que está realizando. Dicha acción vuelve a consultar los resultados a la base de datos para refrescar los registros si fuese necesario.

- **Calendar**: (Ver Figura nº17) La tercera pantalla. El usuario accede a ella desde la barra inferior de navegación, pulsando el icono con el dibujo de un calendario. Esta pantalla no ofrece ningún tipo de interacción con el usuario. Únicamente muestra el horario semanal definido en la base de datos para ese usuario.

3.3.2. Librerías empleadas:

El framework de React Native permite importar distintos componentes en función de las necesidades de la aplicación. Cuenta con una amplia gama de componentes y funciones nativas tales como botones, campos de texto, campos de inserción de texto.. etc. Las características propias de react como el caso del useContext o el useState, o los componentes de navegación son imprescindibles para manejar la información y almacenar valores en sus correspondientes variables durante la ejecución y para navegar entre las distintas pantallas de nuestra aplicación. Por otra parte, los componentes como Button o TextInput no siempre ofrecen el resultado deseado ya que son por lo general bastante básicos y permiten poca customización. Existen gran variedad de librerías de terceros con componentes más customizables que también podemos importar a nuestro proyecto. A continuación enumeraré las más relevantes que he empleado:

- React-Navigation/Stack: Permite la navegación en stack. Este tipo de navegación coloca las pantallas que se ubican dentro de este tipo de navegación apiladas una encima de otra. En mi caso la he empleado para colocar la pantalla de login encima de las demás.
- React-Navigation/Bottom-Tabs: Permite la navegación en Bottom tab. Las pantallas que se incluyan dentro de este tipo de navegación aparecerán en la barra inferior representadas por un icono.
- React-Navigation/Native: Es un elemento necesario para incorporar tipos de navegación ya que permite definir el límite del tipo de navegación deseado y qué pantallas introducir en él.
- React-Native-Paper: Es una librería que amplía las posibilidades de customización de los elementos básicos de react native sustituyéndolos por componentes similares.
- React-Native-Vector-Icons: Permite importar iconos de múltiples estilos y colecciones en formato vectorial.
- React-Native-Figma-Squircle: Importa una figura geométrica, el “Squircle”, una combinación de círculo (circle) y cuadrado (square) muy estética que he utilizado para el botón principal.
- Moment: Permite trabajar con unidades de tiempo, convertirlas a string, aplicarles múltiples formatos y definir el tiempo actual.
- DateTimePicker: Ofrece varios formatos de selección de fecha (spinner, calendario, reloj…) y permite almacenar la selección en una variable.

3.3.3. Aspectos del diseño:

Fig.10. Login.

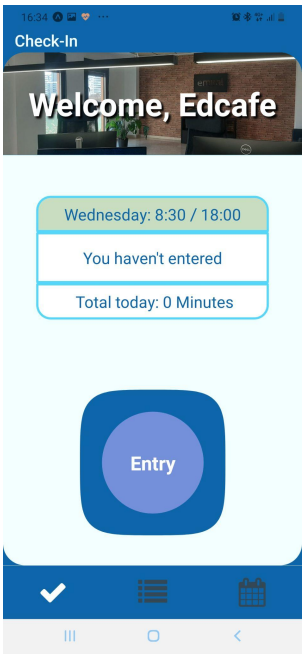


Fig.11. Check 1

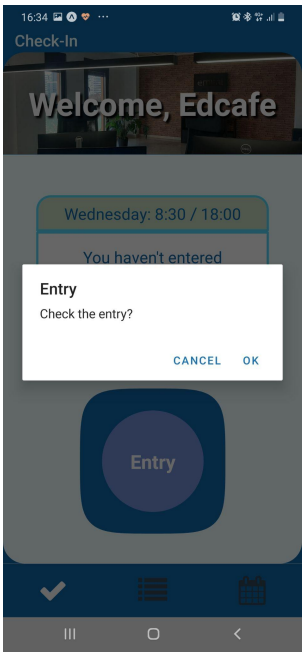


Fig.12. Check 2

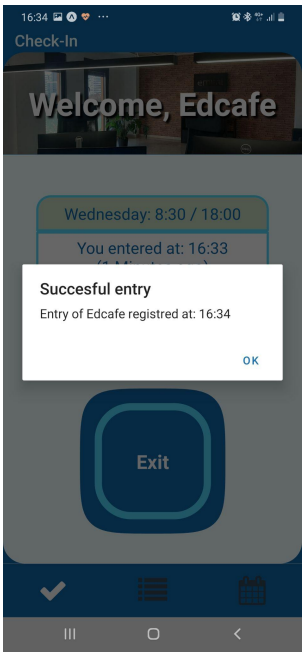


Fig.13. Check 3

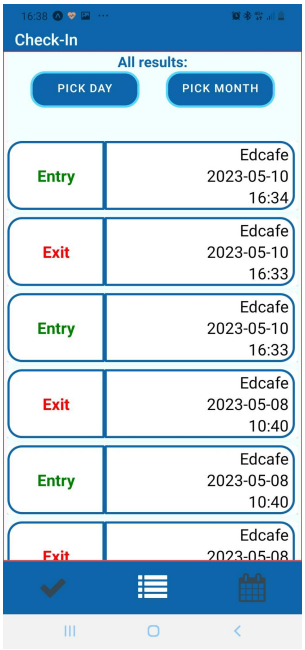


Fig. 14. Historic 1

Fig.15. Historic 2

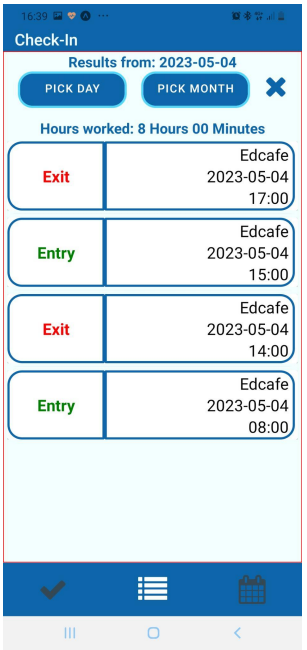


Fig.16. Historic 3

Fig.17. Calendar

3.3.4. Mockups y Diagramas.

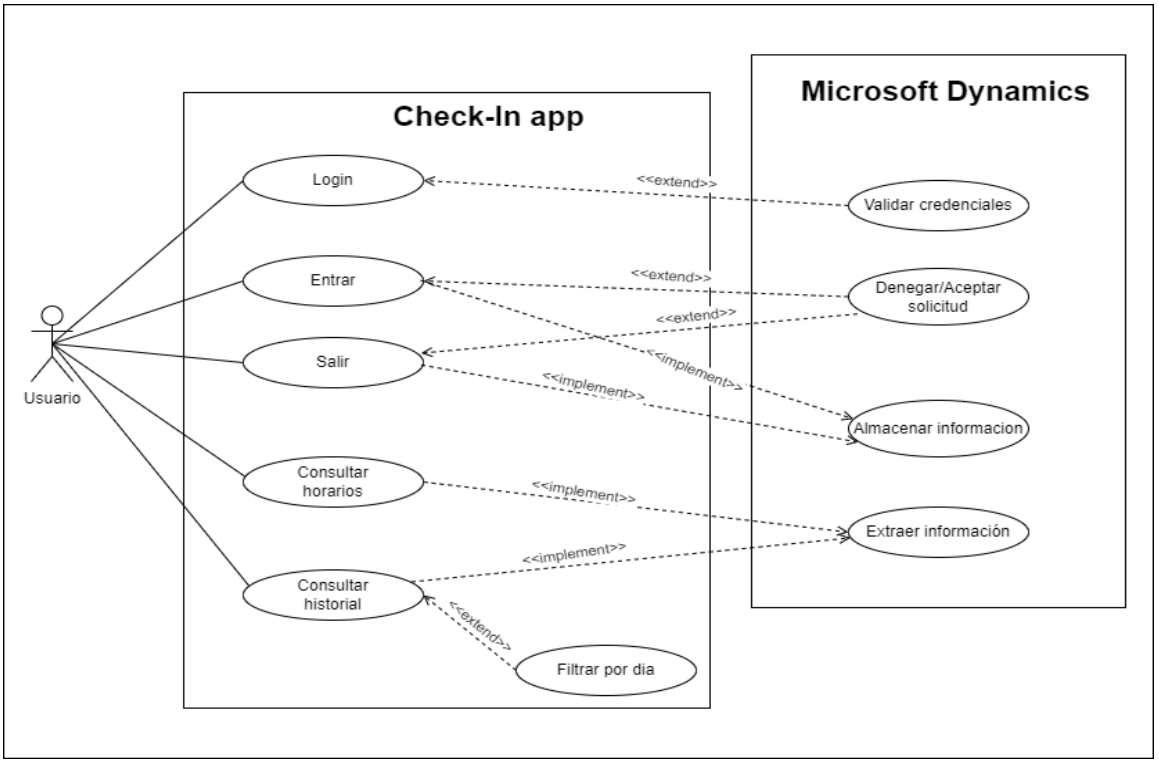


Fig. 18. Esquema de Caso de uso para Usuario de la App Móvil.

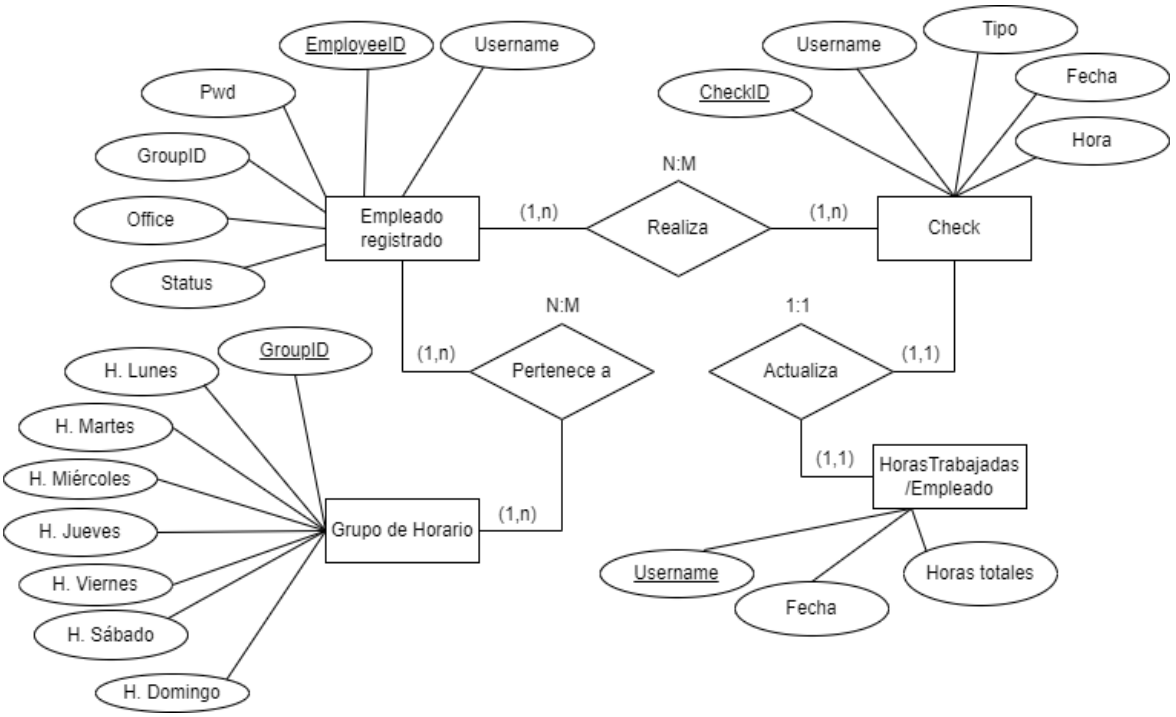


Fig. 19. Esquema relacional de las tablas de la base de datos.

3.3.5. Aspectos del back-end:

4. CONCLUSIONES:

4.1. Problemas/dificultades en el desarrollo.

4.2. Futuro de la aplicación.