UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

Trabajo final del GºIng.Informática:

App para el diagnóstico de fallos en fresadoras

Presentado por Juan Francisco Benito Cuesta
en junio de 2018
Tutor Raúl Marticorena Sánchez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

D. Raúl Marticorena Sánchez, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Juan Francisco Benito Cuesta, con DNI 45575555C, ha realizado el Trabajo final del GºIng.Informática titulado: App para el diagnóstico de fallos en fresadoras.

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual, Se autoriza su presentación y defensa.

En Burgos a de junio de 2018

Raúl Marticorena Sánchez

Índice de contenido

Índice de ilustraciones

Índice de tablas

Resumen

Aplicación en Java para Android que diagnostica los fallos de fresadoras a partir de las alarmas que producen estas. Realiza una serie de preguntas, dependiendo de la alarma que se haya activado, y el usuario debe ir eligiendo las respuestas según el estado de la fresadora.

Una vez finalizadas las preguntas, el problema se habrá resuelto o la aplicación te dirá una posible solución.

Estas alarmas con sus respectivas preguntas y respuestas están definidas en ficheros JSON. La aplicación obtiene de un API REST la alarma o alarmas que se han activado y parsea los ficheros JSON correspondientes para obtener las preguntas y respuestas de dichas alarmas.

Abstract

Application in Java for Android that diagnoses the failures of milling machines from the alarms that produce these. It asks a series of questions, depending on the alarm that has been activated, and the user must choose the answers according to the state of the milling machine.

Once the questions are finished, the problem will have been solved or the application will tell you a possible solution.

These alarms with their respective questions and answers are defined in JSON files. The application obtains from an REST API the alarm or alarms that have been activated and parses the corresponding JSON files to obtain the questions and answers of those alarms.

I - *INTRODUCCIÓN*

Este trabajo está pensado para los empleados de una empresa que trabajan con fresadoras. La aplicación que se realiza en este proyecto sirve para ayudar o incluso resolver los problemas y fallos que producen dichas fresadoras. Se trataría de un asistente que te ayuda realizando preguntas y que cualquier empleado puede tener en su teléfono móvil.

El trabajo consta de un proyecto de Android Studio, varios ficheros JSON y una serie de imágenes.

- El proyecto es la aplicación en sí, que consta de la funcionalidad: los activity[Activity] de cada pantalla y el modelo de clases de las alarmas, y de la interfaz gráfica: los layouts. [Layouts]
- Los ficheros JSON representan las diferentes alarmas que pueden saltar en las fresadoras. Cada fichero es una alarma y consta de su número, título, descripción, preguntas, respuestas y sus imágenes asociadas.
- Las imágenes son ayudas visuales que se aportan para resolver la situación. Cada alarma puede tener una o varias imágenes.

Esta memoria se divide en unos objetivos que se persiguen con la realización del proyecto, unos conceptos teóricos para explicar algunos detalles del trabajo, las técnicas y herramientas utilizadas en la elaboración del proyecto, aspectos relevantes a mencionar, trabajos que pueden tener relación con este, una serie de conclusiones y la bibliografía.[Bibliografía]

II - OBJETIVOS DEL PROYECTO

Los objetivos principales del proyecto son:

- Desarrollar una aplicación para que los empleados de una empresa puedan diagnosticar fácilmente los fallos de fresadoras.
- Utilizar Java para Android para desarrollar dicha aplicación.
- Realizar una interfaz gráfica usable y amigable para el usuario.
- Que dicha interfaz tenga los mismos colores, la misma fuente y logo de la empresa.
- Que la aplicación conecte con una API REST para la obtención de las alarmas activadas.
- Representar el título, descripción, preguntas y respuestas de las alarmas en ficheros JSON.
- Que la aplicación parsee dichos ficheros.[Cod. archivo][Norma ISO][Leer JSON]
- Guardar el registro de la actividad que va realizando el usuario en la aplicación, dentro de un archivo csv. La actividad se corresponde a las alarmas que está consultando, el camino de preguntas que va siguiendo y las respuestas que va eligiendo. Todo ello acompañado de un timestamp.
- Que la aplicación muestre al usuario las alarmas en inglés o español según el idioma de su móvil. Siempre y cuando exista una versión de la alarma en inglés.
- Que el usuario también pueda cambiar el idioma dentro de la aplicación, eligiéndolo en la pantalla de inicio.

III - Conceptos teóricos

A continuación se explican brevemente algunos conceptos que se usan en el trabajo y sirve de ayuda entenderlos.

1. JSON

JSON (JavaScript Object Notation) es un formato de texto ligero que representa datos estructurados y se usa para su intercambio.

Puede ser utilizado independientemente de JavaScript, ya que muchos contextos de programación tienen la capacidad de parsearlo y generarlo. Es una alternativa al lenguaje XML, siendo JSON más fácil de usar.

Los tipos de datos que puede representar son: números, cadenas (string), booleanos, null, arrays de cualquiera de los tipos anteriores y propios objetos JSON.[JSON wikipedia][Intr. JSON][Info JSON]


```
{
    "Número": 713,
    "Título":"AUTOMÁTICO BOMBA ASPIRACIÓN SALTADO",
    "Descripción":"La entrada E461 ha cambiado a 0, indicando que el térmico G11-Q21 se ha saltado.",
    "Preguntas":[
        {
            "Id":1.0,
            "Texto":"¿Fisicamente, el térmico G11-Q21 se ha saltado por consumo?",
            "Imagen":"",
            "Respuestas":[
                {
                    "Id":"A",
                    "Texto":"Si, está saltado.",
                    "Camino":1.1,
                    "Mensaje":""
                },
                {
                    "Id":"B",
                    "Texto":"No, no está saltado.",
                    "Camino":0.0,
                    "Mensaje":"Comprobar cableado circuito auxiliar del térmico y la entrada del autómata."
                }
            ]
        },
        {
            "Id":1.1,
            "Texto":"Si se rearma a mano, ¿se vuelve a saltar de nuevo al poco tiempo?",
            "Imagen":"",
            "Respuestas":[
                {
                    "Id":"A",
                    "Texto":"Si, se vuelve a saltar.",
                    "Camino":0.0,
                    "Mensaje":"Comprobar consumo de bomba y/o cambiarla."
                },
                {
                    "Id":"B",
                    "Texto":"No, no se vuelve a saltar.",
                    "Camino":0.0,
                    "Mensaje":"Dejar bomba en observación."
                }
            ]
        }
    ],
    "Imágenes":[
        "a713",
        "a713_2"
    ]
```


Ilustración 1: Ejemplo de la alarma 713

2. API REST

API REST (Application Programming Interface Representational State Transfer) es una interfaz entre sistemas que utiliza el protocolo HTTP para obtener datos o realizar operaciones sobre esos datos en cualquier formato posible, como los mencionados anteriormente JSON y XML.

Las operaciones más importantes que podemos hacer son: POST (crear), GET (leer y obtener), PUT (modificar) y DELETE (eliminar). Para acceder o manipular esos datos u objetos de la REST se utiliza la URI, que es su identificador único.[API REST][REST][API REST 2]

IV - Técnicas y herramientas

La herramienta de desarrollo principal que se ha usado para llevar a cabo el proyecto ha sido el programa Android Studio.[AS] Se trata de un entorno de desarrollo integrado que fue anunciado por Google en el año 2013, y por tanto es el IDE oficial para la creación de aplicaciones para Android. Con ella se ha programado la funcionalidad de las diferentes pantallas y su diseño gráfico.[Botones][Creación App][SplashScreen][Agregar botones][Spinner][Cambiar texto Spinner]

También se ha utilizado Eclipse para el desarrollo de un prototipo inicial de la aplicación, y StarUML para su diagrama de clases correspondiente. Se trataría de una versión en modo texto para comprobar la correcta navegación entre las preguntas de cada alarma.

Para la creación de los ficheros JSON se ha usado los editores de código fuente Notepad++ y Sublime Text.

Y para la realización del prototipo de la interfaz gráfica de la aplicación se ha utilizado el programa Pencil.

1. *Metodología usada*

La metodología usada para el desarrollo del software ha sido Scrum, planificando y realizando tareas cada dos semanas. Este control de las tareas se ha llevado a cabo con un repositorio en GitHub, realizando un commit por cada tarea.

Además, en este repositorio se encuentran todos los archivos que se han ido utilizando durante el trabajo.

Para almacenar y estructurar la información de las alarmas se ha utilizado los ficheros JSON mencionados en los otros apartados.

Para representar una alarma en el proyecto de Android Studio se ha empleado un modelo de clases Answer, Question y Alarm, con sus diferentes atributos y métodos.

Y para guardar las alarmas que se van parseando se ha creado una clase singleton, para que de esta forma se tenga un acceso global a la misma. Dicha clase utiliza un LinkedHashMap, siendo las claves los números de las alarmas y los valores los objetos alarmas. Se ha decidido que sea esta estructura de datos para que las alarmas se guarden en el mismo orden en el que se parsean. [Enviar datos]

2. *Alternativas estudiadas*

Alternativa	
Librería externa para parsear JSON	

Tabla 1: Alternativas estudiadas

2.1. Librería externa para parsear JSON

Al principio me descargué de Internet una librería para parsear ficheros JSON, llamada “json-simple-1.1.jar”[Librería JSON], que la utilicé en la aplicación en modo texto desarrollada en Eclipse. Cuando empecé con Android Studio mi intención era también usarla, pero descarté esta

opción porque Android ya tiene librerías propias para parsear JSON.

V - Aspectos relevantes del desarrollo del proyecto

El ciclo de vida utilizado para el desarrollo del proyecto ha sido Scrum, que es un tipo de método ágil. Este ha consistido en que a partir de unos requisitos para el avance de la aplicación, se han ido realizando sprints o tareas cada dos semanas para el cumplimiento de dichos requisitos. De esta forma se obtiene un "entregable" después de ese tiempo y se va avanzando en el desarrollo de la aplicación. Después de los dos primeros meses el tiempo entre sprints pasó a ser de una semana, para llevar un mayor control del proyecto.

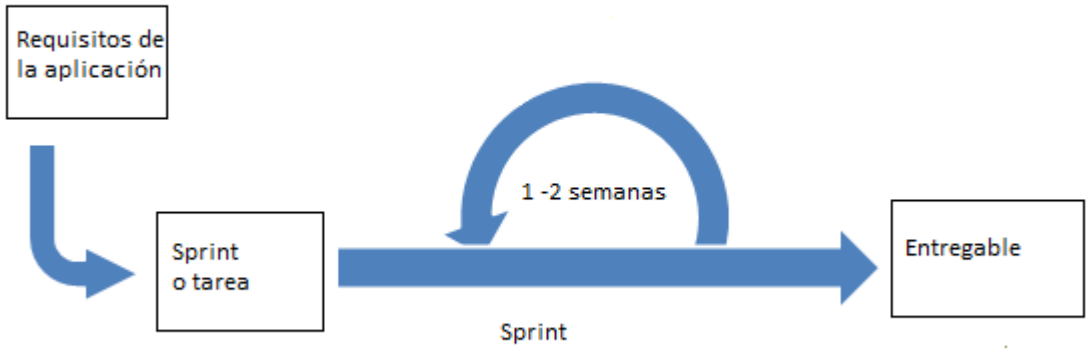

Ilustración 2: Modelo Scrum

Durante este desarrollo se ha seguido una serie de fases:

- Fase de análisis: en esta primera fase se toman las decisiones de cómo hacer el proyecto y qué pasos se han de seguir. Como por ejemplo de qué forma representar y estructurar las alarmas, cómo debe ser la interfaz de la aplicación, etc.
- Fase de diseño: en esta segunda fase se realiza el prototipo de la aplicación con la herramienta Pencil y se elabora un primer proyecto en Eclipse que representa la aplicación en modo texto.
- Fase de implementación: en esta última fase se realiza el desarrollo de la aplicación en la herramienta Android Studio, teniendo en cuenta todo lo que se ha hecho anteriormente. [Scrum]

1. Fase de análisis

En esta fase la parte más importante fue la "traducción" de la información de las alarmas contenida en una hoja de cálculo a la representación de dicha información en ficheros JSON:

Las alarmas son identificadas por un número, poseen un título y una descripción. En la hoja de cálculo algunas alarmas tienen una nota a tener en cuenta. Esta nota está representada en el fichero JSON dentro del apartado "Descripción", como si formara parte de esta.

Las preguntas están identificadas también por un número, que en el JSON es de tipo double, y que gracias a esto podemos diferenciar dos tipos de preguntas: las principales que serían las 1.0, 2.0, 3.0, etc., y las preguntas que llevan en el enunciado una posible solución y derivan de las principales, que serían las 1.1, 1.2, 2.1, etc. Estas últimas casi siempre acaban con la frase "¿Desaparece la alarma?".

De las respuestas comentar que en el JSON se identifican por letras: A, B, C, etc. y que cada una de ellas tiene un camino a seguir. Este camino es un valor de tipo double que puede ser 0.0 si la respuesta ya no tiene continuación, en ese caso se mostraría un mensaje final. O puede ser el id de una pregunta, que en este otro caso se mostraría dicha pregunta[Long]. Esta información en la hoja de cálculo se muestra al lado de la respuesta: en el primer caso es una frase que sería el mensaje final, y en el segundo caso aparece la palabra "continuar" si se debe seguir con la siguiente pregunta o directamente el número de la pregunta a la que se debe saltar.

También los ficheros JSON contienen un array con el nombre de los archivos de las imágenes que están asociadas a la alarma. Una pregunta también puede tener asociada una imagen, y está representada en el fichero de la misma forma.

Por parte de la interfaz se decidió que tenga un splash screen que contenga el nombre de la aplicación y de la empresa. Una pantalla de inicio donde se pueda elegir una alarma, otra pantalla donde muestre la información correspondiente de dicha alarma, otras que muestren las preguntas y respuestas, y una pantalla final en la que aparezca el mensaje de la solución o lo que debe hacer el usuario. Estas pantallas están explicadas con más detalle en el siguiente apartado.

2. Fase de diseño

En esta fase se realiza un prototipo de la interfaz comentada anteriormente, y compuesta por las siguientes pantallas:

- Splash screen: se trata de una imagen en la que aparece arriba el nombre de la aplicación, Diagnóstico Fresadoras, y abajo el nombre de la empresa: Correa. Primero se escribieron estas palabras en un Word con la tipografía y el color de la empresa: "Korataki" y un azul oscuro que su código hexadecimal es #00395D. Se hizo una captura y se guardó la imagen para poder ser utilizada en Android Studio. El splash screen dura 2000 milisegundos, o lo que es lo mismo, dos segundos.

  Comentar que el código hexadecimal del color se sacó gracias a la herramienta selector de color del programa Paint, utilizando una imagen de la empresa. Y por último usando una página web que te dice este código a partir del RGB.

  Ficheros correspondientes en Android Studio: SplashActivity.java y activity_splash.xml.

- Inicio: en esta pantalla aparece el nombre de la aplicación arriba, un spinner donde poder elegir la alarma que se quiera consultar y un botón "Aceptar" que te lleva a la siguiente pantalla. Las alarmas en el spinner están identificadas por su número y título.

  Ficheros en Android Studio: MainActivity.java y activity_main.xml.

- Información de la alarma: esta pantalla contiene el número, título, descripción e imágenes de la alarma que se ha elegido. El número y título aparecen arriba de la pantalla de manera fija, y de esta forma aparecen también a lo largo del recorrido de las preguntas y en el mensaje final. Debajo del número y título se muestra la descripción y las imágenes. Como pueden ser varias imágenes, nos podemos mover entre ellas de forma horizontal. También se puede hacer zoom a cada una de ellas. Por último, en la parte baja de la pantalla se encuentra un botón de comenzar que inicia las preguntas de la alarma.

  Ficheros en Android Studio: InfoAlarmaActivity.java y activity_inicio_alarma.xml.

- Preguntas y respuestas: esta pantalla muestra la pregunta, su imagen que tiene asociada en caso de que tenga una y las respuestas posibles que puede elegir el usuario. En la parte de arriba sigue apareciendo el número y título de la alarma como se ha mencionado anteriormente. Las respuestas son botones que te llevan a la misma pantalla, pero con los datos cambiados. Es decir se carga otra pantalla con la misma apariencia, pero que con-

tiene otra pregunta con su imagen y sus respuestas correspondientes.

Ficheros en Android Studio: QuestionsActivity.java y activity_questions.xml.

- Mensaje final: en esta última pantalla se muestra un texto con la conclusión final, que puede ser un mensaje diciendo que el problema se ha resuelto o una posible solución que debe realizar el empleado para resolverlo. En la parte de abajo de la pantalla hay un botón "Inicio" que te lleva a la primera pantalla, y así poder elegir otra alarma si se desea.

También en esta fase se realiza una aplicación simple en modo texto con el programa Eclipse. Los objetivos de crear esta aplicación previa a la principal es diseñar el modelo de clases que representa la información de las alarmas, comprobar el buen funcionamiento de los archivos JSON y el volcado de su contenido a ese modelo de clases.

El proyecto de Eclipse consta de la carpeta src con el modelo de clases y dos archivos JSON con el contenido de dos alarmas para poder hacer las pruebas. Los datos de estas alarmas son de una versión antigua, y por tanto no son los utilizados en el proyecto de Android Studio. La carpeta src se divide en:

- Paquete "ejecucion", donde se encuentra la clase main. En ella se parsea el contenido de los ficheros JSON y se guarda en los diferentes objetos creados que representan las alarmas, preguntas y respuestas. También hay un bucle para ir navegando entre las preguntas de una alarma.
- Paquete "modelo", donde se encuentran las clases que representan las alarmas, preguntas y respuestas. Estas clases solo contienen atributos y sus métodos get y set. Los atributos son por ejemplo el id y el texto de las preguntas y respuestas, el número y título de la alarma, etc.

Al ejecutar el programa muestra por pantalla el número, título, descripción y la primera pregunta de la alarma. Se debe introducir por teclado el id de la respuesta a elegir, y te muestra la siguiente pregunta en función de la respuesta que se haya elegido. Termina cuando se elige una respuesta que no tiene continuación y solo muestra un mensaje final.

3. Fase de implementación

La fase de implementación abarca todo el desarrollo del proyecto en Android Studio, es decir, la creación de la aplicación Android con su interfaz gráfica y su funcionalidad. El proyecto se divide principalmente en tres carpetas: java, assets y res.

- En la carpeta java se encuentra la funcionalidad de la aplicación, con el modelo de clases y los activity. Estos últimos representan cada pantalla de la app, mencionadas anteriormente en la fase de diseño. El modelo de clases está dentro de una subcarpeta llamada "modeloAlarmas" y consta de las siguientes clases:
  - Answer.java que representa las respuestas de cada pregunta. Tiene los siguientes atributos y métodos:
    - id: se trata del identificador de la respuesta. Si es la respuesta A o B o C, etc.
    - text: el texto de la respuesta.
    - next: el id de la pregunta con la que debe continuar. Este atributo vale -1.0 en caso de que ya no haya continuación, que entonces se mostraría un mensaje.
    - finalMessage: conclusión o posible solución que muestra la aplicación después de realizar varias preguntas.

- Métodos get y set de los atributos anteriores.

- Question.java que representa las preguntas de las alarmas. Consta de los siguientes atributos y métodos:
  - id: identificador de la pregunta.
  - text: texto de la pregunta.
  - images: un ArrayList con las imágenes correspondientes a esa pregunta.
  - answers: otro ArrayList con las respuestas de la pregunta.
  - Métodos get y set de los atributos anteriores, y métodos add para los ArrayList.
- Alarm.java que representa una alarma en concreto. Sus atributos y métodos son:
  - num: el número de la alarma.
  - title: título de la alarma.
  - desp: la descripción de la alarma.
  - images: igual que en la clase anterior, son las imágenes correspondientes a la alarma.
  - questions: las preguntas de la alarma.
  - Métodos get, set y add de los atributos anteriores.
- AlarmTable.java se trata de una clase singleton que almacena las alarmas que va parseando la aplicación y un diccionario con todas las imágenes que se usan. Es singleton para que se tenga un acceso global y así poder acceder desde cualquier activity a los datos de las alarmas. Tiene los siguientes atributos y métodos:
  - instance: la instancia de la clase.
  - alarms: un Map que almacena las diferentes alarmas de forma ordenada según se hayan parseado. La clave es el número de la alarma y el valor el propio objeto alarma.
  - diccImages: un Map con todas las imágenes de las alarmas y preguntas. La clave es el nombre del fichero y el valor es un Integer que representa la ruta donde se encuentra el fichero (R.drawable.X). Se ha decidido almacenar las imágenes de esta forma porque tenía problemas para obtener la ruta del fichero a partir de su nombre. Y por tanto se ha creado un diccionario "a mano" para poder acceder a las imágenes de una forma más simple.
  - Métodos get y add de los atributos anteriores. Y también un método llamado fillDiccImages que rellena el diccionario de imágenes de forma completa.

Los activity que representan cada pantalla de la aplicación son los siguientes:

- SplashActivity: se trata del splashscreen, una imagen que se muestra cuando arrancas una aplicación y que dura un corto período de tiempo. En este caso dura dos segundos y es una imagen que ocupa la pantalla completa gracias al parámetro "FLAG_FULLSCREEN".
- MainActivity: es la pantalla de inicio donde eliges la alarma que quieres consultar. Internamente se realizan varias funciones:
  - El método createRegistryFile crea un archivo csv, en caso de que no exista, para guardar el registro de la actividad que vaya realizando el usuario. Utiliza la clase OutputStreamWriter para crear ese archivo y escribir información. Para compro-

bar si el archivo existe se ha creado el método existsFile, que usa a su vez el método fileList() que devuelve el nombre de los ficheros que tiene internamente la aplicación.

- El método getDataApiRest obtiene la información de las alarmas del Api Rest. Mediante una url y una cabecera saca los datos correspondientes. Utiliza la clase HttpsURLConnection para establecer la conexión, InputStream y InputStream-Reader para obtener y guardar el contenido que devuelve, y JsonReader para parsear dicho contenido y quedarnos solo con lo que nos interesa.
- El método createJsonFromAssets crea un String en formato JSON a partir de los archivos encontrados en la carpeta assets. Según el idioma del teléfono móvil escoge el fichero JSON en español o en inglés. Para obtener el idioma se utiliza la clase Locale y los métodos getDefault y getLanguage. En caso de que no exista una versión en inglés directamente escoge la versión en español. Utiliza la clase InputStream para extraer el contenido de los archivos.
- Una vez tenemos el JSON lo parseamos en el método getAlarm y creamos el objeto alarma con sus preguntas y respuestas.
- Estos dos últimos puntos se realizan una vez elegida la alarma en el spinner y pulsar el botón “Aceptar”. También se añade el objeto alarma recientemente creado al Map de alarmas de la clase AlarmTable. Finalmente se pasa como dato el número de la alarma al siguiente activity. Para pasar datos de un activity a otro se usa la clase Intent.

- InfoAlarmaActivity: la pantalla donde se muestra la información principal de una alarma: número, título, descripción y sus imágenes correspondientes. Tiene la siguiente funcionalidad:
  - El método fillData rellena los TextView con la información anterior.
  - Y el método fillImages crea un LinearLayout de forma dinámica con las imágenes de la alarma. Estas las obtiene a partir del diccionario de la clase AlarmTable mencionado anteriormente. Se puede hacer zoom a estas imágenes gracias a una importación llamada PhotoViewAttacher.
  - Finalmente al pulsar el botón “Comenzar” se pasan como datos el número de la alarma y el id de la primera pregunta.
- QuestionsActivity: pantalla que muestra las preguntas con sus imágenes, en caso de que tenga, y con sus respuestas. Consta de la siguiente funcionalidad:
  - El método fillData que tiene la misma función que el mencionado anteriormente.
  - El método fillButtons crea un LinearLayout dinámicamente donde va añadiendo tantos botones como respuestas tiene la pregunta. Para manejar las acciones que realizan los botones al pulsarlos se utiliza la clase ButtonsOnClickListener.
  - La clase mencionada anteriormente sirve para realizar una acción u otra en función de lo que valga el atributo next de la respuesta. Si vale -1.0 el botón te lleva al activity del mensaje final, pasando el susodicho mensaje como dato. Si no, el botón te lleva a otra pregunta y se pasa el id de esta como dato.

    Esta clase también tiene el método writeRegistry que escribe en el fichero csv, creado anteriormente, el número de la alarma, la pregunta y la respuesta pulsada, además de la marca temporal en la que se ha realizado. Este método se ejecuta cuando el usuario pulsa una de las respuestas.
- MessageFinalActivity: pantalla que muestra el mensaje final.

- Tiene un método fillData como los mencionados en los activity anteriores.
- Y el botón “Inicio” al pulsarlo te lleva a MainActivity por si se quiere elegir otra alarma.

- En la carpeta assets se encuentran todos los ficheros JSON que contienen la información de cada una de las alarmas. Varias de ellas tienen una versión en inglés, con los textos de cada apartado en este idioma. Las alarmas que tienen las dos versiones tienen por tanto dos ficheros, uno se llama Alarma XXX.json y el otro Alarm XXX.json.
- La carpeta res contiene principalmente las imágenes que utiliza la aplicación y los layout de cada pantalla.
  - Los nombres de los archivos de las imágenes siguen el siguiente formato: empiezan por la letra “a” seguido del número de la alarma a la que corresponde la imagen y seguido de barra baja y un número en caso de que una alarma tenga más de una imagen Si la imagen es de una pregunta de la alarma entonces tiene el mismo formato que el anterior más una “q” seguido del id de la pregunta y de “a”o “b” o “c”, etc. en caso de que esa pregunta tenga varias imágenes. Por ejemplo: la imagen a712q8_3.jpg corresponde a la pregunta 8.3 de la alarma 712, y esta pregunta solo tiene esa imagen.
  - Los layout definen la interfaz gráfica de la aplicación, o lo que es lo mismo, el diseño gráfico de cada uno de los activity de la carpeta java. Se tratan de ficheros XML y son los siguientes:
    - activity_splash: es el layout de SplashActivity. Esta formado por un RelativeLayout con fondo de color blanco y un ImageView que se trata del splashscreen.
    - activity_main: es el layout de MainActivity. Esta formado por un RelativeLayout, dos TextView para mostrar el título de la aplicación y una frase, un spinner donde se encuentran las catorce alarmas a elegir y un Button para continuar a la siguiente pantalla.
    - activity_inicio_alarma: layout de InfoAlarmaActivity. Tiene un LinearLayout arriba con un TextView donde se muestra el número y título de la alarma. Abajo un RelativeLayout con un Button, un TextView y un HorizontalScrollView para mostrar la descripción y las imágenes. De esta forma si la alarma tiene varias imágenes te puedes desplazar horizontalmente por la pantalla para verlas todas. Los ImageView de estas imágenes se crean de forma dinámica ya que cada alarma tiene un número distinto de imágenes.
    - activity_questions: es el layout de QuestionsActivity. Este también tiene un LinearLayout arriba para mostrar el título de la alarma. Y debajo de ello tiene un RelativeLayout con un TextView para el texto de la pregunta, y un ScrollView con un LinearLayout para las respuestas. Los Button que van dentro que corresponden a cada una de las respuestas se crean dinámicamente, ya que hay preguntas que tienen más de dos respuestas.
    - activity_message_final: layout de MessageFinalActivity. Su contenido principal está formado por un RelativeLayout con un TextView para el texto del mensaje y un Button debajo para volver a la página de inicio.

    Comentar por último que algunos textos como el nombre de la aplicación y el número y título de las alarmas que aparece de forma fija encima de las pantallas, tienen el color de la empresa mencionado anteriormente: #00395D.

El orden en el que se fue realizando el desarrollo de esta parte del proyecto, la de Android

Studio, es el mismo que en el que se ha ido explicando. Primero la pantalla de inicio, después la de la información, luego la de preguntas y por último la del mensaje final. El SplashActivity se realizó una vez que se tenía todo lo anterior.

Poco a poco también se han ido cambiando diferentes aspectos de los activity y los layout a medida que pasaban las semanas. Estos cambios están reflejados en el repositorio de GitHub.

4. Manual de usuario

En este apartado se explica cómo un usuario debe usar la aplicación, aunque se trate de una app sencilla:

Al arrancar la aplicación el usuario debe elegir en la lista desplegable la alarma que se ha activado en la fresadora. Tiene que pulsar en dicha lista y moverse de arriba a abajo para poder ver todas las alarmas. Estas aparecen indicadas por su número y título para poder ser reconocidas fácilmente. Cuando haya encontrado la alarma que se ha activado debe pulsar en ella. Una vez elegida, se pulsa el botón “Aceptar”.

A continuación aparece la información principal de la alarma elegida. Si hay más de una imagen el usuario puede visualizar todas moviéndolas de izquierda a derecha. También puede hacer zoom en una de ellas pulsándola dos veces consecutivas. Si se ha equivocado de alarma siempre puede volver atrás con el botón de volver del propio móvil y elegir otra. Y una vez que el usuario haya leído la información correspondiente a la alarma elegida, debe pulsar el botón “Comenzar” para empezar con las preguntas.

Saldrá la primera pregunta, con sus imágenes si tiene y sus respuestas. La interacción con las imágenes es la misma que en la pantalla anterior. Para elegir una respuesta se debe pulsar en ella, ya que son botones, y te llevará a la siguiente pregunta o al mensaje final.

El mensaje final es un texto plano para que simplemente el usuario lo lea. Si se quiere volver a la primera pantalla para elegir otra alarma se debe pulsar el botón “Inicio”.

VI - Trabajos relacionados

Algunas aplicaciones que se encuentran en Play Store relacionadas con fresadoras son las siguientes:

- “Cálculos de Fresado”: permite calcular diferentes parámetros para realizar los cortes con la fresadora.
- “Operador de torno: simulador”. Realiza una simulación gráfica en 3D de cómo usar una fresadora.
- “Milling Calculator”: otra aplicación, pero esta vez en inglés, que te permite hacer cálculos y te muestra unos resultados, antes de hacerlo con una fresadora de verdad.

La mayoría de aplicaciones que hay en Google Play sobre fresadoras son de este tipo, para poder hacer cálculos antes de llevarlos a la práctica.

Otras aplicaciones que tienen que ver con este TFG son los asistentes para solucionar problemas de máquinas. Algunas de ellas son:

- “Lavadoras Asistente Reparación”: un asistente que te ayuda a averiguar el problema de tu lavadora e incluso solucionarlo.
- “EcoWash 360”: otro asistente para lavadoras similar que el anterior, pero que además te permite controlar el funcionamiento de ellas
- “EOBD Facile - Diagnóstico del Coche OBD2 & ELM327”: esta aplicación te permite

diagnosticar los problemas que puede tener el motor de tu coche.

VII - Conclusiones y líneas de trabajo futuras

Las conclusiones sobre las herramientas utilizadas durante el proyecto son:

- El lenguaje JSON es de mucha utilidad a la hora de estructurar datos. Es más sencillo, más fácil de comprender y de parsear que el XML.
- El IDE Android Studio me parece una potente herramienta para la creación y desarrollo de aplicaciones. Te permite una gran variedad de opciones y funcionalidades para construir tu proyecto.

Las conclusiones sobre los resultados del proyecto son:

- Me parece buena idea "traducir" la colección de preguntas y respuestas que tiene la empresa sobre cada alarma, en una aplicación sencilla que te permita ir respondiendo y navegando sobre esas preguntas hasta llegar a una posible solución final. Me parece útil para los empleados de la empresa que dispongan de esta aplicación y puedan usarla cada vez que falle una fresadora.

El proyecto se puede mejorar o continuar en los siguientes aspectos:

- Que un empleado pueda iniciar sesión en la aplicación y pueda ver un registro de su actividad: las veces que ha utilizado la aplicación, cuando lo ha hecho, las alarmas que ha consultado, etc.
- Que se puede buscar por palabras clave la alarma correspondiente. Un usuario introduciría varias palabras que tengan que ver con el fallo de la fresadora, y la aplicación te mostraría la alarma o alarmas que más se aproximen a lo que haya escrito el usuario.
- El fichero csv donde se va guardando el registro de la actividad que realizan los usuarios va creciendo a medida que se utiliza la aplicación y se escogen diferentes respuestas. Es necesario el volcado del contenido del fichero a otra unidad de almacenamiento, ya sea la nube o la tarjeta SD del móvil.
- Al pinchar en una imagen que se muestre en pantalla completa. Y una vez se vea así poder hacer zoom pinchando dos veces consecutivas.
- A parte de las imágenes, introducir vídeos en la aplicación que le sirvan de ayuda al usuario a la hora de responder una pregunta. Estos vídeos ya están proporcionados por la empresa, pero nos lo enviaron a falta de pocas semanas de finalizar el plazo de entrega y no han podido ser implementados.
- Crear un menú lateral que te permite volver a la pantalla de inicio en cualquier momento de la aplicación o volver a una pregunta que ya se había contestado anteriormente y se quiere cambiar de respuesta. También que aparezca de algún modo todas las preguntas de la alarma que se haya elegido y así el usuario pueda moverse directamente a cualquiera de ellas. Al final y al cabo un menú para poder navegar libremente por la aplicación.

VIII - Referencias

Bibliografía

Activity: TuTutorial, Cómo pasar de un Activity a otro en una aplicación en Android Studio, 2017, https://www.youtube.com/watch?v=41SYosZP708

Layouts: Código Alonso, Tutorial 010 Programación Android: Layouts, organización de componentes con Android Studio., 2013, https://www.youtube.com/watch?v=DXhMcrA45q8&t=792s

Bibliografía: Cristina Batallas, Base de datos bibliográfica en Writer, 2011, https://www.youtube-.com/watch?v=19N97pi0Ky8
Cod. archivo: Usuarios de Stack Overflow, ¿Qué codificación debería usar en un XML con texto en español?, 2015, https://es.stackoverflow.com/questions/316/qu%C3%A9-codificaci%C3%B3n-deber%C3%ADa-usar-en-un-xml-con-texto-en-espa%C3%B1ol
Norma ISO: Colaboradores de Wikipedia, ISO/IEC 8859-1, 2017, https://es.wikipedia.org/w/index.php?title=ISO/IEC_8859-1&oldid=102959948
Leer JSON: Gerard Coll, Leer archivo JSON localmente en Android, 2013, http://dev4phones.-blogspot.com.es/2015/12/leer-archivo-json-localmente-en-android.html
JSON wikipedia: Colaboradores de Wikipedia, JSON, 2018, https://es.wikipedia.org/w/index.-php?title=JSON&oldid=106104589
Intr. JSON: Colaborador de la página, Introducción a JSON, 2018, https://json.org/json-es.html
Info JSON: Juan Sepulveda, Trabajando con JSON, 2017, https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON
API REST: BBVA Open4U, API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos, 2016, https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos
REST: Colaboradores de Wikipedia, Transferencia de Estado Representacional, 2018, https://es.wikipedia.org/w/index.php?title=Transferencia_de_Estado_Representacional&oldid=105107212
API REST 2: Fazt, Api Rest con Nodejs | ¿Que es Api Rest?, Clase 1, 2017, https://www.youtube.com/watch?v=wMwON-gwyVM&t=1s
API REST 3: Fazt, Api Rest con Nodejs | Verbos HTTP, Clase 2, 2017, https://www.youtube.-com/watch?v=ArdQcI2X1cc
AS: Colaboradores de Wikipedia, Android Studio, 2018, https://es.wikipedia.org/w/index.php?title=Android_Studio&oldid=107398073
Botones: Miguel, Creación dinámica de botones, 2017, http://umhandroid.momrach.es/creacion-dinamica-de-botones/
Creación App: TuTutorial, Cómo crear una aplicación para Android con Android Studio, 2017, https://www.youtube.com/watch?v=XKU7MlqmgWo&t=631s
SplashScreen: TuTutorial, Cómo añadir un Splash Screen o pantalla de bienvenida a tu aplicación Android, 2017, https://www.youtube.com/watch?v=fR1xcOxprSY
Agregar botones: TuTutorial, Cómo agregar botones a tu aplicación en Android Studio, 2017, https://www.youtube.com/watch?v=R-YMH5I677A
Spinner: Cesar Augusto Perez Tafur, Tutorial 12 Como usar un Spinner en mi aplicacion Android, 2016, https://www.youtube.com/watch?v=F9GV7vuIghw&t=273s
Cambiar texto Spinner: Gonzalo Eduardo Pérez Correa, Android - Cambiar el tamaño del texto de un spinner, 2016, https://www.youtube.com/watch?v=ciQnbxjP7SY&t=20s
Enviar datos: TuTutorial, Cómo enviar datos de un Activity a otro en Android Studio, 2017, https://www.youtube.com/watch?v=lAnpyZmrwX8
Librería JSON: Desconocido, Download json-simple-1.1.jar : json simple « j « Jar File Download, 2017, http://www.java2s.com/Code/Jar/j/Downloadjsonsimple11jar.htm
Scrum: Colaboradores de Wikipedia, Scrum (desarrollo de software), 2018, https://es.wikipedia.org/w/index.php?title=Scrum_(desarrollo_de_software)&oldid=107627551
Long: Usuarios de Stack Overflow, Convert Long into Integer, 2011, https://stackoverflow.com/questions/5804043/convert-long-into-integer

Índice alfabético

Impreso en Burgos el jueves, 14 de junio de 2018