



BILBOKO
INGENIARITZA
ESKOLA
ESCUELA
DE INGENIERÍA
DE BILBAO

TicTacker

Memoria del Proyecto

Desarrollo Avanzado de Software

4º Curso

<https://github.com/ffernandezco/TicTacker>

Grado en Ingeniería Informática de Gestión y Sistemas de
Información

Francisco Fernández Condado

22 de abril de 2025

Índice

1	Introducción	2
1.1	Instalación y configuración	2
2	Elementos utilizados y funcionalidades	2
3	Clases, estructura, bases de datos y ficheros del servidor usados en la aplicación	4
3.1	Clases	4
3.1.1	Activities	4
3.1.2	Fragments	4
3.1.3	Adaptadores	5
3.1.4	Clases generales	5
3.1.5	Helpers	5
3.1.6	Receivers	6
3.1.7	Services	6
3.1.8	Eventos	6
3.1.9	Widgets	6
3.1.10	Workers	6
3.2	Base de datos	7
3.3	Ficheros PHP usados en el servidor	8
4	Manual de usuario de TicTacker	9
4.1	Inicio de sesión y registro de usuarios	9
4.2	Ajustes de la aplicación	10
4.3	Fichajes e historial	12
4.4	Gestión de perfiles	14
4.5	Envío de notificaciones mediante FCM	15
5	Dificultades encontradas al realizar el proyecto	16
6	Fuentes y referencias utilizadas	17

1. Introducción

TicTacker es una aplicación móvil para Android basada en Java + Gradle que permite ayudar a empresas y a autónomos a seguir un registro de su jornada laboral. Para ello, permite realizar un seguimiento completo de las horas y minutos trabajados cada día, permitiendo además configurar una jornada laboral con sus correspondientes horas y días laborales a fin de conocer las horas extraordinarias realizadas cada día y recibir notificaciones en caso de sobrepasar la jornada descrita. Permite a los empleados registrar su perfil para obtener una experiencia más personalizada y se sincroniza con un servidor PHP para almacenar toda la información en una base de datos MySQL remota. Es además posible personalizar su logotipo y el idioma para adaptarse a las necesidades de la empresa, así como generar ficheros CSV con los fichajes registrados que posteriormente puede importarse si así se desea o configurar recordatorios para fichar a tiempo.

1.1. Instalación y configuración

Para hacer uso de *TicTacker*, puede utilizarse el fichero APK para realizar la instalación. Una vez instalada, será necesario autenticarse, para lo que es posible crear una nueva cuenta usando el asistente o bien utilizar el usuario de demostración haciendo uso de las siguientes credenciales:

- **Usuario:** demo
- **Contraseña:** demo

Conexión a Internet

Esta edición de *TicTacker* realiza conexiones con el servidor de Internet <http://ec2-51-44-167-78.eu-west-3.compute.amazonaws.com/ffernandez032/WEB/>. Se recomienda verificar que se puede acceder a dicha dirección a través del navegador del dispositivo antes de comenzar a utilizar la aplicación.

A partir de entonces, podrán comenzar a utilizarse las funcionalidades de la aplicación.

2. Elementos utilizados y funcionalidades

Esta aplicación es una continuación de la primera versión entregada. Para construirla, se han hecho uso de las siguientes características y elementos propios de Android, además de los detallados en la documentación de dicha versión:

- **Uso de una base de datos remota para el registro y la identificación de usuarios:** la base de datos anterior, además de haberse ampliado, se ha alojado en una base de datos remota a la que se accede desde ficheros PHP contenidos en el servidor ec2-51-44-167-78.eu-west-3.

compute.amazonaws.com. Se permite registrar usuarios e iniciar sesión, se almacenan los fichajes, los datos de los perfiles y la configuración.

- **Integrar los servicios Google Maps u OpenStreetMap y Geolocalización en una actividad:** a la hora de registrar un fichaje, se almacena la ubicación en caso de estar disponible y se guarda en la base de datos. Accediendo al historial de fichajes y seleccionando uno existente, es posible ver, además de los detalles, un pequeño mapa de OpenStreetMap con OSMdroid en el que se detalla la ubicación.
- **Uso de algún content provider para añadir, modificar o eliminar datos:** al guardar un fichaje, ahora se añade también al calendario propio del dispositivo para poder ver todos con mayor detalle. Además de añadir los eventos, en caso de no haber un calendario editable se encarga de crearlo en la cuenta de Google.
- **Captar imágenes desde la cámara, guardarlas en el servidor y mostrarlas en la aplicación:** en la sección de perfil, cada usuario puede elegir si tomar una foto de perfil haciendo uso de la cámara del dispositivo o si prefiere obtenerla desde la galería de imágenes del dispositivo. Para evitar los límites de PHP y saturar el servidor, las imágenes se almacenan en Base64 directamente en la base de datos, con una resolución máxima de 160x160 px. Después se muestran tanto en la vista del perfil como en el menú.
- **Implementación de un servicio en primer plano y gestión de mensajes broadcast durante el servicio:** al comenzar un fichaje, se registra en primer plano (*Foreground*) y se muestra una notificación silenciosa persistente que se actualiza periódicamente para permitir ver el tiempo que ha pasado desde el fichaje. Deja de actualizarse al marcar la salida.
- **Uso de mensajería FCM:** la aplicación cuenta con la mensajería FCM de Google Firebase para permitir la recepción de notificaciones push. Se almacenan los diferentes tokens generados en la base de datos, y desde el servicio web <http://ec2-51-44-167-78.eu-west-3.compute.amazonaws.com/ffernandez032/WEB/notificar.php> es posible enviar una notificación a todos los usuarios¹ de la aplicación.
- **Desarrollar un widget que tenga, al menos, un elemento que se actualice automáticamente de manera periódica:** se ha incluido un widget que permite consultar, de un vistazo, el estado del fichaje (con un botón para fichar o salir), así como las horas restantes en caso de tener un fichaje en curso.
- **Uso de algún servicio o tarea programada mediante alarma:** a través de la configuración, es posible habilitar un recordatorio para fichar. Este recordatorio configura una alarma y, en

¹Se entiende que cada usuario tendrá únicamente un dispositivo, por lo que si se inicia sesión en un nuevo dispositivo se actualiza el token en la base de datos y, por tanto, el dispositivo anterior deja de recibir notificaciones. Se recomienda crear diferentes usuarios nuevos si se van a realizar pruebas con diferentes dispositivos.

caso de que no se haya registrado ningún fichaje para la hora establecida, se envía una notificación programada. Para poder hacer uso de este servicio, será necesario habilitar los permisos correspondientes en la configuración del dispositivo.

3. Clases, estructura, bases de datos y ficheros del servidor usados en la aplicación

TicTacker hace uso de diferentes clases que se comunican entre sí y que utilizan los elementos XML de Android para gestionar la vista correspondiente. Adicionalmente, se hace uso de una base de datos MySQL desde un servidor con PHP para gestionar los usuarios, los fichajes, los perfiles, los tokens de FCM y las configuraciones personalizadas.

3.1. Clases

La aplicación cuenta con diferentes clases Java diferenciadas que dan respuesta a una o varias de las funcionalidades de la aplicación. A continuación se describen las siguientes clases utilizadas, pudiendo ver las relaciones entre las mismas por medio de diagrama de clases de la Figura 1.

3.1.1. Activities

- **MainActivity**: actividad principal que se encarga de gestionar la navegación y la interfaz de usuario principal de la aplicación.
- **LoginActivity**: gestiona la autenticación de usuarios existentes, validando credenciales y guardando la sesión.
- **SignupActivity**: gestiona el registro de nuevos usuarios, incluyendo validación de los datos, que no se repitan nombres de usuario y creación de cuentas junto con un perfil básico a partir de los datos introducidos.
- **SettingsActivity**: ofrece opciones de personalización de la aplicación, como cambiar el idioma, la jornada laboral, el logotipo o los recordatorios de fichaje.
- **NoInternetActivity**: se lanza en el momento en el que se detecta que no se cuenta con conexión a Internet, bloqueando el uso de la app.

3.1.2. Fragments

- **ClockInFragment**: fragment que permite a los usuarios registrar sus fichajes, mostrando el estado actual y el tiempo trabajado.

- **FichajeDetailsFragment:** permite a los usuarios ver los detalles de un fichaje concreto seleccionado del historial, dejando ver la ubicación en un mapa de OpenStreetMap en caso de estar disponible.
- **HistoryFragment:** fragment que muestra el historial de fichajes del usuario, permitiendo exportar e importar datos por medio de ficheros CSV.
- **SettingsFragment:** fragment que permite a los usuarios ajustar sus preferencias, como las horas de trabajo semanales y los días laborables.
- **EditFichajeDialog:** dialog fragment utilizado a la hora de editar la hora de entrada y/o salida de un fichaje existente en la aplicación.

3.1.3. Adaptadores

- **FichajeAdapter:** utilizado para mostrar listas de fichajes a través de un *RecyclerView*.

3.1.4. Clases generales

- **TicTacker:** clase principal de la aplicación que se encarga de aplicar las preferencias de usuario al iniciar, así como de instanciar OSMdroid para poder utilizar los mapas de OpenStreetMap.
- **Fichaje:** entidad que representa un fichaje con los atributos correspondiente, usado para instanciar objetos fácilmente con su fecha, hora de entrada, hora de salida, latitud, longitud y nombre de usuario para almacenar en la base de datos.
- **WorkTimeCalculator:** clase que proporciona métodos para calcular el tiempo trabajado y el tiempo restante.
- **ApiClient:** clase utilizada para realizar las consultas POST y GET a la API del servidor, permitiendo realizar modificaciones sobre la base de datos.
- **NetworkConnectivityChecker:** clase utilizada para comprobar si se tiene conexión a Internet al ser requerida para funcionar. En caso de no tener conexión, se lanza *NoInternetActivity*.
- **UserProfile:** clase utilizada para gestionar los perfiles de usuario, almacenando valores como el nombre, el email o la foto de perfil.

3.1.5. Helpers

- **DatabaseHelper:** gestiona la base de datos MySQL realizando llamadas al *DatabaseWorker* que, a su vez, utiliza *ApiClient* para llamar a la API y hacer cambios en el servidor.
- **NotificationHelper:** clase que gestiona la creación y envío de notificaciones.

3.1.6. Receivers

- **BootReceiver**: clase auxiliar utilizada para reconfigurar las alarmas de los recordatorios de fichajes en caso de que el dispositivo se reinicie.
- **ClockInReminderReceiver**: clase que interpreta y gestiona las alarmas enviadas por la clase *ClockInReminderService* y que se encarga de generar las notificaciones de recordatorios de fichajes.

3.1.7. Services

- **ClockInReminderService**: clase utilizada para enviar alarmas para recordar al usuario que inicie un fichaje si se supera una hora determinada y no se ha fichado.
- **ForegroundTimeService**: servicio empleado para gestionar la actividad de fichaje en primer plano (*Foreground*), lanzando una notificación silenciosa persistente para informar del estado del fichaje.
- **MyFirebaseMessagingService**: servicio utilizado para registrar los tokens FCM de Firebase y recibir notificaciones enviadas a distancia.

3.1.8. Eventos

- **FichajeEvents**: clase que gestiona los eventos relacionados con los fichajes, permitiendo notificar cambios a los listeners registrados.

3.1.9. Widgets

- **TicTackerWidget**: clase utilizada para gestionar todos los aspectos relacionados con el widget de la aplicación.

3.1.10. Workers

- **DatabaseWorker**: worker que envía y recibe las consultas en formato JSON realizadas a la API para realizar cambios sobre la base de datos.
- **WorkTimeCheckWorker**: worker que comprueba periódicamente el tiempo trabajado y envía notificaciones al usuario si alcanza su jornada laboral.

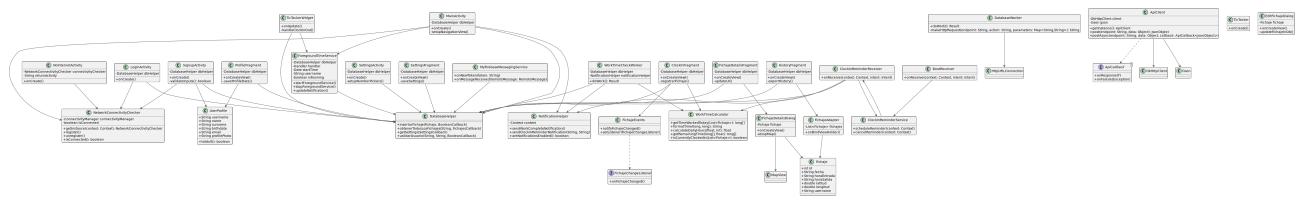


Figura 1: Diagrama de clases del proyecto

3.2. Base de datos

Además de las *SharedPreferences*, para gestionar la persistencia de la aplicación se hace uso de una base de datos de tipo MySQL controlada desde un servidor PHP, que tal y como se muestra en el diagrama de la Figura 2, contiene las siguientes tablas:

- **Tabla fichajes:** almacena la información sobre los fichajes de los usuarios.
 - **Atributos principales:** id, fecha, hora_entrada, hora_salida, latitud, longitud, username.
 - **Índices:** id es un campo único y clave primaria.
 - **Tabla users:** almacena los usuarios registrados en la aplicación.
 - **Atributos principales:** id, username, password.
 - **Índices:** username es un campo único indexado para prevenir duplicados.
 - **Tabla settings:** almacena las configuraciones de horas de trabajo y recordatorios.
 - **Atributos principales:** id, weekly_hours, working_days, reminder_enabled, reminder_hour, reminder_minute.
 - **Índices:** id es un campo único y clave primaria.
 - **Tabla fcm_tokens:** almacena los tokens de notificaciones push para cada usuario.
 - **Atributos principales:** id, username, token, created_at, updated_at.
 - **Índices:** id es un campo único y clave primaria.
 - **Tabla user_profiles:** almacena información adicional del perfil del usuario.
 - **Atributos principales:** id, username, name, surname, birthdate, email, profile_photo.
 - **Índices:** id es un campo único y clave primaria.

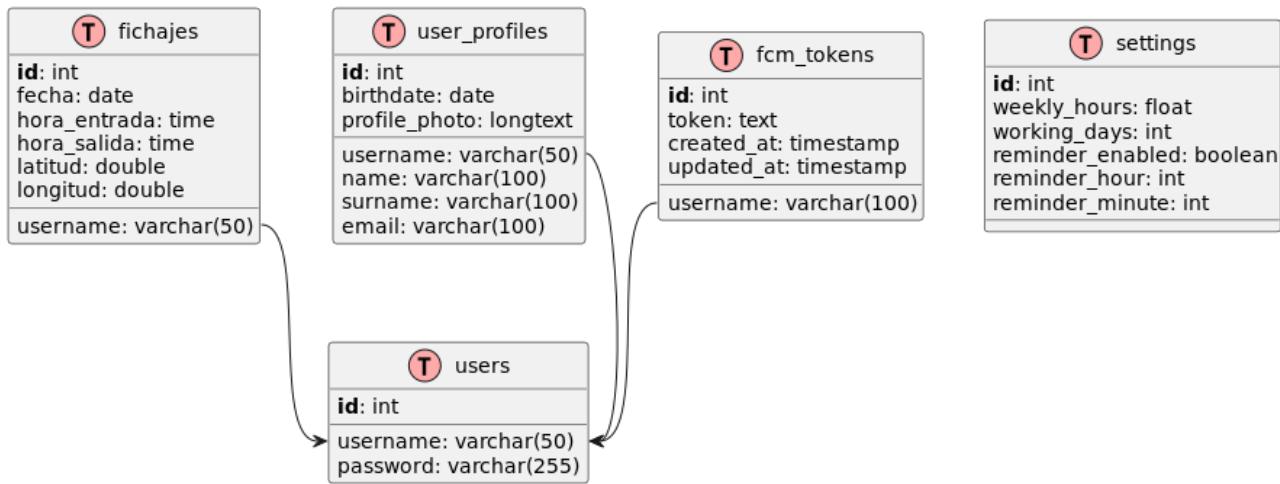


Figura 2: Diagrama de la base de datos usada del proyecto

3.3. Ficheros PHP usados en el servidor

Para ejecutar consultas sobre la base de datos MySQL mencionada anteriormente, así como realizar los correspondientes envíos de la mensajería FCM, se han cargado a través de SFTP los siguientes ficheros en la dirección de Internet <http://ec2-51-44-167-78.eu-west-3.compute.amazonaws.com/ffernandez032/WEB/>, disponibles para su consulta en el directorio server del repositorio de GitHub:

- **db_connect.php**: recoge las credenciales de conexión a la base de datos. Por motivos de seguridad, se han ocultado algunos campos en el repositorio de GitHub. Modificando los ficheros y las direcciones de la conexión es posible usar otro servidor si así se desea.
- **auth_user.php**: verifica que un usuario y contraseña coinciden con el *hash* almacenado en la base de datos para permitir su autenticación.
- **fcm_tokens.php**: se encarga de almacenar los tokens FCM de Firebase y de actualizar si es necesario el usuario al que está asociado cada uno.
- **fichajes.php**: gestiona todas las operaciones relacionadas con los fichajes, incluyendo añadirlos, modificarlos o eliminarlos.
- **firebase_service_account.json**: fichero extraído del gestor de cuentas de servicio IAM de Google que contiene los datos necesarios para enviar mensajes usando la mensajería Cloud Messaging de Firebase. Por motivos de seguridad, no se ha incluido en el repositorio de GitHub.
- **jwt_utils.php**: adaptación compilada del repositorio [firebase/php-jwt](https://github.com/firebase/php-jwt) para gestionar la autenticación de Firebase desde PHP.

- **notificar.php:** habilita una interfaz HTML que permite enviar una notificación a todos los usuarios con un token FCM almacenado en la base de datos. Está disponible de forma pública en la dirección <http://ec2-51-44-167-78.eu-west-3.compute.amazonaws.com/ffernandez032/WEB/notificar.php>.
- **profile.php:** gestiona todas las operaciones relacionadas con los perfiles de usuario, como modificar o añadir el nombre, los apellidos, la foto de perfil o el correo electrónico.
- **send_notification_v1.php:** permite enviar notificaciones a los usuarios a través del token usando Firebase Cloud Messaging (FCM).
- **settings.php:** gestiona todas las operaciones relacionadas con las configuraciones, como la configuración de la jornada o el envío de notificaciones.
- **users.php:** gestiona todas las operaciones relacionadas con los usuarios, como añadir un usuario o modificar la contraseña asociada.

4. Manual de usuario de TicTacker

Una vez se ha realizado la instalación de la aplicación por medio del fichero APK, será posible comenzar a utilizarla. Es necesario contar con una conexión a Internet para poder usar la misma. Al lanzarla por primera vez, en función de la versión del sistema operativo Android, se lanzarán alertas para conceder permisos de cara al envío de notificaciones, la ubicación y la gestión de archivos de imagen. Es recomendable habilitar todos ellos mientras la app esté en uso para evitar problemas de funcionamiento y aprovechar todas las características.

4.1. Inicio de sesión y registro de usuarios

Tras aceptar los permisos, se mostrará la pantalla de inicio de sesión representada en la Figura 3a. Es posible hacer uso de la cuenta de demostración, usando “demo” como usuario y contraseña, si bien también puede pulsarse sobre el enlace de registro para dar de alta un nuevo usuario, como representa la Figura 3b. En caso de elegir esta opción, deberá indicarse un nombre de usuario que no puede existir ya en la base de datos, así como una contraseña con 4 o más caracteres que, por motivos de seguridad, deberá introducirse dos veces. También será necesario introducir el nombre, un correo electrónico válido y la fecha de nacimiento, que debe ser superior a los 18 años.

La pantalla de bienvenida de la Figura 3c ilustra una situación de fichaje en curso. La navegación por medio de los distintos apartados de la aplicación se realiza presionando sobre el botón de las tres líneas que se muestra en la esquina superior izquierda. La aplicación presenta tres áreas separadas sobre las que se darán más detalles a continuación:

- **Fichar:** permite consultar el estado de un fichaje en curso o fichar a medida.

- **Historial:** ofrece información sobre los fichajes almacenados, además de permitir exportar e importar datos.
- **Ajustes:** permite definir aspectos de configuración a nivel de la aplicación. Para obtener la mejor experiencia de usuario, se recomienda comenzar accediendo a esta sección.

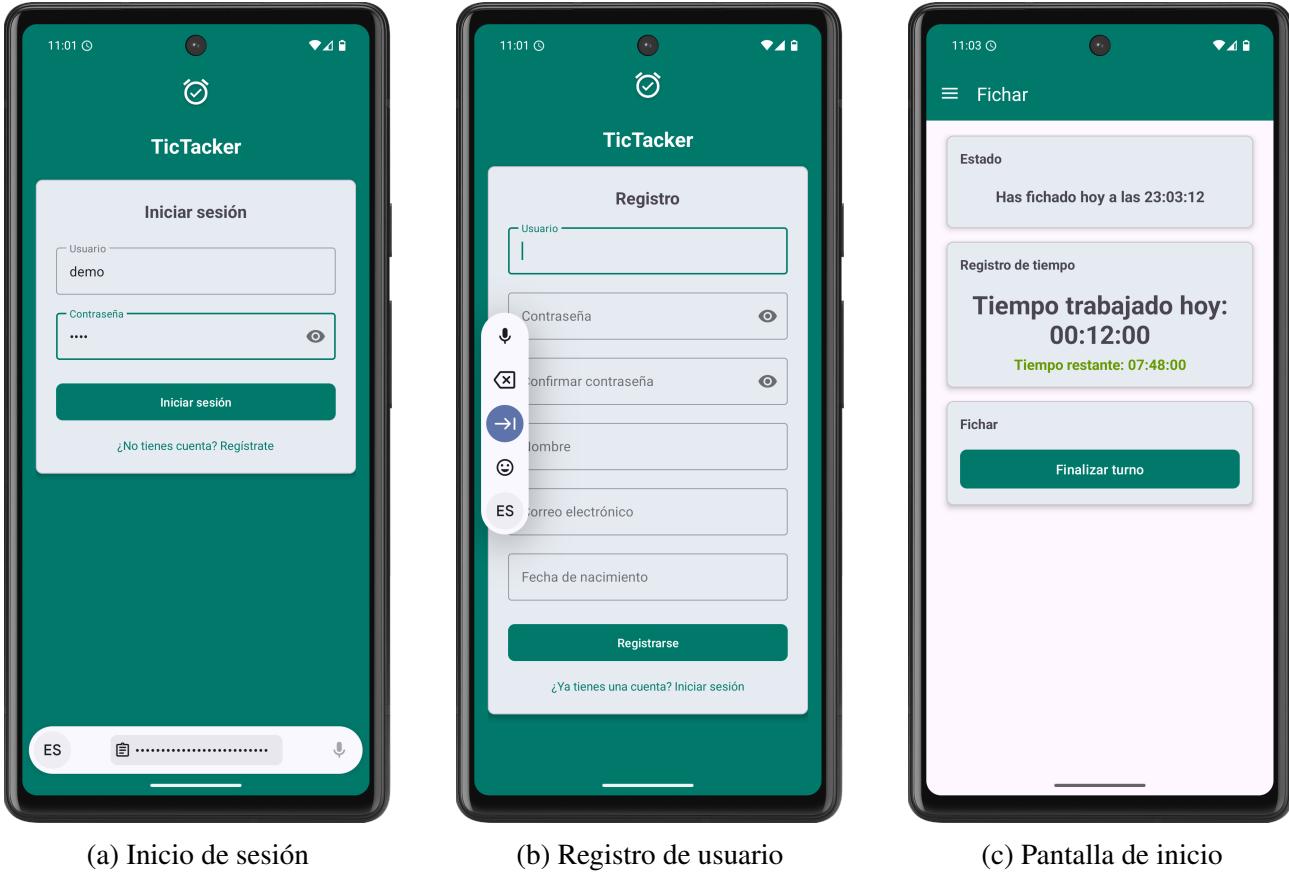


Figura 3: Pantalla de bienvenida e inicio de sesión

4.2. Ajustes de la aplicación

Pulsando sobre la sección de “Ajustes” en el menú de la aplicación, se accederá a la configuración. Es importante presionar sobre el botón de “Guardar” para que se refresque la vista de la aplicación y se puedan ver los cambios realizados de forma correcta si se modifica alguno de los campos.

Para obtener la mejor experiencia posible, se permite elegir, tal y como se muestra en la Figura 4a, entre Español e Inglés como idiomas, siendo suficiente con elegir en el desplegable el preferido para configurarlo como el idioma por defecto para toda la aplicación. A continuación, el usuario deberá

completar los datos de su jornada laboral, especificando los días laborables y las horas que se trabaja a la semana usando los controles. Esta información permite a la aplicación calcular las horas que se trabajará cada día, así como enviar notificaciones en caso de superar la previsión diaria y mostrar un cuadro de diálogo si se trata de salir antes de lo previsto. También se puede configurar una alerta para recordar fichar, de modo que si se alcanza dicha hora y no se ha registrado ningún fichaje, se recibirá una notificación. Esta funcionalidad hace uso de las alarmas programadas de Android, por lo que funciona incluso aunque la app esté cerrada, pero es preciso habilitar el permiso correspondiente.

Sincronización con Internet

Como la idea es que cada empresa con interés en la aplicación tenga su propia instancia de la misma, y se entiende que todos los empleados serán iguales a juicio de la empresa, algunos de los campos de configuración son sincronizados con Internet y, por tanto, realizar un cambio en un dispositivo hace que se modifique para todos los usuarios.

De forma opcional, es posible incluir una imagen como logotipo, que se mostrará en el menú de la aplicación tal y como ilustra la Figura 4c. Para ello, basta con pulsar sobre el botón de “Cambiar” y elegir el nuevo logotipo en el explorador, como muestra la Figura 4b. Debe ser un fichero de imagen, y se recomienda que sea un PNG con fondo transparente y un color claro para mejor visibilidad. En caso de querer restaurar el logotipo por defecto, puede usarse el botón de “Quitar”.

De forma adicional, esta pantalla también permite cerrar sesión para cambiar de usuario, lo que nos llevaría directamente a la pantalla de inicio de sesión, así como eliminar los datos de fichaje almacenados, lo que no se recomienda salvo que se hayan exportado previamente.

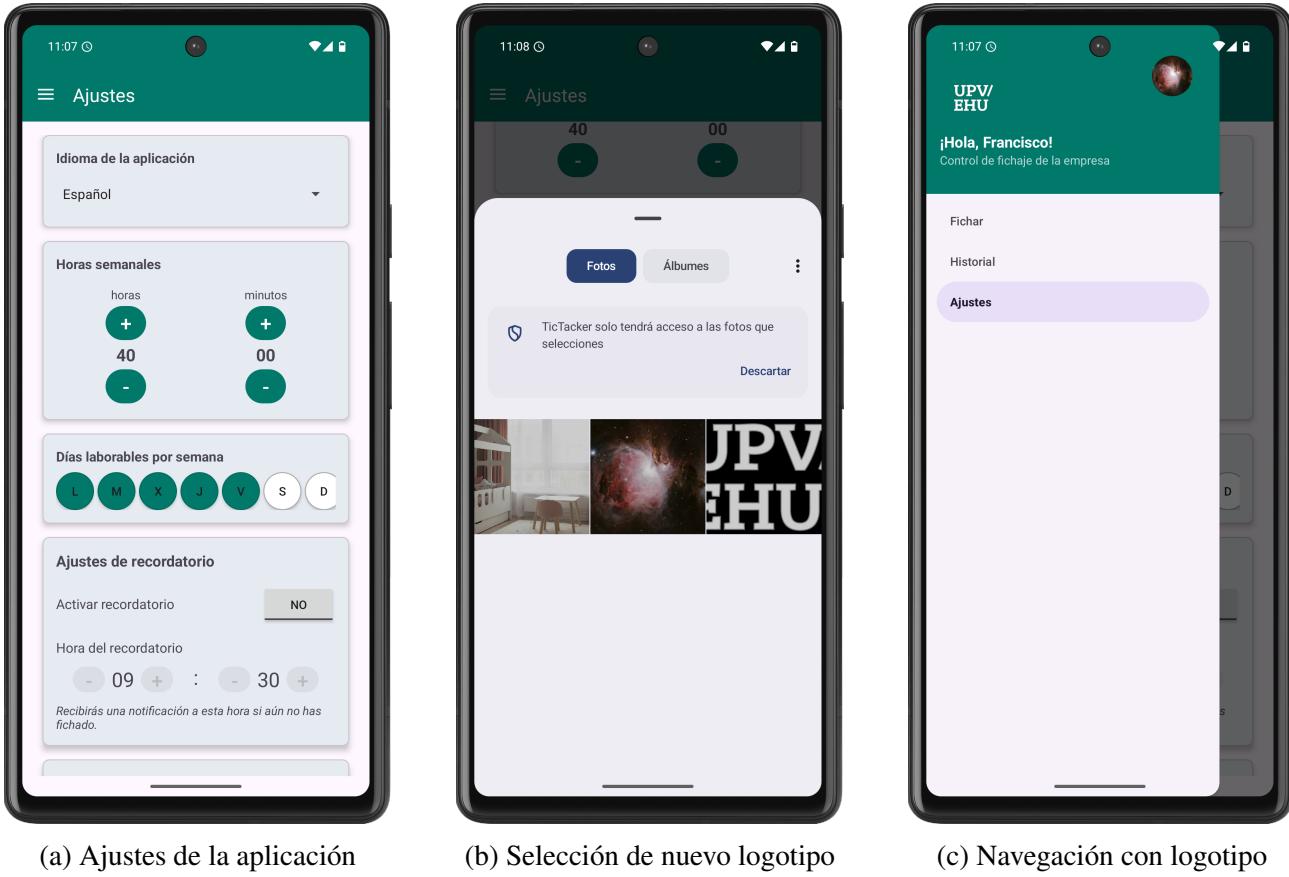


Figura 4: Ajustes de la aplicación

4.3. Fichajes e historial

Accediendo a la pantalla para fichar, es posible consultar de un vistazo los datos de un fichaje activo, tal y como muestra la Figura 5a. Es posible marcar una entrada y una salida por medio del botón asociado de forma sencilla, y consultar la hora, minuto y segundo exacto en el que se fichó, así como la duración actual, el tiempo restante para alcanzar la jornada o, en su defecto, las horas extraordinarias realizadas. Al llegar a las horas estimadas, se recibe una notificación si la aplicación está activa.

Además, durante el tiempo de fichaje, como se gestiona en primer plano, se muestra una notificación persistente silenciosa que permite conocer en tiempo real el tiempo durante el que se ha fichado. Además, tal y como se muestra en la Figura 5d, se ha incluido también un widget que se actualiza de forma dinámica y que permite fichar o salir con un solo clic, o bien consultar el tiempo fichado y restante. Añadir el widget a la pantalla de inicio varía según el dispositivo usado.

Una vez guardado el fichaje, desde el menú es posible acceder al historial que muestra la figura 5b. En él, puede verse un registro de todos los fichajes almacenados en la aplicación, mostrándose

primero el más reciente. Si se pulsa sobre uno de ellos, como se muestra en la Figura 5c, es posible ver el detalle completo, consultando la fecha y hora exactas de salidas y entradas, así como las coordenadas geográficas desde las que se produjo el fichaje en caso de haber otorgado permisos para acceder a la ubicación (en su defecto, se mostrará que no está disponible), y un mapa de OpenStreetMap que permite ver dicha ubicación.

En la parte inferior de la ventana de detalles, existen opciones para editar un fichaje guardado, que permite modificar la hora de entrada o salida en caso de haberse identificado algún error, así como la opción de ver en un mapa la ubicación registrada. Al pulsar sobre esta opción, se abre la aplicación de mapas asociada del dispositivo (Google Maps en gran parte de las ocasiones).

Indicar también que, al finalizar un fichaje, se añade también un evento al calendario para reflejarlo, en el que se detalla el usuario y el tiempo de la jornada. En la mayoría de casos, se añadirá al calendario predeterminado pero, en caso de no haber ninguno disponible con permisos de escritura, se creará un nuevo calendario con el nombre “TicTacker”. Como en las últimas versiones de Android este *content provider* va ligado con las cuentas, es posible que el teléfono solicite acceso a los Contactos para realizar la tarea.

De forma adicional, la parte inferior de la aplicación permite importar o exportar una hoja de fichajes. Al presionar sobre exportar, se genera un fichero CSV con los fichados registrados, para el que deberá elegirse una ubicación dentro del dispositivo en la que guardarlos, mientras que si se selecciona importar, se puede elegir un fichero generado previamente por la aplicación o de forma manual con el mismo formato. Al hacerlo, se almacenarán los detalles en la base de datos y se podrá ver en el historial junto al resto de datos.

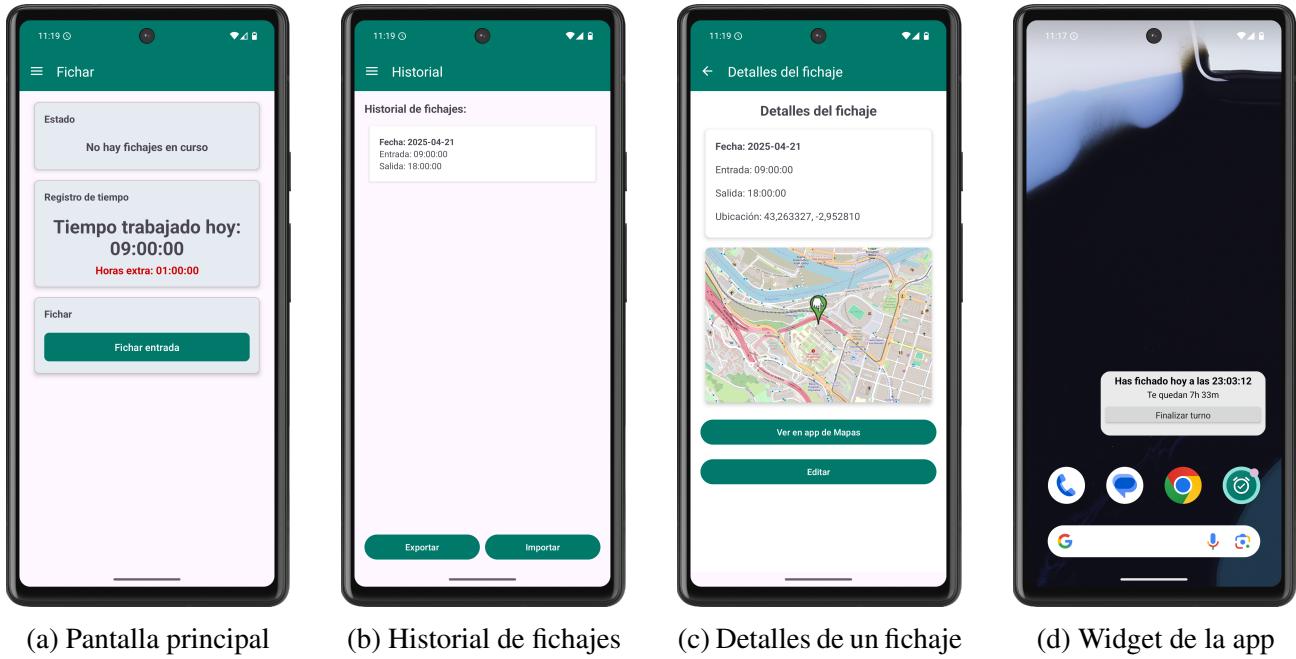


Figura 5: Uso general de la aplicación

4.4. Gestión de perfiles

Con el fin de dar una mejor experiencia de uso, es posible modificar el perfil de usuario. Desde el menú lateral, presionando sobre el ícono de la parte superior derecha, es posible acceder a la configuración del perfil. Como se aprecia en la Figura 6a, puede establecerse el nombre, los apellidos, el correo electrónico (debe seguir la estructura *nombre@dominio.extension* para que se reconozca como correcto) y la fecha de nacimiento (debe ser superior a los 18 años), así como asignar una foto de perfil, que puede hacerse directamente desde la app o elegirse desde el propio dispositivo. Deben completarse todos los campos para generar el perfil completo de usuario.

Los datos proporcionados se usarán para mejorar la experiencia de usuario. La imagen de perfil se mostrará en el menú, así como el nombre elegido, tal y como representa la Figura 6b. Puede combinarse con las opciones de la sección de ajustes para conseguir una mayor personalización.

Desde esta misma sección, existe también la opción de cambiar la contraseña, siendo necesario introducir la contraseña actual del usuario y dos veces la nueva contraseña. En cualquier caso, la contraseña debe tener más de 4 caracteres para que se considere válida. También se permite cerrar sesión.

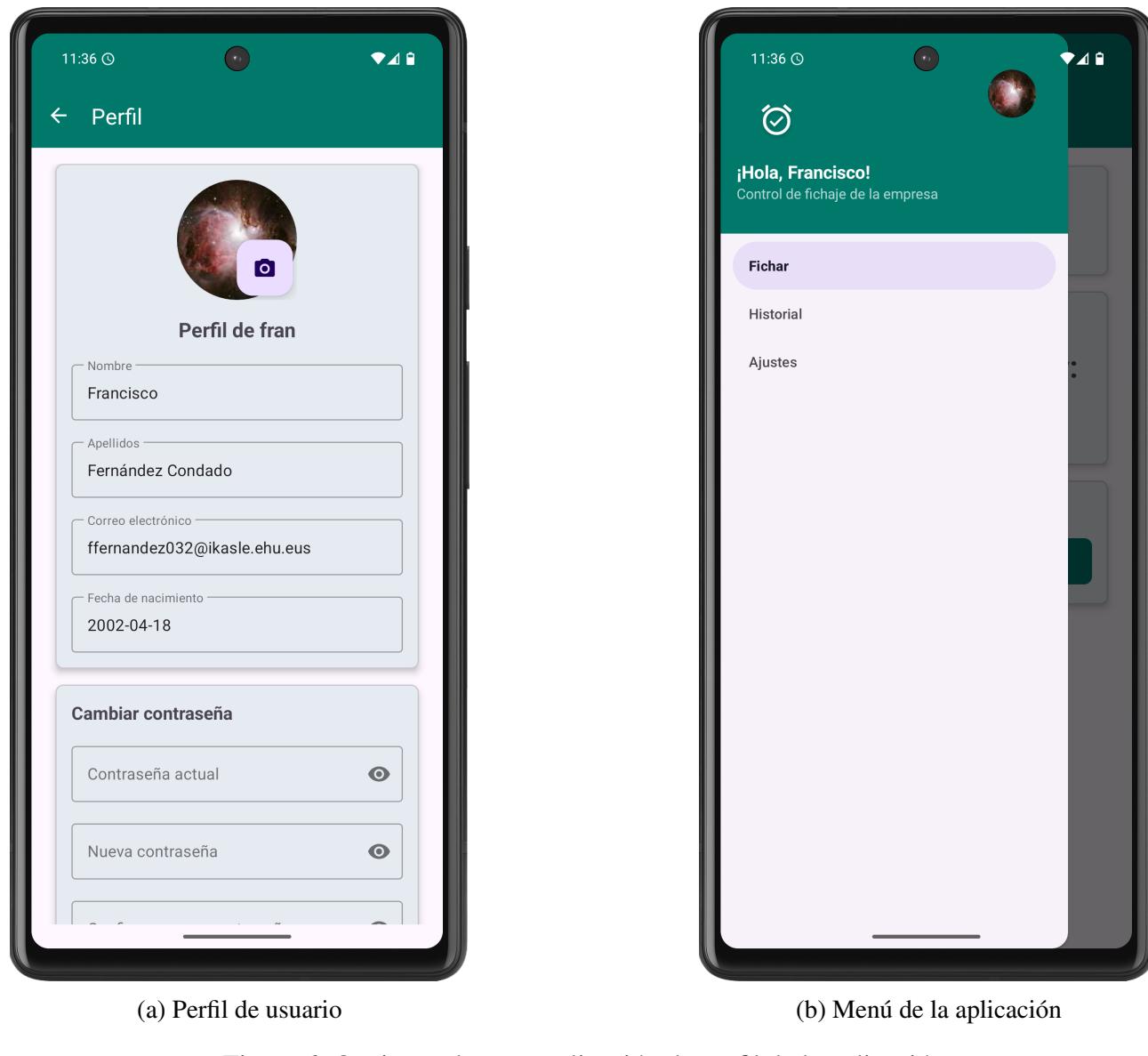


Figura 6: Opciones de personalización de perfil de la aplicación

4.5. Envío de notificaciones mediante FCM

La aplicación cuenta con el servicio de recepción de notificaciones push a través del servicio Cloud Messaging de Firebase, lo que quiere decir que existe la posibilidad de enviar notificaciones de forma remota a los usuarios. Para ello, se recoge un token cada vez que la aplicación es instalada en

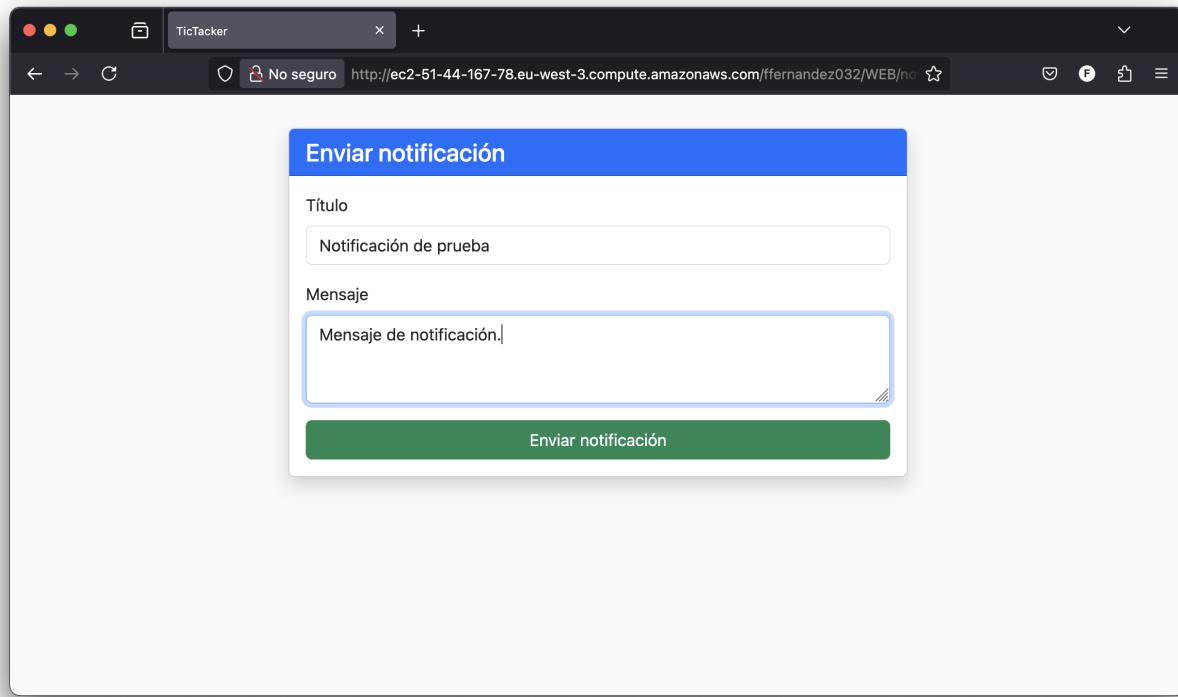


Figura 7: Pantalla de envío de notificación remota

un nuevo dispositivo y se inicia sesión en la misma², que se almacena en la base de datos.

Tal y como se muestra en la Figura 7, accediendo al fichero `notificar.php` del servidor, disponible en la dirección de Internet `http://ec2-51-44-167-78.eu-west-3.compute.amazonaws.com/ffernandez032/WEB/notificar.php`, es posible enviar una notificación de forma remota a todos los usuarios. Las notificaciones se reciben tanto si la aplicación está abierta como en caso contrario, por lo que es una buena vía de comunicación para mensajes urgentes.

5. Dificultades encontradas al realizar el proyecto

A lo largo del proyecto, han surgido múltiples dificultades. En primer lugar, el rendimiento de Android Studio y del emulador no es el esperado, lo que ha dificultado algo la tarea. Además, tuve problemas a la hora de gestionar los permisos y cómo se realizaba el conteo del tiempo, pues había

²Por tanto, si se inicia sesión en un nuevo dispositivo con un nuevo usuario, el nuevo token quedará asignado a dicho usuario y, por tanto, se dejarán de recibir notificaciones en el dispositivo anterior. Se entiende que cada empleado utilizará un solo teléfono móvil para fichar.

que pulsar varias veces para que se registrase una entrada o una salida, y las notificaciones no siempre se enviaban o, peor aún, se enviaban cada segundo obligando a detener la aplicación, algo que fue corregido.

También ha resultado algo complicado implementar la lógica del inicio de sesión, principalmente porque la aplicación se creó siendo pensada para un solo usuario y, por tanto, ha sido necesario adaptar cada método para que funcione correctamente con múltiples usuarios, siendo necesario acceder y añadir un parámetro más en cada consulta. Tampoco ha sido sencillo almacenar las imágenes de perfil en el servidor debido a las limitaciones del mismo, aunque se ha solventado comprimiendo las imágenes y reduciendo su tamaño.

Además, ha resultado algo complicado gestionar el salto de una alarma programada a una hora exacta, principalmente por los permisos y porque en ocasiones se borran. El uso de FCM tampoco ha sido sencillo al haber modificado Google los métodos de acceso para Firebase, requiriendo usar su nueva autenticación. Por último, el *content provider* del calendario tampoco ha sido fácil, debido a que es necesario acceder a las Cuentas para verificar que puede editarse un calendario para añadir un evento.

6. Fuentes y referencias utilizadas

A la hora de realizar el proyecto, ha sido preciso realizar consultas a diferentes sitios de Internet, bien para resolver dudas o para obtener información y guías sobre cómo realizar diferentes acciones. A continuación se exponen las fuentes y referencias más relevantes que se han utilizado:

- Apuntes de la asignatura “Desarrollo Avanzado de Software”. Iker Sobrón Polancos. eGela UPV/EHU.
- Android Developers Guide.
- Firebase Documentation.
- JSON PHP. W3Schools.
- OSMDroid.
- jwt.
- php-jwt.
- Bootstrap.
- *Android Studio mensaje: Missing Constraints in ConstraintLayout.* Stack Overflow.
- Launcher icon generator.
- Material design color palette.