UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Desarrollo de una aplicación para la gestión de ordenes de trabajo en una empresa de maquinaria agroalimentaria."

TRABAJO FINAL DE GRADO

Autor/a:
ALBERTO FERRER ROSELLO

Tutor/a:
JORDI BATALLER MASCARELL

GANDIA, 2020

Resumen

El propósito de este TFG es la implementación de un código orientado a la optimización del control y seguimiento de las ordenes de trabajo de una empresa. La aplicación debe ser capaz de descargar las ordenes de trabajo de cada usuario para que este mediante el sistema NFC del dispositivo móvil pueda interactuar con las máquinas y así poder ir cumplimentando la orden de trabajo “in situ”. Esta aplicación será capaz de funcionar con un mínimo de conexión a internet ya que contará con su propia base de datos local.

Palabras claves: NFC, Orden de trabajo, Algoritmo, Base de Datos Local, Aplicación móvil.

Abstract

The purpose of this TFG is the implementation of a code aimed at optimizing the control and monitoring of a company's work order. The application must be able to download the work orders of each user so that they can interact with the machines through the NFC system of the mobile device and thus be able to fill out the work order "in situ". This application will be able to work with a minimum of internet connection since it will have its own local database.

Key words: NFC, Work Order, Algorithm, Local Database, Mobile Application.

Contenido

1. Introducción

La empresa Citrosol.S.A. se dedica al tratamiento postcosecha de frutas y hortalizas, en el sector químico agrícola. Uno de los problemas que tiene la empresa es que los operarios del taller que se dedican a la reparación de las maquinas que suministra la empresa deben cumplimentar unas ordenes de trabajo de forma online. Para ello, actualmente utilizan la página web de la empresa que no permite poder cumplimentarlas in situ y de forma intuitiva. La mejora de esta problemática es la que justifica el desarrollo del presente proyecto.

Una orden de trabajo es una descripción de las tareas que los operarios deben realizar en las instalaciones de un cliente a una de las máquinas de tratamiento de fruta que la empresa Citrosol ha instalado. Por tanto, en la orden de trabajo figura la dirección de las instalaciones del cliente, el nombre y modelo de la máquina afectada, la descripción del problema de dicha máquina, fecha de apertura, y motivo de la orden (formación, revisión, mantenimiento o avería). El operario responsable de una orden de trabajo debe cerrarla cuando considera que ha finalizado su labor incorporando un informe y la fecha de finalización.

Las ordenes de trabajo se dan de alta los encargados del taller, para lo cual usan una página web de la empresa desde un ordenador de sobremesa, lo cual no supone ninguna incomodidad. Sin embargo, es en el uso por parte de los operarios de dicha página web lo que acarrea varios inconvenientes derivados de la necesidad de usar un ordenador para cumplimentar los formularios. Esto aplaza el cierre porque no puede realizarse en la instalación del cliente, lo cual supone incurrir en inexactitudes tanto en el tiempo como en la descripción certera del cierre.

Los responsables de la empresa han juzgado necesario que el operario pueda proceder al cierre de cada orden en el momento y lugar en que verdaderamente ha terminado la tarea.

Además del cierre de las ordenes de trabajo, la aplicación podrá rellenar los campos automáticamente de estas órdenes utilizando la tecnología NFC. Esto servirá tanto para trabajar sobre una orden de trabajo concreta como para poder crear una nueva orden de trabajo en el lugar donde haya una incidencia.

1.1 Objetivos.

Como acabamos de comentar, este proyecto tiene como motivación principal cubrir la necesidad real de una empresa para poder administrar y organizar las ordenes de trabajo. A continuación, enumeramos los principales objetivos de este proyecto:

- Estudiar las necesidades de la empresa y su metodología actual de trabajo, incluyendo la estructura y tramitación de las ordenes de trabajo.
- Revisión y aprendizaje de las tecnologías necesarias:
    - En los dispositivos móviles Android, estudiar el funcionamiento de la tecnología NFC, de Base de Datos Locales y GPS.
    - Estudiar las tecnologías que permiten el desarrollo de la parte de la aplicación correspondiente al servidor, como por ejemplo Mongodb, Node.js y Express.

- Diseñar la aplicación para la administración de ordenes de trabajo:
    - Elaborar una interfaz de usuario gráfica utilizando la herramienta Android Studio de Android.
    - Diseñar la arquitectura interna de la aplicación (métodos y algoritmos) tanto del cliente Android como de la aplicación servidor que vamos a utilizar.
- Seguir una serie de buenas prácticas que nos permitan implementar un código eficiente y robusto acorde con la arquitectura y los diseños previamente desarrollados.
- Planificar y realizar un conjunto de pruebas que acrediten la corrección de la aplicación desarrollada.
- Revisar todo el proceso para obtener conclusiones del trabajo realizado que permitan plantear mejoras.

1.2 Organización de la memoria.

La estructura de lo que resta de esta memoria es la siguiente.

- **Capítulo 2: Análisis de requerimientos.** Se desarrollan más extensamente los objetivos fijados y se detalla la planificación que se va a seguir con hitos concretos para lograr estos objetivos.
- **Capítulo 3: Diseño de la interfaz gráfica de usuario.** Durante este capítulo se muestra el entrono gráfico a crear a través de bocetos.
- **Capítulo 4 Diseño de la arquitectura de la aplicación.** Se realiza el diseño y la arquitectura interna de la aplicación, las funciones a crear y utilizar y los principales algoritmos. De esta forma se exponen todas las partes y procesos que forman parte del software de la aplicación.
- **Capítulo 5: Implementación.** Se realiza la parte práctica para justificar los principios teóricos y los algoritmos expuestos. Por otra parte, se concretan todas las herramientas utilizadas en el proyecto.
- **Capítulo 6: Manual de instalación.** En este capítulo veremos cómo poder instalar y configurar la aplicación para su correcto funcionamiento
- **Capítulo 7: Guía de usuario.** Veremos paso a paso el funcionamiento de la aplicación para que cualquier persona pueda utilizarla de forma correcta.
- **Capítulo 8: Conclusión.** Se exponen las conclusiones alcanzadas realizar el proyecto. También se hace un breve repaso de todos los puntos vistos en la memoria de forma objetiva y concisa.
- **Bibliografía.** Detalla las fuentes de información utilizadas durante la elaboración del proyecto.

2. Análisis de requerimientos de la aplicación.

Para la correcta elaboración de la aplicación se establecen una serie de requerimientos necesarios. El código debe ser lo más robusto posible. Debe procesar las diferentes formas de poder realizar las ordenes de trabajo y contemplar todas las opciones posibles. Además, debe de tener una interfaz gráfica sencilla e intuitiva para el usuario, de este modo se garantiza el correcto uso de la aplicación.

Para lograr este fin se plantean una serie de objetivos he hitos concretos a seguir:

1) **Estudiar los principios de funcionamiento de las ordenes de trabajo dentro de la empresa.** Hitos concretos:

    a. Conocer el funcionamiento general de las ordenes de trabajo que utiliza la empresa.
    b. Investigar sobre las tecnologías que utiliza la empresa para poder utilizarlas.
    c. Informarse de las necesidades tanto de los operarios como de la empresa para la nueva aplicación.
    d. Buscar mejoras para implementarlas en la aplicación.

2) **Estudiar el funcionamiento de los sistemas NFC.** Continuando con el trabajo de investigación, pero centrado en el sistema concreto a desarrollar. Hitos concretos:

    a. Investigación acerca de los principios básicos de los sistemas NFC.
    b. Listado del material y herramientas necesarias para desarrollar una aplicación con NFC.

3) **Estudiar las diferentes formas de conseguir una base de datos local dentro de la aplicación de Android.** Continuando con el trabajo de investigación, se busca la forma de conseguir tener una base de datos local en el propio dispositivo móvil para poder trabajar sin requerir una constante conexión con Internet.

    a. Búsqueda de herramientas de Android Studio que puedan servir para este fin.
    b. Una vez conocidas las herramientas de Android Studio posibles, probar las distintas posibilidades para elegir la mejor opción.

4) **Conocer de la estructura y características de las ordenes de trabajo.** Una vez tenemos claro el funcionamiento general de las ordenes de trabajo, el siguiente objetivo es analizar cómo se estructura una orden de trabajo y que características tiene. Hitos concretos:

    a. Búsqueda de los distintos parámetros que conforman una orden de trabajo.
    b. Evaluar cómo se almacenan estos parámetros en la base de datos de la empresa.
    c. Elaborar una lista de posibles cambios para adecuar los parámetros de las ordenes de trabajo a la aplicación de Android.

5) **Buscar y estudiar las necesidades específicas del empleado que utilizará la aplicación.** Esta tarea se basa en conocer las necesidades de los usuarios que utilizarán la aplicación. Hitos concretos:

    a. Preguntar a los empleados que necesidades tienen con respecto a la aplicación.
    b. Durante el proceso de la creación de la aplicación consultar con ellos las posibles mejoras para la aplicación.

6) **Desarrollar y exponer todos los procesos presentes para poder conectar nuestra aplicación con la base de datos de la empresa.** Esta tarea se basa en investigar y adecuar todos los procesos de la aplicación para que las ordenes de trabajo sean compatibles con la base de datos ya existente dentro de la empresa. Hitos concretos:

    a. Recoger en una lista todos los procesos descritos en las tareas anteriores para, manteniendo un orden, poder conectar la aplicación con la base de datos de la empresa.
    b. Determinar el tratamiento previo de las ordenes de trabajo para cada situación.

7) **Realizar un diseño (Boceto) de la interfaz de usuario de la aplicación.** El objetivo es diseñar un entorno gráfico con botones, imágenes y texto, de forma que los empleados puedan seguir cómodamente el tratamiento de las ordenes de trabajo en los diferentes casos que se planteen.

    a. Teniendo en cuenta las características que debe tener nuestra aplicación, dibujar como se distribuyen los diferentes elementos en nuestro entorno gráfico.

8) **Diseñar la arquitectura interna de la aplicación, los métodos y algoritmos que vamos a utilizar.** Con la lista de procesos elaborada y las interconexiones entre ellos realizada, se procede a hacer un diseño de la estructura interna de la aplicación. Debemos diferenciar la parte de la interfaz de usuario y de la aplicación, de la del tratamiento de estos datos desde el servidor o base de datos. Hitos concretos:

   a. Elaborar un esquema de la estructura interna de la aplicación con las relaciones entre procesos y métodos establecidos.
   b. Diseño de cada método utilizado. ¿Qué recibe y qué devuelve?
   c. Elaboración de los algoritmos internos de cada método, y funciones generales.

9) **Elaborar una interfaz de usuario gráfica con la herramienta de Android Studio.** Elaborar de forma gráfica, sin entrar en la programación interna, una interfaz de usuario basada en el diseño elaborado en la tarea 7. Hitos concretos:

   a. Repasar la documentación de las interfaces gráficas de Android Studio.
   b. Diseñar gráficamente la interfaz de usuario imitando el boceto de la tarea 7.

10) **Implementar un código eficiente y robusto para justificar los principios teóricos desarrollados.** Haciendo uso de los esquemas de la estructura interna y los algoritmos de los métodos generamos un código robusto. Hitos concretos:

   a. Generar un código para los métodos específicos de la parte del servidos o base de datos de la aplicación y comprobar su funcionamiento.
   b. Generar un código para las funciones de la parte de la aplicación. Estas funciones serán llamadas por la interfaz gráfica del usuario.
   c. Programaremos las funciones para interconectar la aplicación del dispositivo móvil con la base de datos de la empresa. Así como programas las funciones de los elementos de la interfaz gráfica.

11) **Realizar una fase de pruebas para comprobar los resultados.** En esta fase se probará el código implementado ya con la interfaz gráfica definitiva. Hitos concretos.

   a. Fase de pruebas del código con órdenes de trabajo creadas manualmente para comprobar el funcionamiento de la aplicación y poder resolver los problemas detectados.
   b. Fase de pruebas del código con órdenes de trabajo reales extraídas de la base de datos de la empresa.

12) **Comprobar los resultados para obtener conclusiones de los algoritmos diseñados y proponer posibles mejoras para corregir los problemas detectados.** Con el proyecto finalizado y el código funcionando correctamente se realiza un análisis de los resultados obtenidos en las pruebas y se extraen conclusiones a cerca de las posibles mejoras a implementar.

3. Diseño de la interfaz gráfica de usuario.

Para diseñar la arquitectura interna de la aplicación y determinar los métodos necesarios es imprescindible realizar un boceto inicial del entorno gráfico. Se pretende crear una interfaz de usuario. Esta interfaz de usuario contará con varios "*layouts*", es decir, tendrá diferentes capas en la interfaz.

Ilustración 1: Primer Layout Interfaz Gráfica

La primera capa sirve para poder autentificar al usuario. En esta capa tenemos los siguientes elementos.

- **Logo:** En la parte superior de la capa pondremos una imagen con el logo de la empresa.

- **Texto editable 1:** En este texto el operador deberá escribir su usuario para poder acceder a la aplicación.
- **Texto editable 2:** Este texto se utilizará para que el operador Introduzca su contraseña y así poder acceder a la aplicación.
- **Botón:** El botón se utilizará para iniciar la verificación y comprobar que el usuario y la contraseña son correctos para poder acceder a la aplicación.

Ilustración 2: Segundo Layout Interfaz Gráfica

La segunda capa de la interfaz gráfica será la principal de la aplicación, en ella tendremos una barra de herramientas para poder acceder a las diferentes opciones de la aplicación, así como un listado de las diferentes ordenes de trabajo del usuario.

- **Barra de herramientas:** Este elemento situado en la parte superior de la pantalla de la aplicación sirve para poner los distintos iconos que tendrán su propia funcionalidad dentro de la aplicación. En esta barra de herramientas tendremos cinco iconos diferentes:

  - **Icono 1:** Este icono será para realizar búsquedas entre las diferentes ordenes de trabajo.
  - **Icono 2:** El siguiente icono permitirá el acceso a las funciones NFC de la aplicación.
  - **Icono 3:** Este permitirá alternar entre las ordenes de trabajo que estén pendientes y aquellas que ja estén finalizadas.
  - **Icono 4:** Este icono será un menú desplegable que nos permitirá elegir entre crear una nueva Orden de Trabajo o Forzar una Orden de trabajo. La diferencia entre las dos la veremos más adelante.
  - **Icono 5:** Por último, este icono será el que nos permitirá cerrar la sesión del usuario.

- **Listado de Ordenes de trabajo:** Una lista ordenada de ordenes de trabajo que nosotros hemos diseñado: Estas órdenes de trabajo constan de:
  - **Icono 1:** un icono que representa un calendario para indicar que ese campo se muestra la fecha de creación de la orden de trabajo.
  - **Texto estático 1:** en este texto es donde muestra cuando se creó la orden de trabajo.
  - **Icono 2:** Este icono nos resaltará si la orden de trabajo está forzada o no está forzada.
  - **Texto estático 2:** En este campo se verá la ID de nuestra orden de trabajo.
  - **Texto estático 3:** Aquí se dará información del cliente que ha solicitado esta orden de trabajo.
  - **Texto estático 4:** Este nos dará información sobre la maquina a la que hace referencia la orden de trabajo.
  - **Texto estático 5:** En este caso el texto nos informa de que tipo de trabajo se debe realizar.
  - **Texto estático 6:** Por último, el campo de texto muestra en qué estado se encuentra la orden de trabajo.

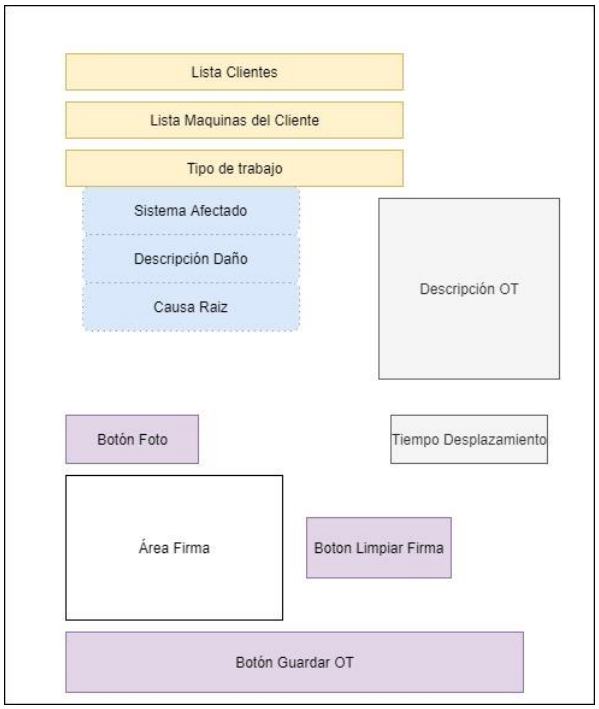

Ilustración 3: Layout Crear Orden de Trabajo.

La tercera capa de la interfaz gráfica será en la cual el usuario podrá crear las nuevas órdenes de trabajo. Para ello tendremos que diseñar un formulario que el usuario deberá rellenar para poder crear la orden de trabajo. Los elementos de esta capa de la aplicación son:

- **Lista desplegable 1:** En esta lista seleccionaremos el cliente el cual solicita la orden de trabajo.
- **Lista desplegable 2:** En ella seleccionaremos de este cliente la maquina a la cual se le va a realizar el trabajo.
- **Lista desplegable 3:** Esta lista sirve para seleccionar el tipo de trabajo que se le va a realizar a la máquina.
- **Lista desplegable 4:** Una lista auxiliar que sirve para en un tipo de trabajo concreto, ya que esta requiere más especificaciones permanece oculta si el tipo de trabajo no es el que corresponde.

- **Lista desplegable 5:** Al igual que la anterior también sirve como especificación de un tipo de trabajo concreto y también permanece oculta si no es el trabajo concreto.
- **Lista desplegable 6:** Igual que las dos listas anteriores, funciona de la misma forma.
- **Texto editable 1:** En este campo se rellenará una descripción breve del servicio que se va a realizar.
- **Texto editable 2:** En este campo se especifica el tiempo de desplazamiento en minutos que debe realizar el operario.
- **Botón 1:** Al pulsar el botón accederemos a la cámara para poder realizar una fotografía y poder señalar o registrar lo que el operario considere oportuno.
- **Área de dibujo:** En esta área el operario debe de firmar la orden de trabajo.
- **Botón 2:** Sirve para limpiar el área de dibujo de la firma y así poder volver a firmar.
- **Botón 3:** Botón para guardar la OT y poder empezar a realizar el trabajo.

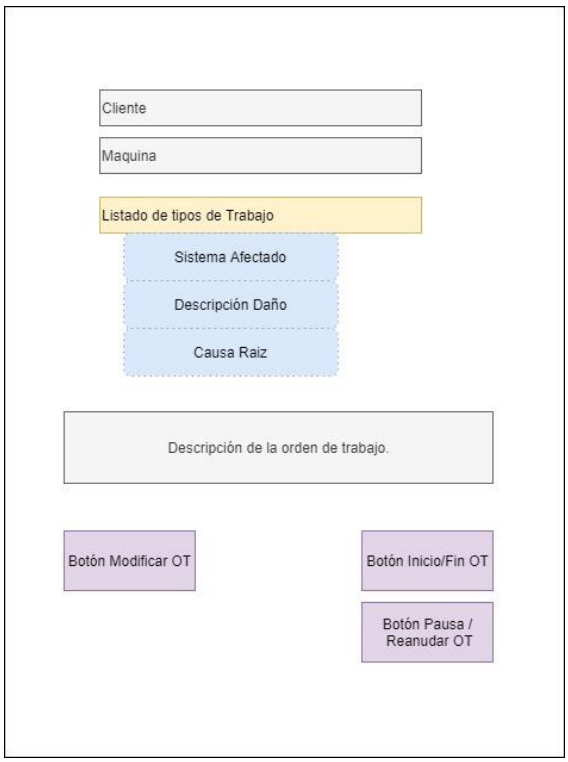

Ilustración 4: Layout Modificar Orden de Trabajo

La cuarta capa grafica de la interfaz de usuario de la aplicación corresponde a la funcionalidad de esta de poder modificar las ordenes de trabajo. Para poder modificarla la interfaz gráfica tiene los siguientes elementos:

- **Texto estático 1:** En este elemento se muestra a que cliente hace referencia la orden de trabajo.
- **Texto estático 2:** Este texto indica la maquina a la que hace referencia la orden de trabajo.

- **Lista de tipos de trabajo:** Esta lista se usa para poder modificar el tipo de trabajo que se va a realizar en la máquina.
- **Lista desplegable 1:** Una lista auxiliar que sirve para en un tipo de trabajo concreto, ya que esta requiere más especificaciones permanece oculta si el tipo de trabajo no es el que corresponde.
- **Lista desplegable 2:** Al igual que la anterior también sirve como especificación de un tipo de trabajo concreto y también permanece oculta si no es el trabajo concreto.
- **Lista desplegable 3:** Igual que las dos listas anteriores, funciona de la misma forma.
- **Texto editable 1:** En este texto podemos modificar la descripción de la orden de trabajo.
- **Botón 1:** Este botón guardará los cambios modificados de la orden de trabajo.
- **Botón 2:** Es el botón para iniciar una orden de trabajo que también servirá para finalizarla.
- **Botón 3:** Este botón servirá tanto para pausar la orden de trabajo como para reanudarla.

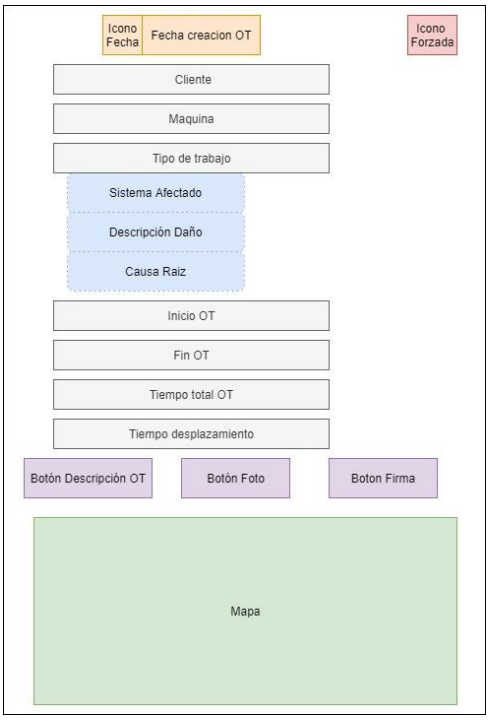

Ilustración 5: Layout Ordenes de Trabajo Finalizadas

La quinta capa de la interfaz gráfica de usuario la utilizaremos para ver un resumen de las ordenes de trabajo finalizadas. En esta capa tenemos:

- **Icono 1:** Un icono con el símbolo del calendario para indicar la fecha en la cual fue creada la orden de trabajo
- **Texto estático 1:** Se podrá observar la fecha en la cual fue creada la orden de trabajo.
- **Icono 2:** Este icono nos dirá si la orden de trabajo ha sido forzada o no.
- **Texto estático 2:** Se colocará el nombre del cliente de la orden de trabajo.

- **Texto estático 3:** En él se podrá observar la máquina en que el operario ha intervenido.
- **Texto estático 4:** Se mostrará el tipo de trabajo que ha realizado el operario.
- **Textos estáticos 5, 6 y 7:** Son los textos estáticos auxiliares para un tipo de trabajo en concreto. Estos permanecerán ocultos si no son requeridos.
- **Texto estático 8:** En él podremos comprobar la fecha y la hora a la cual se inició la orden de trabajo.
- **Texto estático 9:** Este texto sirve para ver la fecha y la hora en la cual la orden de trabajo finalizó.
- **Texto estático 10:** En este texto se observa cuanto a tardado en operario en realizar la orden de trabajo.
- **Texto estático 11:** En este último texto se observa el tiempo de desplazamiento empleado por el operario.
- **Botón 1:** Sirve para poder ver la descripción que hizo el usuario sobre la orden de trabajo.
- **Botón 2:** Este botón se utiliza para poder ver la foto que hizo el operario sobre la orden de trabajo.
- **Botón 3:** Es utilizado para poder ver la firma del operario en la orden de trabajo.
- **Mapa:** En este mapa podemos ver la ubicación en la cual el operario estuvo realizando y finalizando la orden de trabajo.

Finalmente, la última capa de la interfaz gráfica del usuario solamente contiene un fondo de pantalla en el cual se nos indica que el NFC del dispositivo está listo para funcionar.

4. Diseño de la arquitectura de la aplicación.

El diseño de esta aplicación para la gestión de las ordenes de trabajo se basa en lo siguiente:

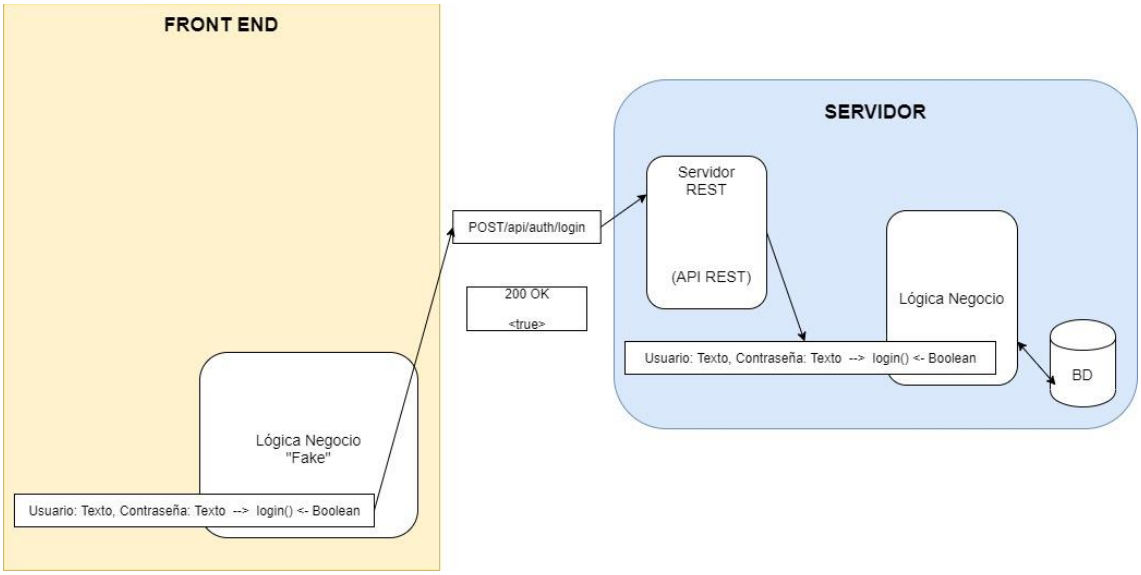

Ilustración 6: Arquitectura Aplicación

Como ya se adelantaba al principio del capítulo 1.3. "Metodología", el proyecto se divide en 2 grandes bloques: la parte del servidor o "back end" y la parte del usuario o "front end.

Como se observa en la imagen, en la parte del servidor tenemos:

- La base de datos donde almacenaremos y consultaremos todos los datos necesarios para el funcionamiento de nuestra aplicación.
- La lógica del negocio son aquellas funciones que nos permitirán interactuar con la base de datos y hacer todas las operaciones que requiera nuestra aplicación.
- Utilizaremos el servidor REST para conectar la parte de usuario o front end con nuestro servidor. Para ello también deberemos realizar una lista de reglas rest.

En la parte del front end vemos que también tenemos la lógica del negocio, pero "fake" esto se debe a que debemos implementar las mismas funciones en la parte del front end antes de que

la aplicación se conecte con el servidor, y con datos inventados por nosotros comprobar que la parte del front end funciona correctamente.

4.1. Estructura del esquema de la base de datos.

Para poder entender el funcionamiento de la aplicación debemos conocer como están estructurados los datos de los elementos de la base de datos que la aplicación va a consumir. Al estar realizando la aplicación para una empresa que ya dispone de una base de datos con estos elementos debemos intentar utilizarlos sin realizar demasiados casos. Los principales elementos de la base de datos que va a consumir la aplicación son: Usuarios, Clientes, Maquinas y Ordenes de trabajo.

4.1.1 Usuarios.

La estructura de datos de los usuarios es la siguiente:

- **Id:** Es un campo único que se utiliza para identificar de forma única el objeto dentro de una base de datos.
- **Roles:** Los usuarios tendrán unos roles dependiendo del trabajo que realicen dentro de la empresa, en este caso los usuarios que utilizan la aplicación deben tener el rol de taller.
- **Nombre.**
- **Apellidos.**
- **Correo:** Es el correo electrónico que tiene asignado dentro de la empresa.
- **Usuario:** Es un Nick que da la empresa a cada usuario y que le permite el acceso a las redes y aplicaciones de la empresa.
- **DNI:** El DNI del usuario.
- **Sesiones:** Este elemento contiene información sobre la fecha del último *login* del usuario, desde que dispositivo se conectó, desde que ip se conectó, en qué país se encuentra y el token. El Token no es más que una firma cifrada que permite a nuestra API identificar al usuario sin necesidad de enviar la contraseña y el usuario constantemente.
- **Hash:** Este campo contiene la contraseña del usuario codificada. En este caso esta codificada utilizando bcrypt.
- **Delegación:** Nos indica en que delegación de la empresa está trabajando el operario.

4.1.2. Clientes.

Ahora veremos la estructura en la base de datos de los clientes:

- **Id:** El campo único que identifica el elemento dentro de la base de datos.
- **Código:** Un código numérico que identifica a cada cliente.
- **Nombre:** El nombre de la empresa.
- **Lugar:** El lugar donde se ubica la empresa.

4.1.3. Maquinas.

Respecto a las maquinas en las cuales los operadores van a realizar su trabajo tenemos la siguiente estructura:

- **Id:** Campo único que identifica el elemento dentro de la base de datos.
- **Nombre:** Nombre descriptivo de la máquina.
- **Código_erp:** Identificador propio de la máquina.
- **m_preventivo:** Indicador booleano que nos indica si la maquina necesita de mantenimiento preventivo o no.
- **Cliente:** Código del cliente que nos informa a quien pertenece la máquina.
- **NFC:** Un booleano que nos indica si la maquina dispone de tag NFC.

4.1.4. Ordenes de Trabajo (OT).

Ahora veremos la estructura del elemento principal de nuestra aplicación que son las ordenes de trabajo:

- **Id:** Identificador único que diferencia el elemento dentro de una base de datos, pero en este caso controlaremos nosotros el identificador para saber en qué año se creó y que cantidad de ordenes de trabajo se han hecho en ese año.
- **Ubicación:** En este campo almacenaremos las coordenadas en las que el operario esté realizando la orden de trabajo.
- **Creada:** Este apartado nos indicará en qué fecha ha sido creada la orden de trabajo.
- **Tiempo_desplazamiento:** Aquí guardaremos el tiempo en minutos de lo que ha tardado el operario en llegar a la ubicación de la máquina que necesita revisarse.
- **Operario:** Aquí se guarda el Id del operario que realiza la orden de trabajo.
- **Cliente:** Guardaremos el código identificador del Cliente al cual le realizamos la orden de trabajo.
- **Máquina:** En este campo guardaremos el identificador de la máquina por la cual se ha realizado la orden de trabajo.
- **Firma:** Almacenaremos la información de la firma del operario. Esta información se almacenará como una cadena de caracteres en base64 que es un sistema de numeración posicional que usa 64 como base. Se utiliza para poder codificar imágenes.
- **TiempoTotal:** Se guarda el tiempo total que tarda el operario en realizar la orden de trabajo una vez iniciada.

- **Descripción:** Una breve descripción del operario respecto a la orden de trabajo.
- **Estado:** En este campo indicamos en qué estado se encuentra la orden de trabajo. Los estados son: asignada, en curso, en pausa y finalizada.
- **FotoPath:** Guardamos la ruta de la foto que realiza el usuario.
- **Forzada:** Booleano que nos indica si el operador ha tenido que forzar la orden de trabajo por algún motivo.
- **Tipo:** El tipo de trabajo que se va a realizar.
- **Sistema_Avería:** En caso de que se trate de una avería en la máquina, aquí se indicara que sistema es el que la causa.
- **Descripción_Avería:** Descripción de la avería que se ha producido en la máquina.
- **Causa_Avería:** Que ha causado la avería en la máquina.

Estos son los 4 elementos básicos de la base de datos que necesitaremos para poder implementar nuestra aplicación.

4.2 Lógica del negocio

En cuanto a la lógica del negocio de nuestra aplicación necesitaremos implementar las siguientes funciones:

- **Login ():** Esta función deberá recibir el nombre del Usuario y la Contraseña, ambos con formato de texto dentro de un JSON. La función deberá devolver un boolean dependiendo de si el usuario y contraseña son válidos o no. En el caso de que sean validos la función también devolverá un token en formato texto.
- **Clientes** (): A esta función no se le pasará ningún parámetro servirá para cargar todos los clientes que existan en la base de datos.
- **Maquinas ():** De forma muy similar a la función anterior esta sirve para cargar todas las maquinas que existan en la base de datos.
- **otsUsuario ():** En cuanto a esta función deberemos pasarle como parámetro la id del Usuario en formato texto dentro de un JSON. Esta nos deberá devolver todas aquellas ordenes de trabajo que tenga el Usuario especifico.
- **guardarOts ():** Para esta función deberemos pasarle un JSON con todos los parámetros que contienen las órdenes de trabajo. La función se encargará de almacenar la nueva orden de trabajo en la base de datos.

Con estas funciones podremos realizar todas las operaciones que requiere nuestra aplicación.

4.3. Lista de reglas REST

Utilizaremos una regla REST por cada una de las funciones que tenemos en nuestra lógica del negocio. En nuestro caso tendremos 5 reglas.

1. POST/auth/login/ se encargará de que el servidor realice la función login () con los datos que le pase el usuario y de devolverle el resultado.
2. GET/clientes llamará a la función clientes () y devolverá los resultados.
3. GET/maquinas esta funcionará igual que la anterior, pero devolviendo las maquinas.
4. POST/ot/ot-usuario se encargará de llamar a la función otsUsuario () y de pasar los resultados.
5. POST/ot/guardarot será la encargada de hacer funcionar el guardado de las ordenes de trabajo.

5. Implementación

Tras plantear la estructura de la aplicación y algunas de las funciones que compondrán el código final se procede a la programación de la aplicación con diversas herramientas que explicaremos a continuación.

5.1. Herramientas utilizadas.

Para la realización de nuestra aplicación hemos utilizado para la parte del servidor las herramientas **Mongodb, Express.js y Node.js.** Por otro lado, para la parte de la interfaz de usuario hemos utilizado la herramienta **Android Studio.** En cuanto a las tecnologías específicas hemos necesitado la utilización tanto del **GPS** como del **NFC.**

Primeramente, explicaremos que la herramienta fundamental para la base de datos es **MongoDB.** MongoDB es una base de datos NoSQL, es una base de datos orientada a documentos, lo que significa que almacena datos en forma de documentos tipo JSON. Esta forma de concebir los datos es más natural frente al tradicional modelo de filas y columnas. Esta es mucho más expresiva y potente. Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que **no es necesario seguir un esquema**. Los documentos de una misma colección pueden tener esquemas diferentes.

Utilizaremos **Node.js** para poder desarrollar la parte del servidor de nuestra aplicación.Node.js es un entorno de tiempo de ejecución de JavaScript. Este **entorno de tiempo** de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables como por ejemplo servidores web. Por lo tanto, será nuestra herramienta principal a la hora de crear nuestro servidor para la aplicación.

Utilizaremos **express.js** para interconectar nuestro servidor con nuestra aplicación.Express.js es el framework web de Node.js, y es la librería subyacente para un gran número de otros

frameworks web de Node.js. Proporciona mecanismos para abordar casi cualquier problema de desarrollo web.

Para desarrollar la aplicación propiamente dicha, así como la parte de la interfaz gráfica de usuario utilizaremos el entorno de desarrollo integrado oficial para la plataforma Android llamado **Android Studio.** Android Studio admite los lenguajes de Java, C++ y Kotlin. En nuestro caso hemos utilizado el lenguaje Java para programar la parte del "front end" de la aplicación.

El NFC es una comunicación de campo cercano en ingles Near Field Communication. Es una comunicación de muy corta distancia. El NFC es una tecnología de radio convencional evolucionada. Su función principal es establecer conexiones bilaterales con otro dispositivo similar que se encuentre dentro de su rango de acción corto. En nuestro caso utilizaremos, además del NFC del dispositivo móvil, los tags NFC. Estos tags o etiquetas NFC son diminutos circuitos integrados con una antena, generalmente integrados en pegatinas, activos o tarjetas inteligentes. En nuestro caso utilizaremos pegatinas, ya que las etiquetas NFC irán adheridas a las máquinas. En estos Tags NFC podemos escribir información de fácil lectura y ejecutable en los dispositivos NFC, como los teléfonos inteligentes.

5.2. Estructura de la aplicación.

En este apartado veremos la organización de nuestra aplicación en carpetas y el contenido de cada una de ellas. Analizaremos tres partes claramente diferenciadas: La base de datos, el servidor web, y la aplicación en Android Studio.

5.2.1 Base de datos.

Para la parte de la base de datos dentro de la carpeta **src** que se encuentra en la carpeta **back end** un fichero llamado *database.js* en él configuraremos todos los parámetros necesarios para el funcionamiento de nuestra base de datos.

Por otro lado, y también dentro de la carpeta src, encontraremos una carpeta llamada *models* en ella crearemos los archivos de los esquemas de nuestros componentes de la base de datos: usuarios, clientes, maquinas, ordenes de trabajo, etc.

5.2.2 Servidor web.

La mayor parte del código del servidor está dedicado a la realización del servidor web y a la lógica del negocio. Seguidamente explicaremos su estructura. Al igual que en la base de datos, encontraremos el código del servidor web dentro de la carpeta **src.**

Encontraremos un fichero llamado *app.js*. En el hemos definido las configuraciones necesarias para poder establecer una conexión. Por ello en este apartado hemos definido tanto el puerto por el cual conectaremos con el servidor, así como otros parámetros de configuración.

Seguidamente dentro del fichero llamado *index.js* realizaremos las conexiones propiamente dichas utilizando los parámetros definidos en *app.js***.**

Una vez realizada esta configuración encontraremos dentro de la carpeta **rutes** los ficheros de configuración de las reglas REST que necesitaremos para poder llamar a las funciones de nuestra lógica del negocio. Los ficheros son: *auth.js, cliente.js, maquina.js* y *ot.js*.

Una vez tenemos definidas todas las reglas REST debemos implementar nuestra lógica del negocio. Esta lógica la guardaremos dentro de una carpeta llamada **controllers.** Dentro de esta carpeta tendremos los ficheros: *authcontroller.js* que se encargará de realizar todas las funciones relacionadas con el login de los usuarios, *clientcontroller.js* en el realizaremos todas las funciones relacionadas con los clientes, *maquinacontroller.js* de igual manera implementaremos aquellas funciones relacionadas con el tratamiento de las maquinas, y por último, *otcontroller.js* en el que realizaremos todas aquellas funciones que necesitemos para el tratamiento de las ordenes de trabajo.

Finalmente tenemos una carpeta llamada **utils** en la cual hemos realizado algunas funciones auxiliares para que las funciones de la lógica del negocio pueda utilizarlas.

5.2.3 Front end

En este apartado analizaremos la estructura del código del front end. Primeramente, toda la aplicación se encuentra dentro de la carpeta **app.** Dentro de esta hay 3 carpetas que utilizaremos para realizar nuestra aplicación que son: **manifest, java y res.**

En la carpeta **manifest** solo hay un fichero llamado *AndroidManifest.xml* en él describiremos la información esencial de nuestra aplicación como permisos, paquetes y componentes de la aplicación.

En la carpeta **java** es donde ubicaremos todo el código java de nuestra aplicación Android. En nuestro caso nos encontramos con 3 carpetas además de 6 ficheros.

Los 6 ficheros corresponden con las 6 pantallas de la interfaz gráfica del usuario. De esta forma cada capa tiene su correspondiente código y sus funciones.

Las tres carpetas que encontramos dentro de la carpeta **java** son **models, CRUD y funciones.**

Dentro de la primera carpeta *models* tenemos los modelos que necesitaremos en nuestra base de datos local. Esta base de datos local nos permitirá poder trabajar sin conexión a internet.

La siguiente carpeta *CRUD* está relacionada también con la base de datos local. Las siglas CRUD son el acrónimo de Crear, Leer, Actualizar y Borrar en inglés. En esta carpeta ubicaremos aquellos ficheros destinados a manejar nuestra base de datos local.

Por último, tenemos la carpeta *funciones.* En ella ubicaremos todas aquellas funciones que necesita nuestra aplicación para su correcto funcionamiento.

La última carpeta que utilizamos en la implementación de la aplicación es la carpeta **res.** En ella configuraremos y programaremos toda la parte visual y estética de la aplicación.

Esta carpeta contiene 6 subcarpetas más que son: *drawable, layout, menú, mipmap, values, xml.*

- Drawable: En esta carpeta configuraremos aquellos iconos predefinidos de Android y que queramos utilizar en nuestra aplicación.

- Layout: En esta carpeta colocaremos aquellos ficheros que se utilizan para realizar el apartado gráfico de cada pantalla de nuestra aplicación.
- Menu: En este caso la carpeta solo tiene un fichero. Este fichero contiene el apartado gráfico de nuestra barra de herramientas que hemos situado en la parte superior de la página principal de nuestra aplicación.
- Mipmap: La carpeta sirve para almacenar el icono por el cual el usuario de Android podrá acceder a la aplicación.
- Values: En esta carpeta configuraremos aquellos colores o estilos que necesitemos predefinir para luego poder utilizarlos de una forma más cómoda en el resto de la aplicación.
- Xml: En esta última carpeta, en nuestro caso hemos creado una ruta donde se almacenarán las imágenes que hagamos con la cámara del dispositivo móvil.

5.3 Problemas resueltos.

En este apartado hablaremos sobre algunos de los distintos problemas que hemos tenido durante el desarrollo de la aplicación y como lo hemos solucionado.

5.3.1 Token

Uno de los primero problemas que nos encontramos durante la programación de la aplicación fue el tratamiento del token que nos devolvía el servidor en el caso de que la autentificación fueses correcta. El token debe servir para que el usuario pueda entrar y salir de la aplicación sin que tenga que volver a autentificarse. Solo debe volver a identificarse si el usuario cierra su sesión. Para ello debíamos almacenar el token y mantenerlo, aunque la aplicación se cerrase.

Para ello después de investigar y probar algunas soluciones se llegó a la conclusión que la mejor forma de tratar el token era a través de ***Shared Preferences,*** estas nos permiten guardar nuestro el dato con una clave, en nuestro caso TOKEN, y que se guarde a pesar de que nuestra aplicación se cierre. Además, este valor podrá ser consultado en cualquier capa de nuestra aplicación.

5.3.2 Gestión de la ubicación.

Otro problema que nos encontramos durante la realización de la aplicación fue la obtención de la ubicación en tiempo real. En un primer momento conseguimos obtener la ubicación, pero para ello la geolocalización del dispositivo tenía que estar activa en todo momento mientras se utilizaba la aplicación. Esto significaba que la aplicación consumía muchos recursos del dispositivo móvil constantemente y aumentaba el consumo de la batería.

Después, una vez se pudo obtener la ubicación sin que la geolocalización estuviese activa en todo momento, tuvimos el problema que algunas veces al reiniciar el dispositivo móvil o a veces de forma aleatoria, no se encontrase ninguna ubicación y por lo tanto la aplicación dejase de funcionar.

Finalmente afinando el código se llegó a que la aplicación solo utiliza la geolocalización un segundo durante el cambio de pantallas de la aplicación. De esta manera podemos obtener siempre la ubicación exacta y además evita que utilicemos demasiado los recursos del dispositivo móvil.

Otro problema derivado de obtener la ubicación fue limitar que el usuario pudiese utilizar la aplicación si este no activa la geolocalización del móvil.

Una vez solucionado este inconveniente, si la aplicación detecta que el GPS del móvil esta desconectado requerirá al usuario que acepte conectarlo, y en el caso de que no acepte la aplicación no permitirá que el usuario siga utilizándola.

5.3.3 El campo para firmar.

Uno de los problemas fue también la realización de un espacio donde el Usuario pudiese introducir una firma manual.

Primero se intentó utilizar un recurso de imagen de Android Studio que nos permitía dibujar, pero no permitía ni almacenar la firma ni poder borra el campo de la firma.

Después intentamos pasar la firma desde formato imagen a texto. Para ello se utilizó una conversión donde obteníamos un texto en base 64. Pero a pesar de ello continuábamos sin poder limpiar el campo de la firma y además no podíamos reconstruir la firma.

Finalmente encontramos una solución utilizando un LinearLayout que es un contenedor que utiliza Android Studio para poder introducir componentes visuales como botones o textos. Nosotros transformamos ese LinearLayout en un espacio donde se puede dibujar y además pudimos tanto capturar la firma como poder borrar el contenedor para poder volver a firmar.

5.3.4 Conexión entre la parte del usuario y el servidor.

El último problema que analizaremos lo tuvimos probando la conexión entre la interfaz de usuario y el servidor mediante las reglas REST. Al llamar a las funciones de la lógica del negocio mediante estas reglas comprobamos que no había comunicación entre el dispositivo móvil y el servidor. Pero si utilizábamos una herramienta que nos permitía realizar estas peticiones REST como *Postman* nos dimos cuenta de que a través de esta herramienta se establecía la conexión con el servidor y además nos devolvía los datos de forma correcta, por lo que en un principio se pensó que el problema se encontraba en el código de la parte del frontend. Después de cambiar el código y de intentar solucionarlo de múltiples formas, resultó que el problema no era del código de nuestra aplicación. El problema por el cual no funcionaba nuestra aplicación era porque el firewall de Windows bloqueaba nuestras conexiones con el servidor. Una vez añadida la excepción en el firewall de Windows la aplicación funcionó correctamente.

6. Manual de instalación.

En este apartado explicaremos todos aquellos pasos que debemos seguir para poder instalar y hacer funcionar nuestra aplicación.

Primero deberemos tener instalado en nuestro ordenador el editor de código **Visual Studio**, la herramienta **Android Studio** y **Nodejs**. Para ello solo necesitamos ir a las páginas oficiales descargarlos e instalarlos en nuestro ordenador.

Seguidamente deberemos también instalar Mongodb en nuestro PC. Para ello deberemos entrar en la página https://www.mongodb.com/try/download/community. Una vez dentro deberemos seleccionar la opción **On-Premises**.

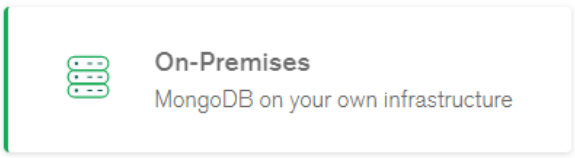

Ilustración 7: Botón On-Premises

En el desplegable de opciones que nos aparecen deberemos seleccionar **MongoDB Community Server** y luego pulsaremos el botón **download**.

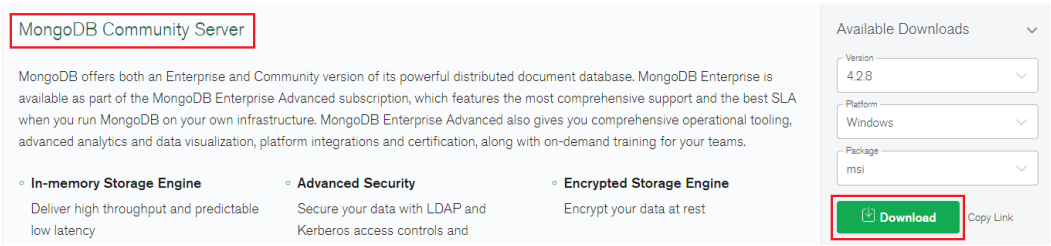

Ilustración 8: Seleccionar MongoDB Community Server y descargar

Una vez descargado el archivo lo ejecutaremos y procederemos a la instalación asistida. Debemos seleccionar la instalación completa y la opción de instalarlo como un servicio, además es importante conocer la ruta de la instalación.

Una vez instalado abriremos la consola y accederemos a la carpeta de instalación **C:\Program Files\MongoDB\Server\4.2\bin** o en tu caso la versión que hayas instalado. Dentro escribiremos **mongod** y lo ejecutaremos. Una vez hecho esto el servidor mongodb estará corriendo.

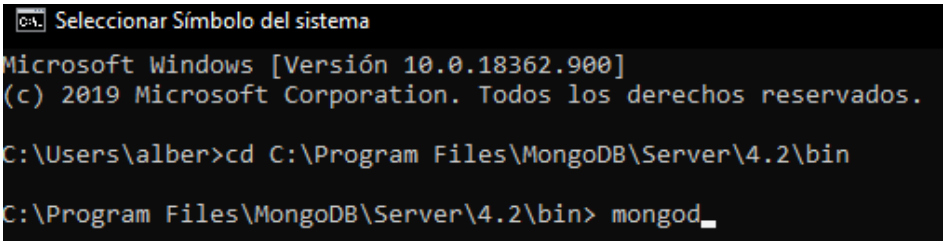

Ilustración 9: Inicializar Mongodb

Una vez instalados todas las herramientas que necesitamos para hacer funcionar nuestra aplicación entraremos en https://drive.google.com/file/d/13JbzuRzAAvDF2ZkgzoVJCb_0rref-6pM/view?usp=sharing y descargaremos el archivo comprimido de la aplicación. Una vez descargado deberemos descomprimirlo.

Después deberemos abrir el **Visual Studio Code**. Luego seleccionaremos la pestaña **File** y la opción **Open Folder** y seleccionaremos la carpeta **backend** del archivo de la aplicación.

Una vez abierta la carpeta **backend** en **Visual Studio** deberemos pulsar el comando **Ctr+ñ**. Este comando nos abrirá una consola de comandos. En ella deberemos escribir **npm run dev**, este comando nos permitirá ejecutar el servidor de nuestra aplicación y al mismo tiempo si hacemos algún cambio en el servidor y guardamos volverá a compilar todo el código del servidor de forma automática.

Si todo ha ido bien deberíamos ver lo siguiente:

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS D:\utorrent\AplicacioDB\AplicacionAndroid\CitroNFC\backend> npm run dev

> backend@1.0.0 dev D:\utorrent\AplicacioDB\AplicacionAndroid\CitroNFC\backend
> nodemon src/index.js

[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
server on port 8600
DB is connected
```

Una vez instalada la parte del servidor de nuestra aplicación pasaremos a instalar la parte del usuario. Para ello debemos ejecutar **Android Studio**. Seguidamente en la panta principal seleccionaremos la opción "Open an existing Android Studio Project" y seleccionaremos la carpeta UX que se encuentra dentro del archivo de nuestra aplicación.

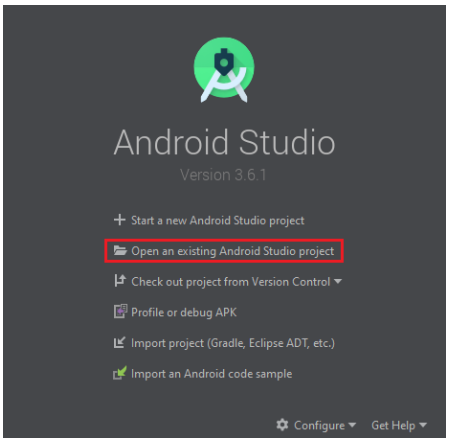

Ilustración 11: Selección el proyecto existente

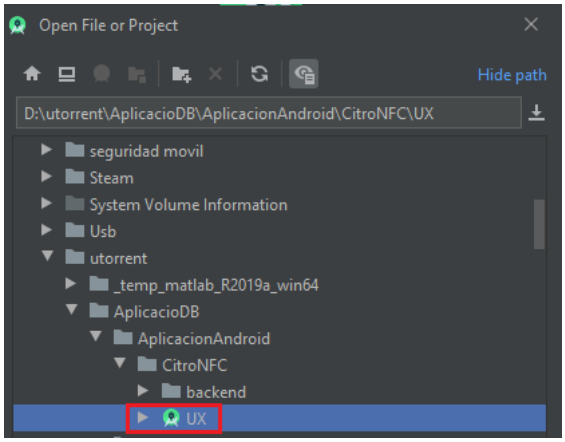

Ilustración 10: Seleccionar UX de la aplicación

Ahora deberemos saber cuál es la ip de nuestro servidor, es decir la ip del ordenador donde se está ejecutando el servidor. Una vez que conozcamos nuestra ip deberemos buscar en **Android Studio** el archivo **MainActivity** este se encuentra dentro de la carpeta **app** a su vez dentro de la carpeta **java.** Deberemos abrir este archivo y en la parte superior deberemos cambiar la ip de *public final static Logica laLogica = new Logica("http://**192.168.1.3**:8600/api");* por nuestra ip.

```
public class MainActivity extends AppCompatActivity {
    SharedPreferences sharedpreferences;
    Button botonLogin;


    JSONObject clients = new JSONObject();

    //private final static Logica laLogica = new Logica("http://172.15.2.4:6512/api"); //base de datos "Real"

    public final static Logica laLogica = new Logica( urlDestino: "http://192.168.1.3:8600/api");//base de Datos Casa
```

Ilustración 12: Modificar la ip de nuestro servidor

Seguidamente deberemos configurar nuestro dispositivo móvil para que acepte la transferencia de archivos por vía USB y luego lo conectaremos a nuestro ordenador.

Finalmente, una vez que Android Studio reconozca nuestro dispositivo móvil deberemos ir a la pestaña **Run** y seleccionar la opción **Run app** o pulsaremos la flecha verde.

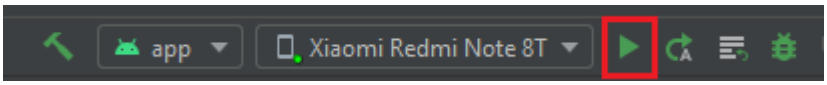

Ilustración 13: Botón para instalar y ejecutar la aplicación.

Con esto la aplicación se instalará en nuestro dispositivo móvil y ya será plenamente funcional. Si por alguna razón la aplicación no se comunica con nuestro servidor deberemos revisar el firewall de Windows y en caso de que sea necesario añadir una excepción al puerto 8600.

7. Guía de uso.

En este apartado explicaremos con detalle cómo se utiliza nuestra aplicación.

Antes de explicar el funcionamiento de nuestra aplicación conviene recordar que para que la aplicación funcione correctamente el dispositivo móvil debe de tener encendido la localización GPS. En caso de no ser así, nos aparecerá un dialogo instándonos a activarla.

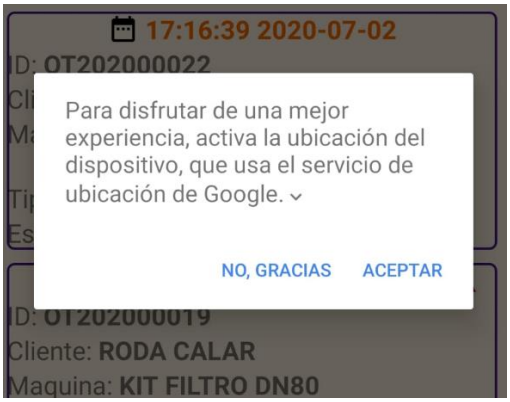

Ilustración 14: Dialogo localización GPS

Si aceptamos, automáticamente se activará la localización GPS de nuestro dispositivo móvil, en caso de que no se acepte, la aplicación no dejará al usuario continuar usándola.

La primera pantalla de nuestra aplicación sirve para identificarse con un usuario y contraseña proporcionados por la empresa. Lo único que debemos hacer es introducir el usuario y la contraseña y pulsar entrar.

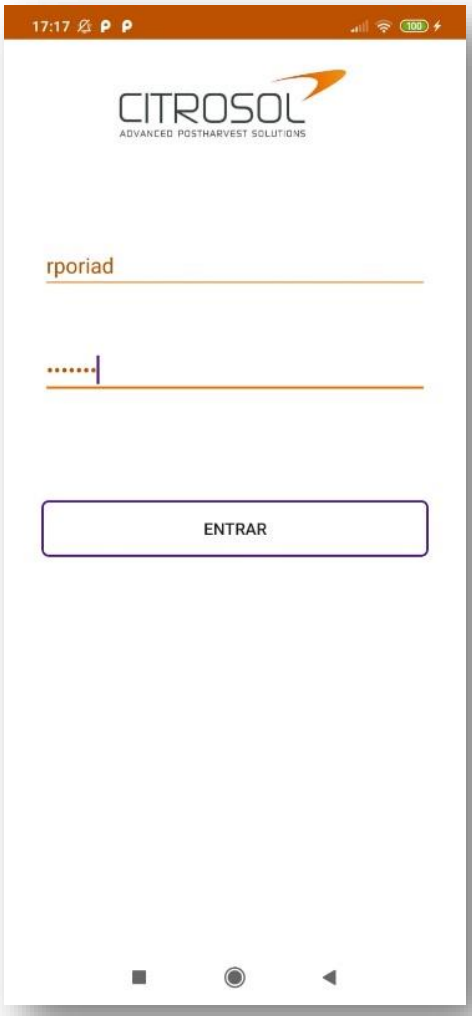

Ilustración 15: Pantalla de autentificación de la aplicación

Cuando demos al botón de entrar pueden suceder varias cosas si no pasamos a la siguiente pantalla de la aplicación. Puede ser que la aplicación nos informe de que el usuario que hemos introducido no existe o bien que la contraseña es incorrecta:

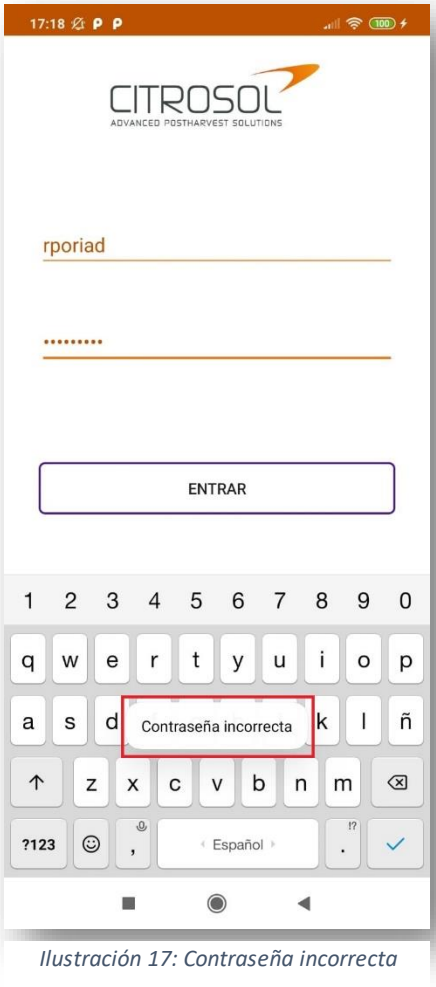

Ilustración 17: Contraseña incorrecta

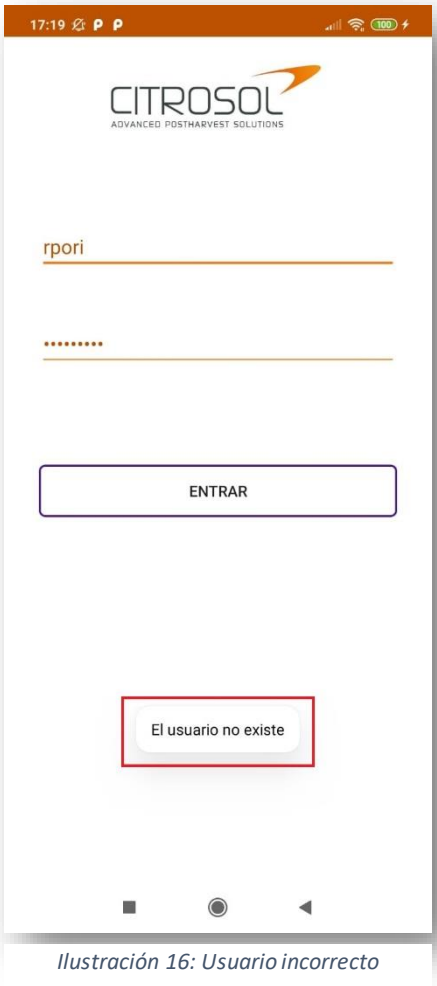

Ilustración 16: Usuario incorrecto

El último caso que puede ocurrir en caso de que no pasemos a la siguiente pantalla, es que la aplicación no puede conectar con el servidor por algún motivo. En ese caso deberíamos revisar si todo está funcionando correctamente.

Si todo funciona correctamente pasaremos a la siguiente pantalla donde se nos mostraran todas las ordenes de trabajo que están asignadas al usuario ordenadas de más recientes a más antiguas hasta un máximo de antigüedad de un mes. Todas aquellas ordenes de trabajo que tengan más de un mes de antigüedad no se enviarán a la aplicación.

Un ejemplo de cómo se vería es el siguiente:

Ilustración 18: Pantalla principal aplicación.

Como podemos observar de cada orden de trabajo podemos conocer su ID, el cliente, que maquina es que tipo de trabajo es el que el usuario ha de realizar y en qué estado se encuentra dicho trabajo.

Para poder modificar una orden de trabajo lo único que deberemos hacer es pulsar sobre la que nos interese y esto nos llevará a otra pantalla donde podremos modificarla.

En el ejemplo siguiente podemos ver como modificarla:

Ilustración 19: Modificar orden de trabajo.

Se puede observar que tanto el cliente como la maquina están estáticos y no se pueden modificar. Lo que podemos modificar son qué tipo de trabajo debemos realizar, así como la descripción de la orden de trabajo.

Para modificar el tipo de trabajo lo único que debemos hacer es pulsar sobre el trabajo a realizar que hay actualmente y se nos abrirá un desplegable con las distintas opciones que dispone la empresa y seleccionaremos el adecuado.

Ilustración 20: Lista de trabajos.

Para cambiar la descripción basta con pulsar encima y nos aparecerá el teclado para que el usuario pueda escribir.

Una vez realizados los cambios el usuario debe de pulsar el botón modificar y así se quedarán guardados los cambios.

Esta pantalla de modificar las ordenes de trabajo también es donde empezaremos, pausaremos, reanudaremos y finalizaremos las ordenes de trabajo. Para ello simplemente bastará con utilizar los botones que aparece debajo a la derecha de la pantalla. Una vez pulsado el estado en que se encuentra la orden de trabajo cambiará y automáticamente volveremos a la pantalla principal donde se encuentra el listado de las ordenes de trabajo.

Existen casos en que si pulsamos una orden de trabajo esta no se abre debido a que la aplicación está pensada para que el usuario tenga que utilizar el lector NFC del dispositivo móvil. Esto garantizará a la empresa que el empelado está junto a la maquina a la hora de realizar y cumplimentar las ordenes de trabajo. Por tanto, si la maquina con la que el operario va a trabajar cuenta con un tag NFC y el operario intenta acceder a la orden de trabajo sin utilizar el NFC ocurrirá lo siguiente al pulsar sobre ella:

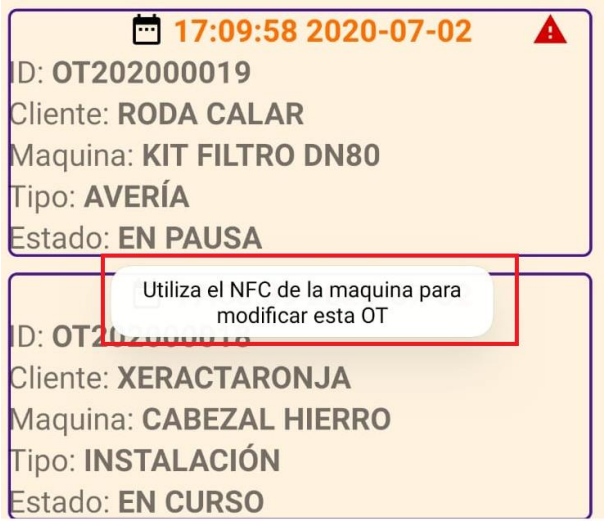

Ilustración 21: Maquina con NFC

Por tanto, para poder modificar una orden de trabajo mediante el NFC deberemos seguir los siguientes pasos:

Primeramente, deberemos activar el NFC de nuestro dispositivo móvil. En caso de que el dispositivo no dispongo de lector NFC o este apagado la aplicación nos avisará con el mensaje siguiente:

Ilustración 22: NFC no disponible

Una vez tengamos el NFC activo en nuestro dispositivo móvil deberemos pulsar el Icono NFC que aparece en la barra de herramientas de la pantalla.

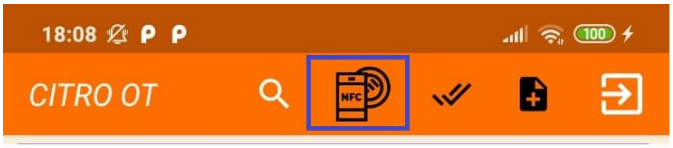

Ilustración 23: Boton NFC, barra herramientas aplicación.

Una vez pulsado el botón si el NFC no está disponible nos aparecerá el mensaje anterior, pero si todo va bien la aplicación pasará a una pantalla donde estará a la espera de recibir la información de la maquina a través del lector NFC.

Ilustración 24: Pantalla lector NFC

Una vez que la aplicación detecte la información de la maquina esta abrirá automáticamente la orden de trabajo asignada a la máquina para que el usuario pueda modificarla. Por tanto, para poder pausar, reanudar o finalizar la orden de trabajo el operario deberá estar cerca de la maquina en cuestión.

Ahora que ya sabemos cómo modificar las ordenes de trabajo pasaremos a explicar cómo crear nuevas.

Existen tres formas diferentes de crear una nueva orden de trabajo.

La primera que explicaremos es la forma ideal de crear una nueva orden de trabajo, y la aplicación está pensada para utilizarla. Para ello solo debemos volver a pulsar el botón del NFC y acercar el dispositivo móvil al tag NFC. Una vez detectada la máquina y si esta no tiene ninguna orden de trabajo activa actualmente, se abrirá automáticamente el formulario para crear una nueva orden de trabajo, con los campos del cliente y la maquina ya seleccionados.

Ilustración 25: Nueva orden de trabajo NFC

Como se observa en la ilustración tanto el cliente como la maquina ya están seleccionados. Seguidamente el operario deberá seleccionar el tipo de trabajo que se realizará, así como una descripción si lo cree conveniente, el tiempo de desplazamiento hasta llegar al lugar de la incidencia y luego deberá firmar en el recuadro habilitado para ello en blanco. Además, si lo considera oportuno mediante el botón de la izquierda podrá realizar una fotografía del estado de la maquina si lo considera oportuno. Una vez rellenados los campos pulsaremos el botón GUARDAR OT y ya tendremos la nueva orden de trabajo lista. Para poder modificarla el operario deberá utilizar el lector NFC.

La siguiente forma de crear una nueva orden de trabajo la utilizará el operario cuando la maquina a la que se le va a realizar un trabajo no disponga de NFC.

Para ello lo primero que debemos hacer es ir a la barra de herramientas y pulsar el siguiente botón que nos desplegara un menú, en él, elegiremos *Nueva OT.*

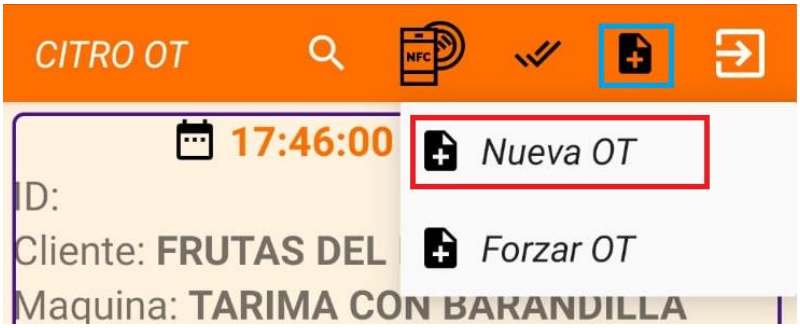

Ilustración 26: Nueva OT sin NFC

Una vez que pulsemos encima de *Nueva OT* se abrirá una pantalla muy similar a la de la ilustración 24 pero donde tendremos que elegir primero el cliente.

Ilustración 27: Selección de clientes Nueva OT sin NFC

Una vez seleccionado el cliente nos aparecerá un desplegable con todas las máquinas del cliente que no tienen el tag NFC.

Todas aquellas máquinas del cliente que dispongan del tag NFC no aparecerán en la lista y no podrán ser seleccionadas. Una vez seleccionada la maquina el resto se deber rellenar de igual forma que en el caso anterior con NFC.

Ilustración 28: Máquinas del cliente sin tag NFC

Por último, la tercera forma por la cual la aplicación nos permite crear una nueva orden de trabajo esta pensada para cuando todo falla, es decir, cuando la máquina que se supone que tiene tag NFC no lo tiene o se este se ha roto o ha habido cualquier problema que impide que el operario pueda utilizar el lector NFC del móvil.

Para poder realizar esta nueva orden de trabajo debemos ir nuevamente al icono que nos despliega un menú, pero esta vez seleccionaremos *Forzar OT.*

Ilustración 29: Forzar OT

Una vez pulsado *Forzar OT* se nos abrirá un dialogo alertándonos de si estamos seguros de querer realizar esta acción e informándonos de que la empresa luego podrá pedir explicaciones.

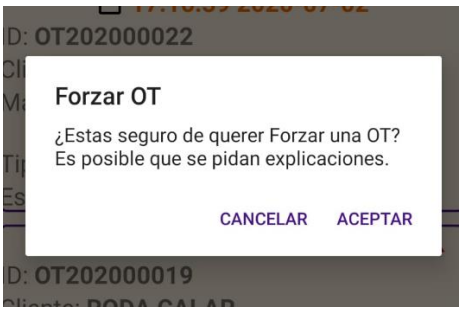

Ilustración 30: Dialogo de alerta de Forzar OT

Una vez aceptemos nos aparecerá la misma pantalla que cuando hemos creado una orden de trabajo nueva. La única diferencia es que ahora cuando seleccionemos el cliente nos aparecerán todas las máquinas que tiene el cliente, tanto si tienen o no tienen tag NFC.

Ilustración 31: Todas las maquinas del cliente

A partir de aquí todo el procedimiento es exactamente igual que en los casos anteriores.

Si en algún momento creamos una OT que este "forzada" en la lista de ordenes de trabajo de la pantalla principal, esta nos aparecerá con un icono rojo en la parte superior derecha de la orden de trabajo indicándonos que ha sido "forzada".

Ilustración 32: Orden de trabajo "forzada"

Para finalizar explicaremos las otras utilidades de la barra de herramientas de nuestra aplicación.

El primer icono que tiene forma de lupa sirve para buscar una o varias órdenes de trabajo específicas. Ya sea por el nombre del cliente, de la máquina, su estado, o el tipo de trabajo a realizar.

Ilustración 33: Icono Buscar

Cuando pulsemos el icono se nos abrirá un campo de texto donde escribiremos lo que deseamos buscar y automáticamente se irán descartando todos aquellas ordenes de trabajo que no estén incluidas en la búsqueda.

Ilustración 34: Búsqueda de un cliente

El siguiente icono que explicaremos es el del doble "check". Este botón sirve para alternar entre las ordenes de trabajo que estén ya finalizadas y aquellas que aun estén en curso.

Ilustración 37: Icono doble "check"

Ilustración 36: OTs en curso

Ilustración 35: OTs finalizadas

Cuando las ordenes de trabajo estén finalizadas, si pulsamos sobre ellas podremos acceder a un resumen de la propia orden de trabajo.

Ilustración 38: Resumen orden de trabajo finalizada

En este resumen podemos ver el cliente, la maquina y el tipo de trabajo que se ha realizado.

Además, también podemos consultar en qué fecha se inició la orden de trabajo y en qué fecha se finalizó

Por otro lado, también se podrá consultar el tiempo empleado por el operario en finalizar el trabajo. Este puede variar de la simple fecha de inicio y fin de la orden de trabajo debido a que existe la opción de poder pausar la orden de trabajo. También se puede consultar el tiempo de desplazamiento en minutos que empleó el operario.

Después, existen tres botones que al pulsarlos nos aparecerá una ventana emergente que nos mostrará tanto la descripción, la foto y la firma que haya realizado el operario durante la creación de la orden de trabajo.

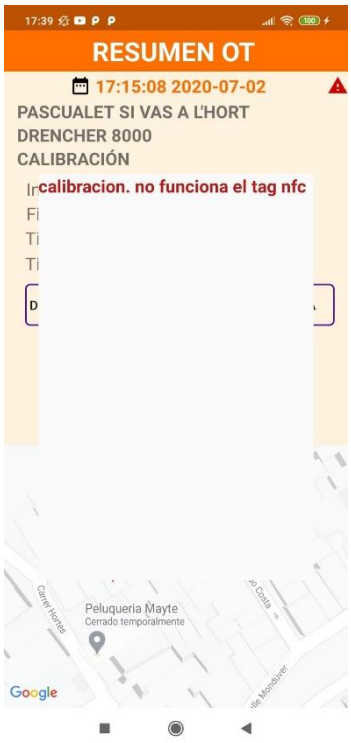

Ilustración 41: Descripción resumen OT

Ilustración 40: Foto resumen OT

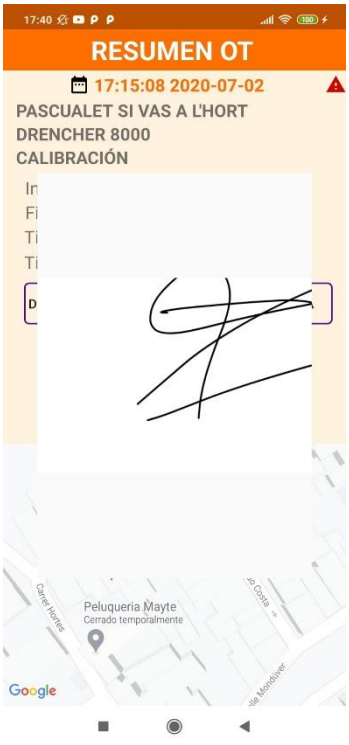

Ilustración 39: Firma resumen OT

Finalmente se puede observar la ubicación donde el operario ha realizado el trabajo mediante un pequeño mapa de Google, como se puede observar en la ilustración 37.

Y ya para finalizar esta guía de usuario pasaremos a explicar el último de los iconos que tenemos en nuestra barra de herramientas.

Ilustración 42: Botón cerrar sesión.

Este botón nos permitirá cerrar la sesión del usuario. Una vez que cerremos la sesión del usuario se mandarán todos los datos de la base de dato local del dispositivo móvil al servidor de la aplicación actualizándose toda la información de este. Además, una vez que cerremos la sesión para poder volver a acceder a la aplicación deberemos de volver a identificarnos.

Cuando pulsemos el botón de cerrar sesión nos aparecerá un dialogo el cual nos pedirá una confirmación.

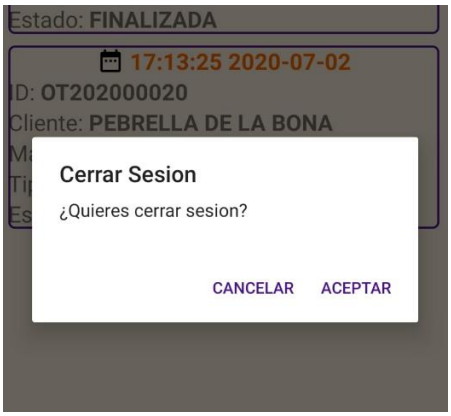

Ilustración 43: Dialogo cerra sesión.

8. Conclusiones

La gestión de las ordenes de trabajo de la empresa Citrosol presentaba una serie de inconvenientes principalmente centrados en el cierre de dichas ordenes por parte de los operarios. El objetivo principal de este trabajo final de grado ha sido el diseño y desarrollo de una aplicación cliente-servidor que permita a los operarios formalizar el cierre de una orden a través de un dispositivo móvil en el lugar donde han realizado su tarea. Adicionalmente, la empresa obtiene los datos de los trabajos realizados de forma más eficaz y fiable.

Respecto al resultado de esta aplicación podemos darnos por muy satisfechos porque es totalmente funcional y la empresa planea en un futuro incorporarla al uso de sus trabajos.

En lo concerniente a la formación personal hemos de juzgar como muy positivo el desarrollo de este trabajo final de grado porque gracias a él hemos aprendido cómo desarrollar aplicación cliente-servidor de forma integral. Hemos utilizado el sistema Android para el front end, y específicamente la tecnología NFC y GPS dentro de él. Por su parte para el servidor o back end hemos adquirido conocimientos en Node.js y Mongodb.

Con respecto a mejoras de la aplicación de cara al futuro podríamos sugerir las siguientes:

- Conexiones intermitentes a la red que permitieran cada cierto tiempo que se actualizarán los datos. Ahora mismo solos se actualizan cuando el usuario cierra la sesión de la aplicación.
- Además, en un futuro se podrían implementar nuevas características a la aplicación tales como escanear las huellas dactilares para identificarse en la aplicación, ya que esto haría más fácil a los operadores poder utilizar la aplicación.
- También podríamos mejorar el apartado gráfico de la aplicación y hacerla visualmente más atractiva.

9. Bibliografía

Canal Fazt Code. (s.f.). Obtenido de https://www.youtube.com/channel/UCMn28O1sQGochG94HdlthbA

Developers Android Studio. (s.f.). Obtenido de https://developer.android.com/studio/intro?hl=es-419

Hebuterne, S. (2018). *Desarrolle una aplicación android. Programación en Java con android studio.* Eni.

MongoDB Tutorials. (s.f.). Obtenido de https://docs.mongodb.com/manual/tutorial/

Schildt, H. (2018). *Java 9.* Grupo Anaya Publicaciones Generales.

stackoverflow en español. (s.f.). Obtenido de https://es.stackoverflow.com/

W3chools. (s.f.). Obtenido de https://www.w3schools.com/java/default.asp