Facultad
de
Ciencias

Desarrollo de componentes software para realidad aumentada en entornos industriales 4.0

(Development of software components for augmented reality for industry 4.0)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMATICA

Autor: Diego Portilla Tejería

Directora: Patricia López Martínez

Codirector: Héctor Fernández Román

Febrero - 2019

Resumen

Haciendo uso de todas las características que nos aporta la tecnología móvil, híbrida y *weareable* (*smartGlasses*), el objetivo del trabajo es desarrollar los componentes software necesarios para dotar de realidad aumentada a los servicios proporcionados por FIELDEAS, una plataforma que incluye soluciones globales en diferentes ámbitos de la movilidad. Estos componentes software servirán para dar solución a problemas actuales del sector industrial:

- Sobreimpresión de información en base a una posición, objeto o señal visual (Códigos QR, OCR, etc.) como ayuda a la actividad de un operario.
- Guiado y solución de problemas (Asistencia Técnica Remota): Poder realizar una asistencia online a un operario mediante capas de AR (imágenes, textos, vídeos y audios).
- Training en planta: Poder realizar tareas de tutorización desatendidas y autónomas para un empleado de una fábrica.
- Navegación: Poder mostrar las indicaciones necesarias para la consecución de una actividad que conlleve un desplazamiento.

En concreto, el software desarrollado consiste en la pasarela entre las diferentes aplicaciones de FIELDEAS y el dispositivo de realidad aumentada. La parte del lado de FIELDEAS ha sido realizada utilizando JavaScript y HTML como requisito técnico de su ya existente API de desarrollo. Para las vistas y la ejecución de las llamadas a los plugins que implementa FIELDEAS en su capa nativa (llamados "wrapper" o contenedores) se ha utilizado el lenguaje Java (Android), y también para establecer una comunicación con la aplicación cliente que se ha implementado en el dispositivo AR, cuya base tecnológica depende del modelo/fabricante del dispositivo AR.

Palabras clave: Realidad Aumentada, Industria 4.0, smartglasses, ATR

Abstract

Using every feature that hybrid mobile technologies and weareables (smartglasses) bring to us, the goal of this project is to develop the software components needed to give augmented reality to the services provided by FIELDEAS, a platform that includes global solutions in different fields of mobility. These software components will help us to solve problems in the industry like:

- Overlay data based on the current position, an object, or a visual signal (QR code, OCR, etc.) helping an operator with his job.
- Guide and problem solving (Remote Technical Assistance): Enabling online assistance to someone with AR layers (images, text, video and audio).
- Training in the field: AR could bring the possibilty to train new factory employees unattended and in their own.
- Navigation: Displaying the steps to follow to accomplish some activity which involves movement.

The software developed is the connection between one of the FIELDEAS applications and the AR device. The side of FIELDEAS is developed using JavaScript and HTML as technical requirement of its current development API. For the conections with the native plugins, called wrappers, implemented by FIELDEAS in their native layer and the graphical user interface, we must use the Java programming language (Android), and also to establish the conection between the client application in the AR device, whose technological base will depend on the model of the AR device.

Key Words: Augmented Reality, Industry 4.0, smartglasses, RTA

Índice

Índice de figuras

1. Introducción y antecedentes

Esta sección describe por una parte el contexto en el que se ha desarrollado el proyecto, así como algunos datos que muestran la relevancia actual de todo lo relacionado con la realidad aumentada (en adelante AR por su traducción al inglés) y, por otra parte, el proceso del entorno de producción que debe ser automatizado a lo largo del desarrollo del proyecto. Finalmente se desglosarán los objetivos del trabajo, y como se ha estructurado esta memoria.

1.1. La realidad aumentada

Durante estos últimos años la realidad aumentada es una de las tecnologías en alza dentro de la sociedad y del entorno empresarial, gracias a esto las grandes tecnológicas como son Google, Facebook y Apple entre otras, están apostando por la realidad aumentada, desplegando grandes presupuestos en este campo. Esto se cree que provocará que en los próximos años aumente aún más la presencia de esta tecnología y una parte de la Industria 4.0 [1], como es la realidad aumentada, se convierta en una realidad.

Para poder apreciar este gran crecimiento, según un artículo de Digi-Capital [2], "en el año 2016 las ganancias generadas conjuntamente por la realidad aumentada y la realidad virtual fueron de 6.100 millones de dólares, mientras que al año siguiente esa cifra se estableció en 13.900 millones de dólares, lo que supone un aumento de casi el 130 %". Al estar combinadas ambas tecnologías en estas cifras no se aprecia qué porcentaje se corresponde a cada una, pero según el artículo, la realidad virtual es la que sale ganando durante los últimos años.

En la gráfica siguiente, extraída de dicho artículo, se ven los ingresos esperados para los próximos años:

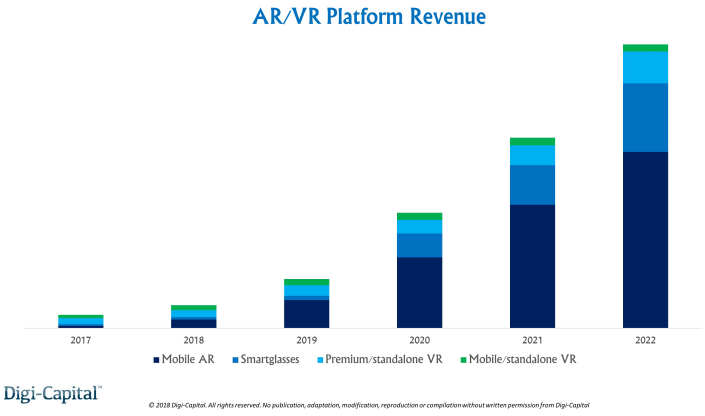

Figura 1: Ingresos de la realidad aumentada y virtual

Como se aprecia en la gráfica, el crecimiento de la realidad aumentada orientada a dispositivos móviles es espectacular. En menor medida también aumentan los ingresos por la venta de hardware específico como son las *smartGlasses*. Y vemos como la realidad virtual también crece al principio, pero luego se estabiliza.

En el artículo se prevé que los ingresos sean de 108.000 millones de dólares en el año 2021, de estos, la realidad aumentada supondrá 83.000 millones de dólares, este hecho representará un cambio en cuanto a la balanza VR/AR.

A continuación, en las gráficas mostradas en la figura 2, también extraídas del artículo, se muestran las previsiones en el número de aplicaciones que están realmente instauradas en las diferentes plataformas que soportan la realidad aumentada y virtual. A la izquierda incluyendo la realidad aumentada para móviles y a la derecha excluyéndola.

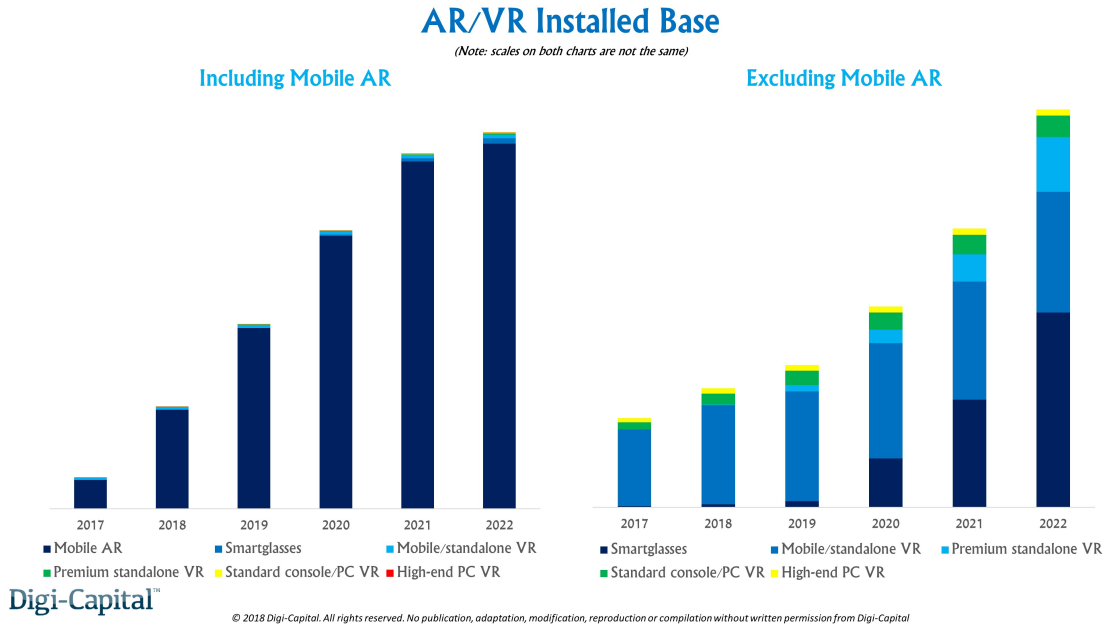

Figura 2: Número de aplicaciones de la realidad aumentada y virtual
(La escala de ambas gráficas no es la misma)

Como dicen en el artículo de Digi-Capital, "al final de este año 2018 el número de aplicaciones de la realidad aumentada activas en dispositivos móviles podría alcanzar los 900 millones de dispositivos, y se prevé que en el año 2022 se alcancen los 3.500 millones de aplicaciones". Lo que se ve en la comparación de ambas gráficas es que el porcentaje de la realidad aumentada para móviles es casi la totalidad de los usos de esta tecnología, esto es debido a que (para el público general), es más fácil disponer de un smartphone capaz de utilizar la realidad aumentada, que poseer dispositivos específicos como son las *smartglasses* o las gafas de realidad virtual.

Digi-Capital prevé que las *smartglasses* produzcan su proliferación en el mercado a partir del año 2020, y adjuntan que "esto será posible porque Apple lanzará unas *smartglasses* dependientes de sus dispositivos móviles", entonces las *smartglasses* se volverán una realidad para el público general.

1.2. FIELDEAS

El proyecto va a formar parte de FIELDEAS [3], una plataforma de movilidad empresarial sobre la que se implementan diversos procesos de negocio. Estos procesos pueden ser autónomos o integrados con los sistemas corporativos de las empresas. Gracias a FIELDEAS las empresas consiguen conectar con sus trabajadores, tanto de exterior como interior, de una forma totalmente segura, en tiempo real y con la máxima eficiencia para su negocio.

Con la realización del proyecto se permitirá a los operarios que utilicen los servicios de FIELDEAS, actualmente soportados en base a aplicaciones móviles, la posibilidad de eliminar la dependencia del teléfono móvil que tienen dichos operarios, agilizando así su labor y aumentando la seguridad de estos al permitirles usar ambas manos. Esto lo lograremos dotando a dichos operarios de un dispositivo (en este caso las *SmartGlasses*) a través del que podrán acceder a la misma funcionalidad que desde el móvil pero de una forma totalmente exenta del uso de las manos durante la realización de tareas, gracias a que dicho dispositivo será controlado mediante la voz o un flujo automático determinado por sus diversos sensores como geoposiciones, códigos QR o de barras y contadores de tiempo. Esto permitirá que el dispositivo móvil pueda ser apartado de la vista o guardado.

1.3. Objetivos

El objetivo del proyecto es proporcionar a los programadores de FIELDEAS de una interfaz que les facilite las tareas de programación de las aplicaciones, en concreto la programación de la comunicación entre el dispositivo móvil y el dispositivo AR.

Dado que el lenguaje de programación del dispositivo de realidad aumentada es Android (Java) y los programadores de FIELDEAS están orientados a tecnologías web y utilizan JavaScript para la programación de las aplicaciones, se tendrá que realizar también una interfaz o pasarela de comunicación entre estos dos lenguajes para eliminar la dependencia del lenguaje nativo a los programadores.

Adicionalmente a la abstracción en la comunicación entre ambos dispositivos, se implementarán una serie de funciones básicas para facilitar la programación del dispositivo AR, como mandar capturar una foto en el dispositivo de realidad aumentada y devolvérselo al móvil, o en el sentido contrario, mandar un audio desde el móvil hacia el dispositivo de realidad aumentada.

Finalmente se implementarán otras funciones más complejas, como un proceso de inventariado o el proceso de carga de una empresa de transporte, para añadirlas a la interfaz. Estas funciones se incluirán en una prueba de concepto junto con otras funciones simples donde se podrá visualizar el potencial alcanzable gracias al trabajo desarrollado.

Como se ha reflejado en los apartados anteriores, las previsiones de crecimiento de la realidad aumentada lo convierten en algo que vale la pena explotar, y al ser una tecnología en actual desarrollo supone una gran oportunidad de negocio para

FIELDEAS, que quiere ofrecer a sus clientes las posibilidades que brinda esta tecnología, que gracias al proyecto se podrá integrar fácilmente en los servicios actualmente prestados por la plataforma.

1.4. Organización de la memoria

Hasta ahora se ha introducido la definición de los objetivos y los antecedentes de la realidad aumentada en la sección 1. A continuación, se describirá la metodología de trabajo utilizada, en donde se añadirá un apartado con la planificación del proyecto según la metodología elegida conformando así la sección 2.

Después de definir cómo hemos trabajado, en la sección 3, se entrará en el qué hemos usado para trabajar, describiendo las herramientas software que hemos utilizado a lo largo del proyecto.

Una vez definido el cómo y el con qué, pasamos a la definición funcional y técnica. En la sección 4 describiremos lo implementado. En esa sección realizaremos una especificación de requisitos tanto funcionales como no funcionales, analizaremos los dispositivos de realidad aumentada que satisfacen dichos requisitos, e incluiremos una descripción detallada de las funciones a implementar.

Finalizada la captura de requisitos y realizadas las elecciones hardware, se diseñará en la sección 5 la arquitectura del proyecto, en donde se han utilizado los diagramas de clases para acompañar a su descripción. También se incluye el apartado de implementación en donde se exponen los problemas encontrados a lo largo del desarrollo.

Finalmente se han escrito unas líneas sobre el futuro del proyecto y se incluirá una breve conclusión en la sección 6 sobre todo lo aprendido durante el periodo de realización de este Trabajo de Fin de Grado.

2. Metodología de trabajo

Como en la empresa en la que se ha realizado el proyecto, FIELDEAS, utilizan MÉTRICA Versión 3 [4] como metodología, se han seguido sus directrices a la hora de realizar la etapa de análisis de requisitos, y de diseño y arquitectura del proyecto.

Al ser una metodología orientada a proyectos para instituciones públicas, la documentación de cada una de las etapas nombradas tiende a ser muy extensa, por esto, se ha optado por reducir dichos requisitos y formalizar los documentos de una manera menos extensa, con lo que solo crearemos el documento de requisitos funcionales y el de diseño técnico, y excluiremos el resto de los documentos definidos dentro de MÉTRICA v.3.

En cuanto a las etapas del ciclo de vida del desarrollo software, MÉTRICA v.3 sigue las etapas de la metodología en cascada típica:

- Fase de análisis de requisitos.
- Fase de diseño y arquitectura.
- Fase de construcción.
- Fase de pruebas.
- Fase de implantación y pruebas de aceptación.
- Fase de mantenimiento.

Como metodología de trabajo no aplicamos MÉTRICA v.3 en una versión pura, sino que se utilizan también técnicas de otras metodologías, como Kanban [5], en concreto el modo de organizar la gestión de las tareas a desarrollar en cada fase del proyecto.

Para ello, agrupamos los diferentes procesos según el estado en el que se encuentren: los que se están implementando en el momento, los que se van a implementar a continuación, las tareas que se han retrasado por depender de otras no completas, las tareas ya realizadas, y el resto de procesos por implementar. Este será el esquema que se siga a lo largo del proyecto, pero se pueden añadir o utilizar diferentes tipos de Kanban.

La planificación de cada fase del proyecto se debe realizar en base a dicha técnica. Las fases que se seguirán en el proyecto serán las definidas anteriormente, a excepción de la de mantenimiento ya que el alcance del proyecto no la contempla, aunque se tendrá en cuenta en la sección de líneas futuras.

El primer paso que debemos seguir para una planificación correcta y eficiente es el de identificar las tareas a realizar dentro de cada fase del proyecto. Lo siguiente será estimar el tiempo que se dedicará a cada una de esas tareas, según su dificultad y el tiempo necesario para llevarlas a cabo.

Las fases iniciales de captura de requisitos, diseño y arquitectura se realizaron junto con el codirector del proyecto, que supervisaba y validaba los documentos requeridos.

La planificación del proyecto se realizará utilizando las herramientas proporcionadas por el gestor de proyectos Jira [6], en donde estarán definidas nuestras tareas pendientes, empezadas, en proceso, pendientes de revisión y terminadas. Un ejemplo de pizarra Kanban utilizada durante el proyecto se muestra en la figura 3.

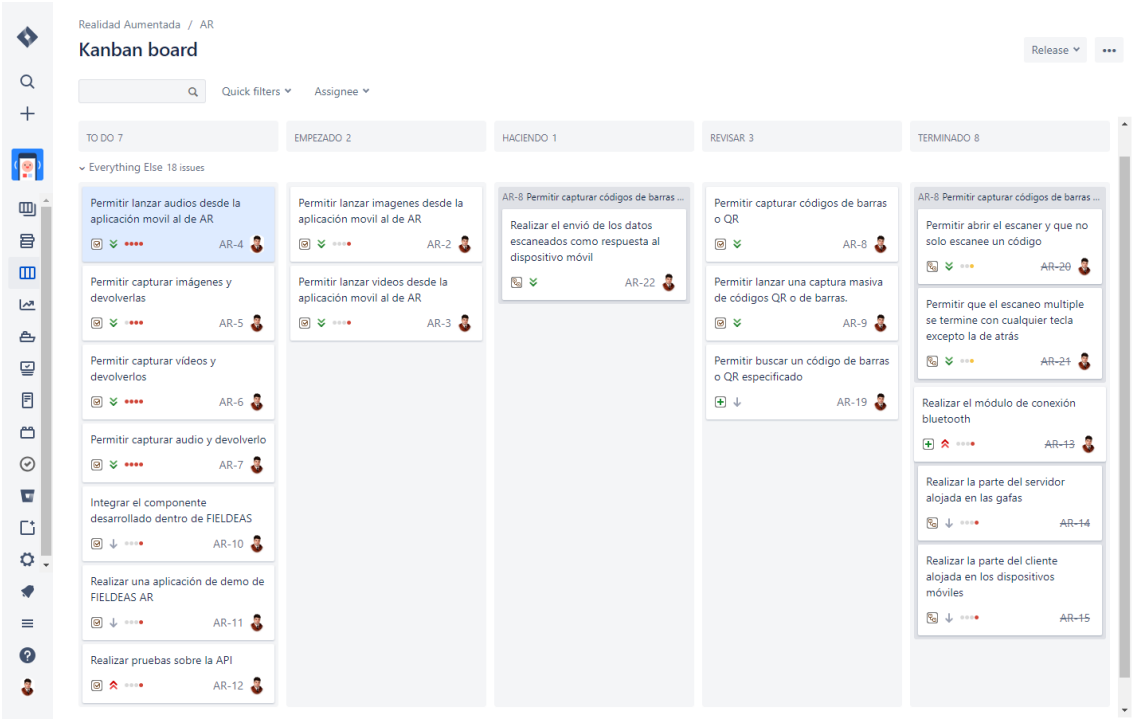

Figura 3: Captura de pantalla de la pizarra Kanban en Jira a mediados de proyecto

3. Tecnologías y herramientas

A continuación, se describen de forma general las herramientas y tecnologías utilizadas en el desarrollo del proyecto, sus iconos se muestran a continuación en la figura 4.

Figura 4: Iconos de las herramientas y tecnologías utilizadas

3.1. Android Studio

Android Studio [7] es el entorno de desarrollo integrado (IDE) principal sobre el que será llevado a cabo el proyecto. Cuenta con soporte para desarrollo de aplicaciones Android, tanto móviles como de escritorio, como para dispositivos del Internet de las Cosas (IoT). También tiene soporte para Gradle, refactorizaciones específicas de Android, plantillas y un gran soporte para los diversos servicios de Google integrados directamente en el IDE.

3.2. Android (Java)

Android [8] es el lenguaje escogido para realizar la mayor parte del proyecto. Se ha elegido como lenguaje porque los dispositivos con base Android son los más adecuados a este proyecto.

En gran parte, también se ha elegido porque ya se tienen conocimientos de Java (base de Android) adquiridos durante el grado, así como porque en FIELDEAS hay varios programadores especializados en el lenguaje que pueden ayudar con posibles problemas generados por el uso del mismo.

3.3. Sublime Text 3

Como editor de texto y de código (durante la realización de las pruebas finales del proyecto y para la aplicación de demostración de este) se ha elegido Sublime Text [9], una herramienta que se ha estado usando a lo largo del grado y que se usa en el entorno profesional por su gran versatilidad y fluidez. Al no ser un IDE como tal, no

tiene añadidos extra que pueden agregar pesadez al entorno de desarrollo. Aunque este editor nos permite añadir *plugins* como puede ser el autocompletado de código o el coloreado de sintaxis.

3.4. JavaScript

JavaScript [10] es el lenguaje para el que está desarrollado el proyecto, y este lenguaje será el encargado de llamar al componente nativo desarrollado durante este. JavaScript es un lenguaje de programación orientado a las tecnologías web (HMTL), es un lenguaje fácil de aprender. Gracias al proyecto podremos facilitar a los programadores de FIELDEAS, que usan este lenguaje, las posibilidades que nos ofrece la realidad aumentada, sin la complejidad que pueda suponer la utilización de los componentes nativos requeridos para esta tarea.

3.5. VueJS

VueJS [11] es el *framework* de JavaScript que actualmente están usando los desarrolladores de FIELDEAS, con este *framework* junto con el software desarrollado durante el proyecto, se realizará una aplicación para la realización de las pruebas finales del proyecto, así como para mostrar lo que hemos desarrollado.

3.6. Framework7

Framework7 [12] es un *framework* muy completo de HTML con el que los diferentes programadores de FIELDEAS desarrollan y así logran un estilo unificado entre las aplicaciones creadas por ellos de una forma sencilla y mantenida por desarrolladores externos.

4. Especificación de requisitos

La captura de requisitos para la especificación de la interfaz de comunicación con la interfaz de programación de aplicaciones del dispositivo de realidad aumentada se realizó mediante una serie de reuniones con el responsable del proyecto con el fin de identificar los requisitos básicos con los que la interfaz debía contar. Los requisitos funcionales y no funcionales de la aplicación que aparecen en las siguientes secciones son los aprobados tanto por el responsable del proyecto como por el equipo de desarrollo.

4.1. Requisitos funcionales

A continuación, se recogen los requisitos funcionales que la interfaz debía satisfacer. Cada uno aparece identificado con un código único y una descripción. Estos requisitos establecen el comportamiento de la interfaz, describiendo qué acciones pueden llevar a cabo los actores que utilizan el sistema.

Tabla 1: Requisitos funcionales

Código	**Requisito funcional**
RF01	La conexión entre el dispositivo AR y el dispositivo móvil, con la aplicación de FIELDEAS instalada, se realiza por medio de Bluetooth.
RF02	La conexión entre el dispositivo AR y el dispositivo móvil puede ser interrumpida a petición del operario.
RF03	El operario del dispositivo AR podrá indicar que desea pausar su trabajo para bien continuarlo más adelante, para terminar su jornada laboral, o porque desea terminar de utilizarlo para seguir de una forma no guiada.
RF04	Todas las acciones invocadas sobre el dispositivo AR permitirán al programador especificar un *callback* que se ejecute cuando se reciba la respuesta en el dispositivo móvil.
RF05	Se permitirá al dispositivo móvil solicitar que el dispositivo AR realice un escaneo en busca de un código QR o de barras, que será devuelto como respuesta al dispositivo móvil. En el caso de que el escaneo se cancele por el operario o por un error, esto se verá reflejado en la respuesta.
RF06	Se permitirá al dispositivo móvil solicitar que el dispositivo AR empiece a realizar escaneos hasta que encuentre un código pasado como parámetro de la solicitud o la acción sea cancelada. En el caso de que la acción sea cancelada, se tendrá que ver reflejado como respuesta.

Tabla 1: Requisitos funcionales

Código	**Requisito funcional**
RF07	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR realice un reconocimiento de caracteres sobre una foto capturada desde el mismo.
RF08	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR realice una foto. Dicha foto podrá ser confirmada o rechazada por el operario del dispositivo AR antes de ser enviada, además de la posibilidad de volver a realizar la captura en caso de ser rechazada.
RF09	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR realice una captura de vídeo. Dicha captura podrá ser confirmada o rechazada por el operario del dispositivo AR antes de ser enviada, además de la posibilidad de volver a realizar la captura en caso de ser rechazada.
RF10	Se permitirá al dispositivo móvil solicitar que el dispositivo AR realice una grabación de voz. Dicha grabación podrá ser confirmada o rechazada por el operario del dispositivo AR antes de ser enviada y la posibilidad de volver a realizar la grabación en caso de ser rechazada.
RF11	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR muestre una imagen por su interfaz de usuario.
RF12	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR reproduzca un vídeo por su interfaz de usuario.
RF13	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR reproduzca un archivo de audio al operario.
RF14	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR muestre un texto por pantalla. En el caso de que dicho texto supere cierto ancho en pantalla se podrá realizar un desplazamiento vertical sobre él utilizando los controles del dispositivo AR.
RF15	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR muestre un menú por pantalla. En el caso de que dicho menú supere cierto ancho en pantalla se podrá realizar un desplazamiento vertical sobre él utilizando los controles del dispositivo AR.

Tabla 1: Requisitos funcionales

Código	**Requisito funcional**
RF16	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR muestre una pregunta o texto por pantalla que tenga uno o dos botones con etiqueta.
RF17	Se permitirá que la reproducción multimedia, la respuesta a las preguntas, la respuesta a los mensajes y la interrupción de acciones pueda ser controlada mediante comandos de voz.
RF18	Se permitirá que la reproducción multimedia, la respuesta a las preguntas, la respuesta a los mensajes y la interrupción de acciones pueda ser controlada mediante los controles propios del dispositivo AR.
RF19	Se permitirá al programador lanzar desde el dispositivo móvil una petición para la creación de elementos de interfaz gráfica en el dispositivo AR.
RF20	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR lance un aviso a la aplicación móvil cuando el dispositivo AR alcance cierta posición en un rango especificado.
RF21	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR lance un aviso a la aplicación móvil cuando el dispositivo AR abandone cierta posición en un rango especificado.
RF22	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR muestre un mensaje por pantalla cuando el dispositivo AR alcance cierta posición en un rango especificado. Dicho mensaje podrá ser personalizado o uno por defecto establecido en el sistema.
RF23	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR muestre un mensaje por pantalla cuando el dispositivo AR abandone cierta posición en un rango especificado. Dicho mensaje podrá ser personalizado o uno por defecto establecido en el sistema.
RF24	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR establezca un disparador que se active cuando transcurra un número determinado de minutos y segundos.

Tabla 1: Requisitos funcionales

Código	Requisito funcional
RF25	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR establezca un disparador que se active a cierta hora.
RF26	Se permitirá al programador lanzar desde el dispositivo móvil una petición para que el dispositivo AR notifique al dispositivo móvil cuando su batería sea baja.
RF27	La conexión entre el dispositivo AR y el dispositivo móvil, con la aplicación de FIELDEAS, se realiza intercambiando mensajes en formato JSON.

4.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que definen características relacionadas con la seguridad, usabilidad o rendimiento que el sistema debe cumplir. Los requisitos no funcionales del sistema desarrollado se muestran en la siguiente tabla diferenciados por un código único, su descripción, su tipo y su importancia para el sistema.

Tabla 2: Requisitos no funcionales

Código	Requisito no funcional	Tipo	Importancia
RNF01	El sistema debe proporcionar mensajes de error que sean informativos y orientados al operario.	Usabilidad	Alta
RNF02	La interfaz gráfica de la aplicación deberá ajustarse a un uso típico de los operarios de una planta de producción.	Usabilidad	Alta
RNF03	El sistema contará con un manual destinado a los programadores.	Usabilidad	Media
RNF04	El dispositivo móvil debe tener instalada la aplicación.	Usabilidad	Alta
RNF05	La duración de la batería del dispositivo AR deberá alcanzar una jornada de trabajo normal.	Eficiencia	Media

Tabla 2: Requisitos no funcionales

Código	Requisito no funcional	Tipo	Importancia
RNF06	El uso del dispositivo no puede interferir con el trabajo normal del operario que lo esté utilizando en ese momento, ni con los operarios que interactúen con él.	Seguridad	Alta

La elección del Bluetooth para la comunicación entre el dispositivo de realidad aumentada y el dispositivo móvil ha sido realizada para ayudar a satisfacer el requisito RNF05, ya que la comunicación a través de Wifi es más costosa desde el punto de vista energético.

4.3. Elecciones hardware

Para escoger el dispositivo que satisfaciese las necesidades del proyecto, se estuvieron recogiendo características de diferentes dispositivos que se ajustaban a los requisitos previamente establecidos. Los modelos que se contemplaron fueron los siguientes:

- El modelo m100 [13] de Vuzix, que fue desestimado porque actualmente está siendo remplazado por su versión más moderna, el m300.
- Las Epson Moverio-BT-350 [14], que fueron rechazadas porque la batería no satisfacía las necesidades especificadas según el requisito RNF05 y no era ampliable, y también por la escasa documentación proporcionada por el fabricante.
- Las SED-E1 de Sony [15], que en principio eran la elección principal, gracias a su gran documentación y su precio más asequible, pero al igual que con muchas otras gafas que se observaron, la batería era un problema, ya que solo tenían soporte para 150 minutos de uso "normal", sin utilizar la cámara.

Finalmente se escogió el modelo actual de Vuzix, las m300 [16], que como se detalla a continuación, cumple con nuestras necesidades:

- Satisface los requisitos RF05, RF06, RF07, RF08 y RF09 al contar con una cámara que permite la captura de fotos y la grabación de vídeos.
- Al disponer de un micrófono satisface el requisito RF10.
- Al contar con la posibilidad de realizar escaneos de diversos códigos como QR o de barras, satisface los requisitos RF05, RF06 y RF07.
- Como también disponen de GPS, satisface RF20, RF21, RF22 y RF23.
- Al disponer de una batería de 160mA interna y otra externa de 860mAh ampliable, cubre las necesidades marcadas en el requisito RNF05.

- Dispone de altavoces por lo que se cumple con los requisitos de reproducción de sonidos RF10, RF12 y RF13.
- Como el modelo m300 cuenta con una API de reconocimiento de voz podemos satisfacer el requisito RF17.
- Los requisitos RF08 al RF12, del RF14 al RF17, el RF19, el RF22, el RF23, los no funcionales RF01, RF02 y RNF06 se corresponden con las indicaciones para la interfaz gráfica, que deberán tenerse en cuenta durante el desarrollo de esta y por sus usuarios.
- Gracias a que el modelo escogido funciona con Android como Sistema Operativo, disponemos de las funciones de control de tiempo que nos ofrece el lenguaje, con las que satisfacer los requisitos RF24 y RF25.
- Vuzix pone a nuestra disposición una serie de accesorios para su modelo m300, como pueden ser cascos que tengan soporte para las gafas y montura de seguridad entre otros, dichos accesorios nos permiten satisfacer nuestras necesidades en cada entorno y por lo tanto el requisito RNF06.
- Y, por último, gracias a que las gafas disponen de tres botones más un *trackpad*, podemos satisfacer el requisito RF18.

Se realizó una valoración de cada uno de estos modelos que fue entregada al director del proyecto, que decidió que el modelo m300 es, por sus ya citadas características, el que mejor satisface las necesidades del proyecto, y por lo tanto el modelo elegido para este.

4.4. Descripción detallada de funcionalidades

A continuación, se definirán de una forma detallada algunas de las diferentes funciones a desarrollar. Como forma de representar los flujos a seguir, se han elegido los diagramas de secuencia acompañados de una descripción.

Los objetos y componentes que participan en las interacciones mostradas en los siguientes diagramas se muestran en el modelo de dominio representado en la figura 5.

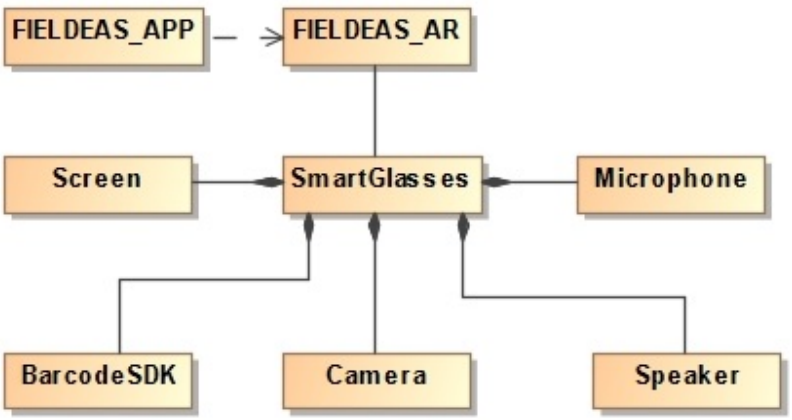

Figura 5: Modelo de dominio

En el diagrama de dominio vemos que FIELDEAS_APP, que representa la aplicación móvil, se conecta con FIELDEAS_AR, que representa el software instalado en las SmartGlasses. Se muestran, a su vez, los diferentes componentes que forman las SmartGlasses.

En primer lugar, se describe la función más básica, la conexión entre el dispositivo AR y la aplicación de FIELDEAS instalada en un dispositivo móvil y la posterior desconexión entre ambos dispositivos.

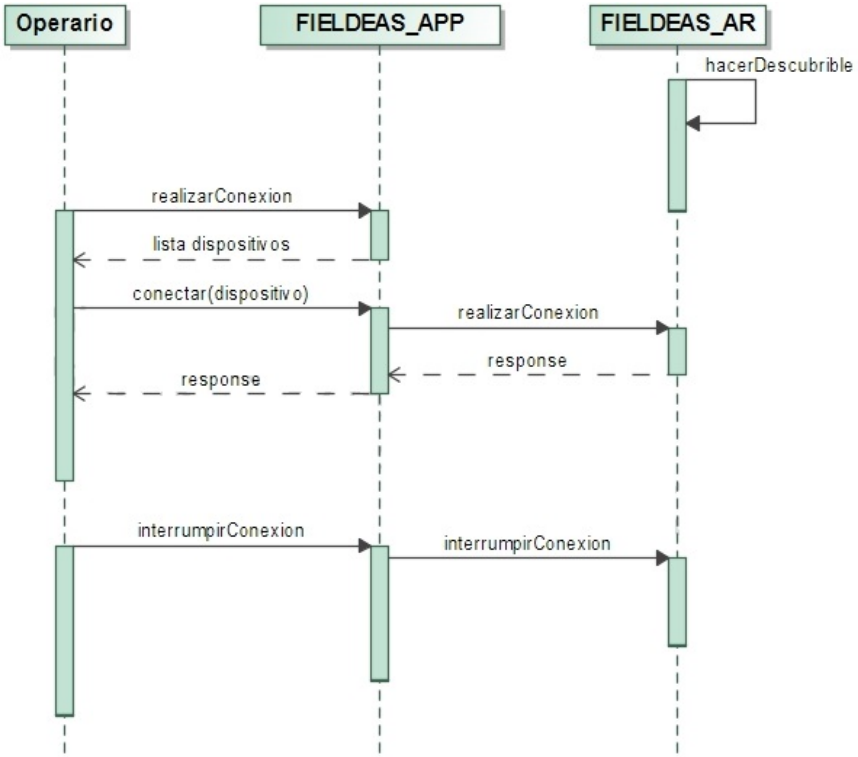

Figura 6: Realización e interrupción de la conexión

Como se observa en la figura, tras haber lanzado la aplicación en las SmartGlasses, se inicia automáticamente el descubrimiento Bluetooth del dispositivo AR. Acto seguido, a través de la aplicación móvil se realizará la llamada a realizarConexion(), la cual mostrará una lista de los dispositivos Bluetooth visibles en ese momento, se deberá seleccionar el dispositivo AR y se procederá a la conexión.

La llamada devolverá una respuesta, en caso de error este será reflejado. Esta llamada no cuenta con parámetros, pero al realizarse, el método busca dentro de los archivos de la aplicación un archivo de configuración en el que se especifican diversos valores de configuración como pueden ser el esquema de colores de la aplicación, entre otros. La respuesta en el caso en que todo vaya como se espera tendrá el siguiente formato:

```
{
  "error": 0,
  "descripcion": "La conexión se ha realizado correctamente"
}
```

En el caso en el que la búsqueda de dispositivo sea cancelada por el operario, la respuesta será la siguiente:

```
{
  "error": 1,
  "descripcion": "El usuario ha cancelado la búsqueda"
}
```

Cuando se requiera se puede interrumpir la conexión, lo que se realizará con una llamada a la función interrumpirConexion(). Dicha llamada no requiere de parámetros adicionales.

A continuación vemos en la figura 7 el proceso para realizar un escaneo de código de barras o QR.

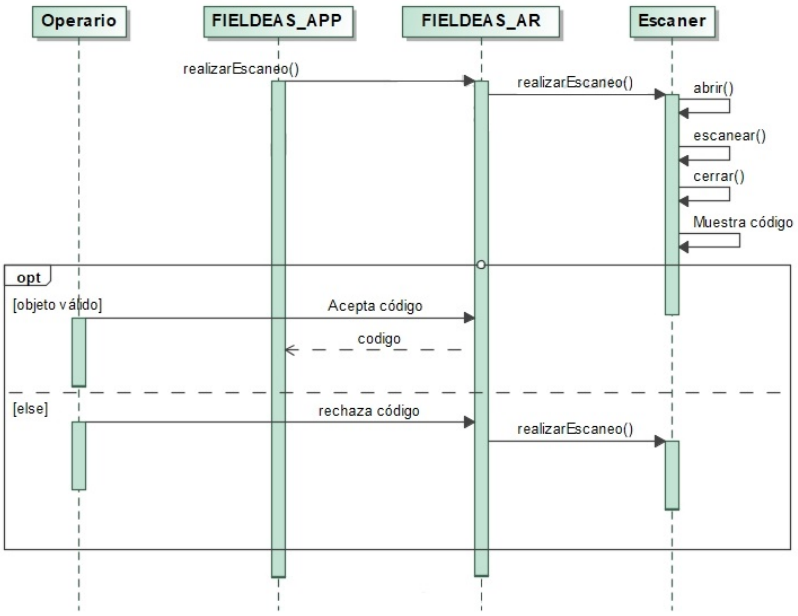

Figura 7: Realizar un escaneo

Es importante decir que no nos hace falta proporcionar el tipo de código a escanear, y que para ello se utilizará el Barcode SDK proporcionado por el fabricante de las SmartGlasses. Además, no es necesario ningún tipo de parámetro en dicha llamada.

Al realizarse la llamada a realizarEscaneo(), se abrirá el escáner del dispositivo AR. Cuando detecte un código realizará un escaneo, y dicho código se mostrará por pantalla al operario que deberá decidir si lo valida o no, en caso negativo se repite el proceso y en caso afirmativo se devuelve el código a la aplicación móvil con la siguiente forma.

```
{
  "error": 0,
  "descripcion": "Código escaneado correctamente",
  "codigo": "5901234123457",
  "tipo": "barras"
}
```

También existe la posibilidad de que el operario cancele la acción, en ese caso, se devolverá un error junto con una descripción del tipo.

```
{
  "error": 1,
  "descripcion": "El usuario ha cancelado el escaneo"
}
```

En el caso del escaneo de imagen, el proceso es igual, a excepción de que deberemos enviar una imagen como parámetro en la llamada, y que la respuesta será de la siguiente forma:

```
{
  "error": 0,
  "descripcion": "Texto escaneado correctamente",
  "texto": "Texto de
            ejemplo con
            saltos"
}
```

Como vemos, escaneoImagen() nos devolverá todo el texto que se reconozca en la imagen, el texto será reconocible si está escrito a máquina o en mayúsculas y con buena caligrafía.

Ahora veremos una situación similar, la búsqueda de un código pasado por parámetro. Su correspondiente diagrama se muestra en la figura 8.

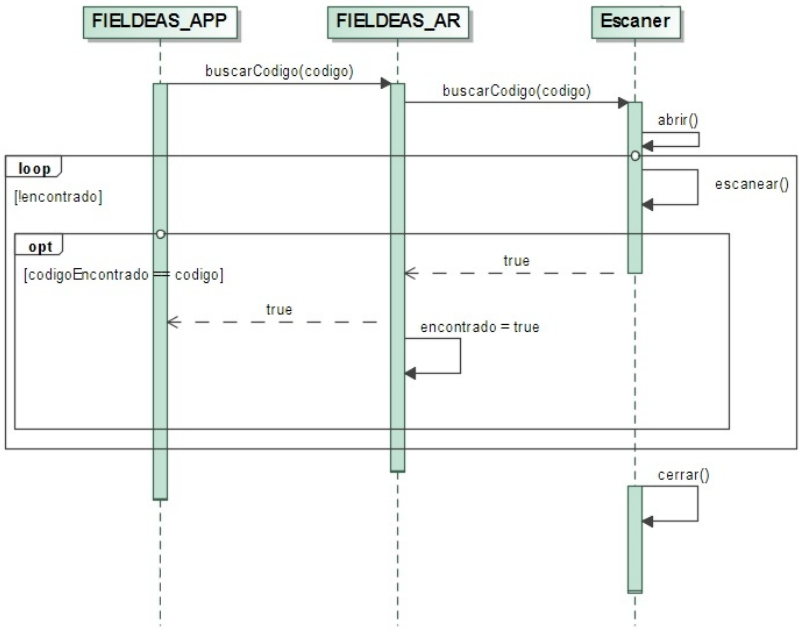

Figura 8: Búsqueda de un código

Como se ve en el diagrama, su funcionamiento es análogo al de realizarEscaneo(), aunque en este caso, se mantiene abierto el escáner hasta que se encuentra el código solicitado, con lo que la respuesta sería la siguiente:

```
{
  "error": 0,
  "descripcion": "Código encontrado"
}
```

También existe la posibilidad de que el operario no encuentre el código y decida cancelar la acción, entonces esto será notificado de la siguiente forma:

```
{
  "error": 1,
  "descripcion": "Código no encontrado"
}
```

Ahora se va a contemplar el ciclo de vida de un elemento de la interfaz de usuario, desde su creación, hasta su eliminación de la interfaz. Se ha creado un elemento genérico. El diagrama 9 muestra la creación y posterior interacción con este elemento genérico.

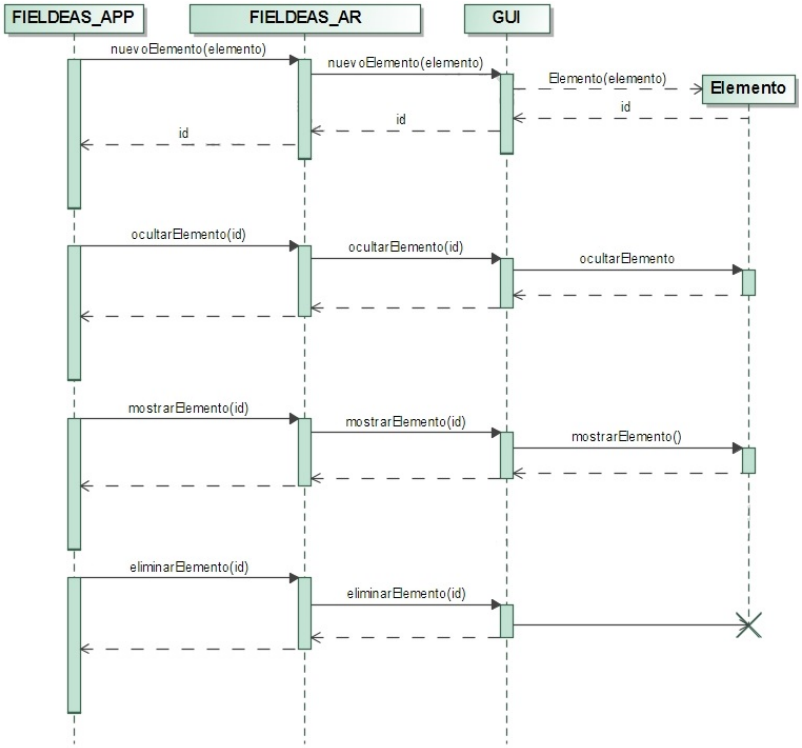

Figura 9: Vida de un elemento de la GUI

La llamada a nuevoElemento(), creará el elemento especificado por los parámetros de la llamada, lo mostrará y retornará el identificador asignado a dicho elemento.

Los parámetros de la llamada son los siguientes:

```
{
  "tipoElemento": 1,
  "posicion"*: [0,0,0],
  "color"*: "#000000",
  "elemento"*1: Elemento,
  "elemento"*2: [
    {
      "id": 0,
      "texto": "Texto"
    }
  ]
}
```

El color y la posición son parámetros opcionales, en el caso de no especificar color, si se ha especificado en el archivo de configuración que se lee en la creación de la conexión, se escoge ese, y en caso contrario el valor por defecto. En cuanto a la posición, si no se especifica se insertará el elemento en la posición [0,0,0] desplazado

hacia la derecha y abajo según corresponda con otros elementos de interfaz.

El parámetro tipoElemento se corresponde con un enumerado: [Texto: 1, Audio: 2, Imagen: 3, Vídeo: 4, Menú: 5]. Según el tipoElemento elegido, en el campo elemento se tendrá que especificar una cosa u otra, para los cuatro primeros tipos (texto, audio, imagen y vídeo), en ese apartado se introducirá directamente, como en $^{*}_{1}$, el contenido del elemento, ya bien sea un base64 o el texto del elemento.

En el caso de los elementos de tipo menú, se deberá especificar de la forma $^{*}_{2}$, en donde se especificará el id de cada entrada que queremos que aparezca, y el texto de esta.

El siguiente proceso a documentar es el de la creación e interacción con un diálogo. El diagrama de secuencia se muestra en la figura 10.

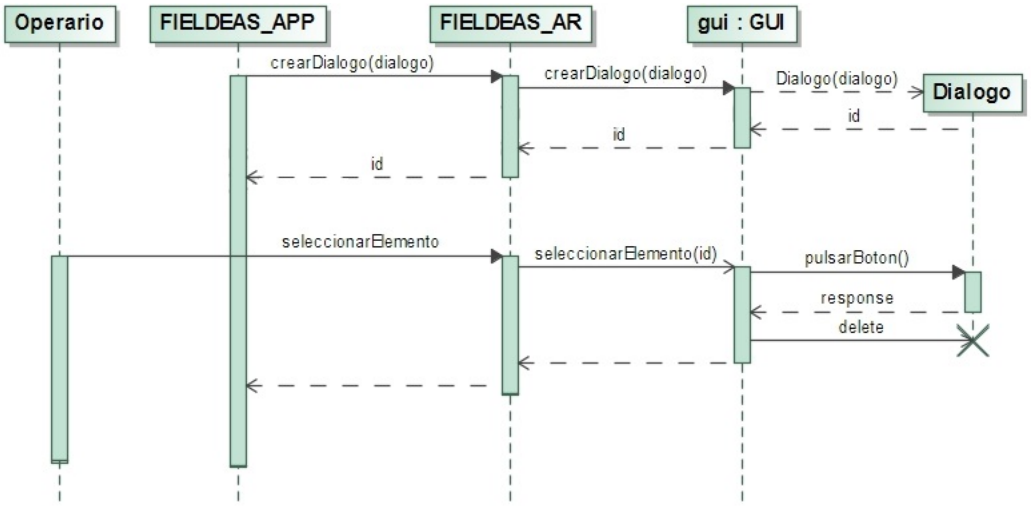

Figura 10: Creación de un diálogo

Como se ve en el diagrama, la llamada a crearDialogo() hace que la GUI invoque al constructor de Dialogo, y despúes retorne el identificador que le ha asignado. Con este identificador, el operario puede seleccionar el diálogo y acto seguido pulsar alguno de los botones que tenga, en el caso en que no se especifiquen botones, se pondrá un botón de "Aceptar" por defecto. Al pulsarse uno de los botones se destruye el diálogo y se devuelve el id de la opción seleccionada de la siguiente forma:

```
{
  "error": 0,
  "id": 1,
  "etiqueta": "Aceptar"
}
```

Los parámetros de crearDialogo() son los siguientes:

```
{
  "titulo": "Titulo del di\'{a}logo",
  "texto": "Texto del di\'{a}logo",
  "botones"*: ["Aceptar", "Rechazar"]
}
```

El identificador del botón que se retorna como respuesta a la llamada es la posición en el array de "botones" comenzando en 0.

5. Diseño e implementación de la interfaz básica

5.1. Arquitectura del sistema

A modo de resumen, la arquitectura y despliegue básicos del sistema completo se muestra en la figura 11.

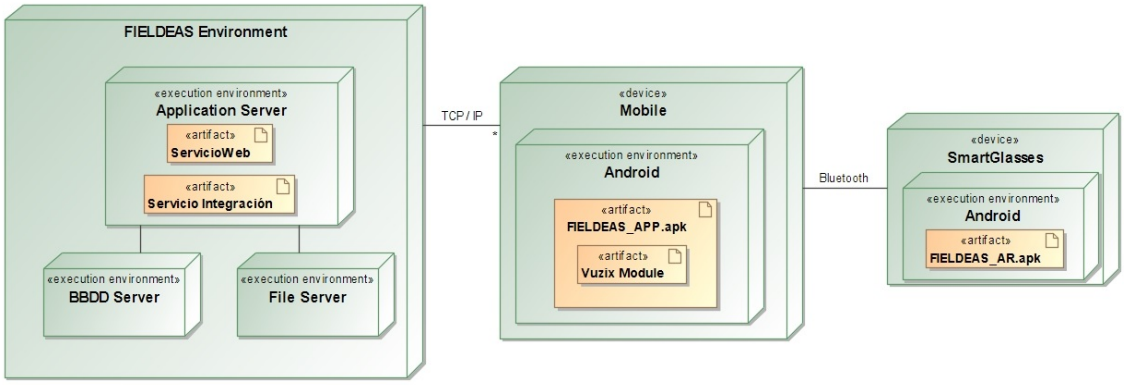

Figura 11: Diagrama de despliegue

En el diagrama podemos ver dos dispositivos, el móvil y las smartGlasses, que son los entornos para los que se ha desarrollado el proyecto. FIELDEAS_AR.apk, que se ha desarrollado completamente durante la realización del proyecto, es la aplicación desplegada en las smartglasses y es la que se encarga de recibir las llamadas realizadas desde la aplicación móvil, procesarlas y realizar lo solicitado.

Por otro lado tenemos FIELDEAS_APP.apk, que es una modificación de la aplicación que se desarrolla en FIELDEAS. Dicha modificación consiste en la adición del módulo "Vuzix Module" desarrollado también como parte de este trabajo, para poder realizar la comunicación entre el dispositivo móvil y las smartGlasses.

Se ha incluido también el entorno FIELDEAS que es con el que se conecta FIELDEAS_APP.apk para obtener los datos del servidor para el dispositivo móvil y también para descargar las aplicaciones web que son ejecutadas dentro de la aplicación móvil.

La plataforma de FIELDEAS establece los requisitos técnicos iniciales con sus recomendaciones mínimas para un uso medio-alto del sistema en:

- Servidor de base de datos
  - Sistema Operativo Microsoft Windows Server 2012, 2012 R2 o superior.
  - Microsoft SQL Server 2012 x64 o superior versión estándar mínimo, no recomiendan usar la express.
  - 16GB de RAM o más.
  - 250GB en particiones.
    - Para el Sistema Operativo 50GB

- Para los temporales 20GB
- Para bases de datos 100GB
- Para registro 30GB
- Para backup 50GB

- 4 Cores o más.
- Un usuario administrador del SQL Server.

- Servidor de aplicaciones
  - Sistema Operativo Microsoft Windows Server 2012, 2012 R2 o superior.
  - 8GB de RAM o más.
  - 4 Cores o más.
  - 250GB en particiones.
    - Para el Sistema Operativo 50GB
    - Resto para aplicaciones
  - Programas y características de Windows Server.

En el caso del servidor de ficheros, no se especifica nada excepto que debe tener un uso compartido y disponer de acceso a este desde un usuario Windows de FIELDEAS.

5.2. Diseño de la interfaz de programación de aplicaciones

El diseño de la interfaz se ha realizado en base al modelo mostrado en la figura 12.

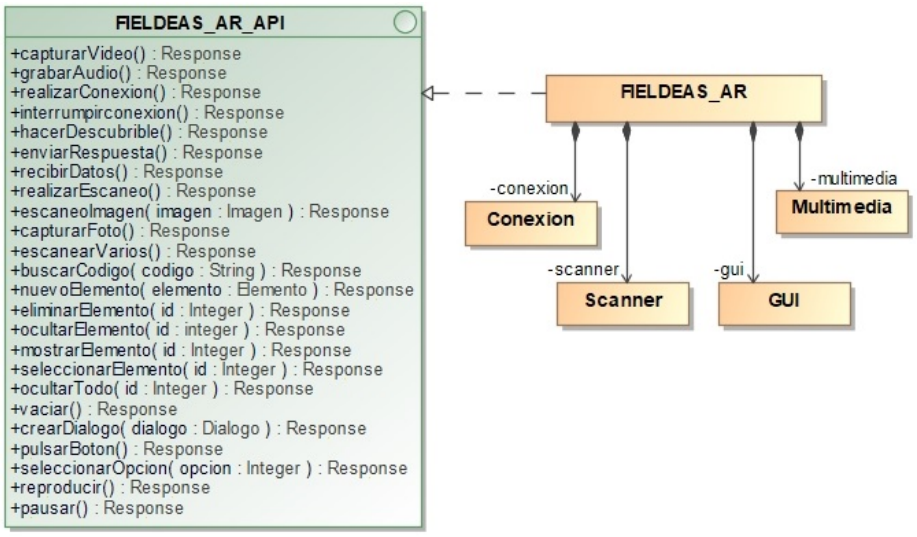

Figura 12: Diseño de la interfaz de aplicaciones

Como se observa en la figura, todos los metodos retornan Response, que es un objeto codificado como JSON que conforma la respuesta de la ejecución. En la mayoría de los casos dicha respuesta se compone solamente de un código de error y una descripción.

Los cuatro grandes grupos de funcionalidades que ofrece la API son implementados a través de las clases Multimedia, GUI, Escaner y Conexion, cada una de las cuales son descritas a continuación.

Por brevedad se han ocultado getters y setters de los diagramas. En el diagrama de la figura 13 se muestra la clase Multimedia.

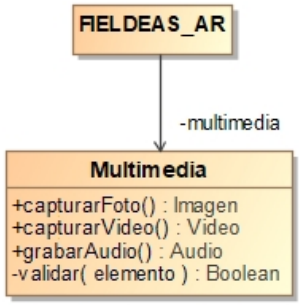

Figura 13: Módulo de multimedia

Esta clase será la encargada de realizar las interacciones con las entradas que proporciona el dispositivo, la captura de vídeo e imagen y la grabación de audio. Así como el proceso de validación de las capturas realizadas, en las que se preguntará al operario antes de enviar el elemento recogido hacia la aplicación de FIELDEAS.

A continuación pasamos a explicar la clase Conexion, cuya estructura se muestra en la Figura 14. A través de la clase Conexion y sus dos enumerados se controlan todos los enlaces entre la aplicación de FIELDEAS correspondiente en ese momento, y la instalada en el dispositivo AR.

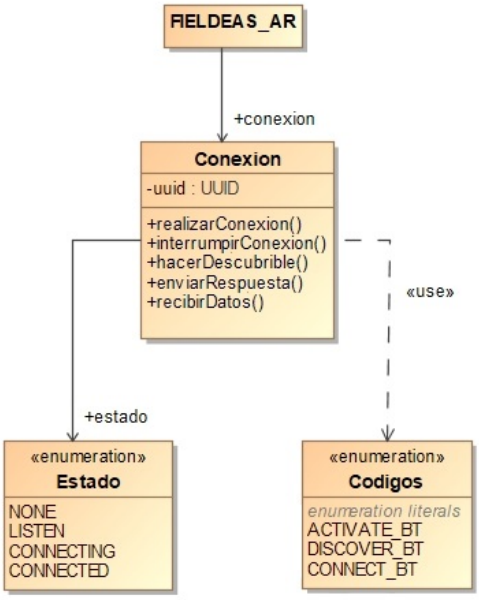

Figura 14: Módulo de conexión

El método hacerDescubrible() se ejecuta al iniciar la aplicación y siempre que el operario lo solicite y mientras no esté la conexión realizada. Los métodos realizarConexion() e interrumpirConexion() son ambos autodescriptivos, recibirDatos() será el que ejecute los demás métodos en función de lo que solicite el programador de la aplicación móvil. Y enviarRespuesta() será el que envíe los datos de vuelta a la aplicación de FIELDEAS.

Continuamos describiendo el tercer gran módulo, el escáner de códigos, tanto QR como de barras, cuya estructura se muestra en la figura 15.

Como solo permitimos un escáner simultaneo, el atributo escaneando controla esto. El proceso de realizarEscaneo(), escaneoImagen() y buscarCodigo() ya se han descrito en la sección de especificación de requisitos. Y los métodos abrir(), cerrar() y escanear() son usados por los otros métodos.

El método realizarFoto() es usado cuando se realiza un escaneo en busca de caracteres con el ObjectScanner, ya que no será posible en tiempo real.

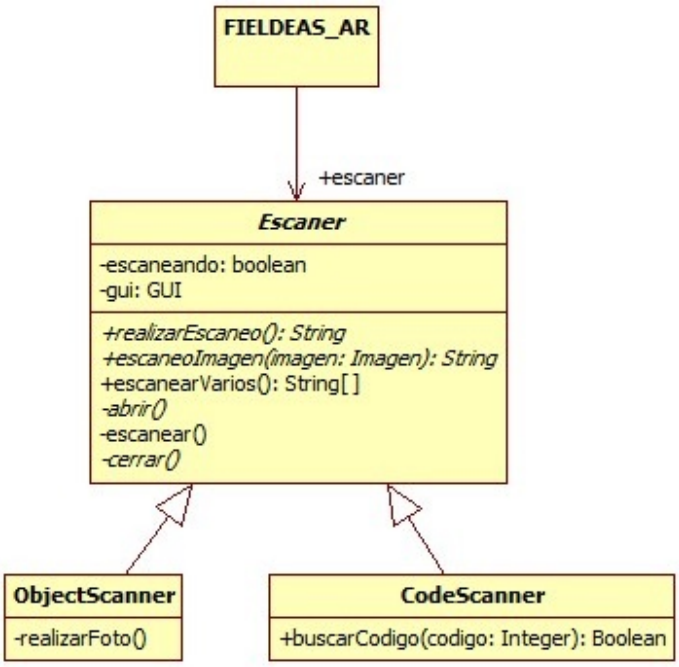

Figura 15: Módulo de escáner

Finalmente el módulo más complejo, el de la interfaz de usuario, se muestra en la figura 16.

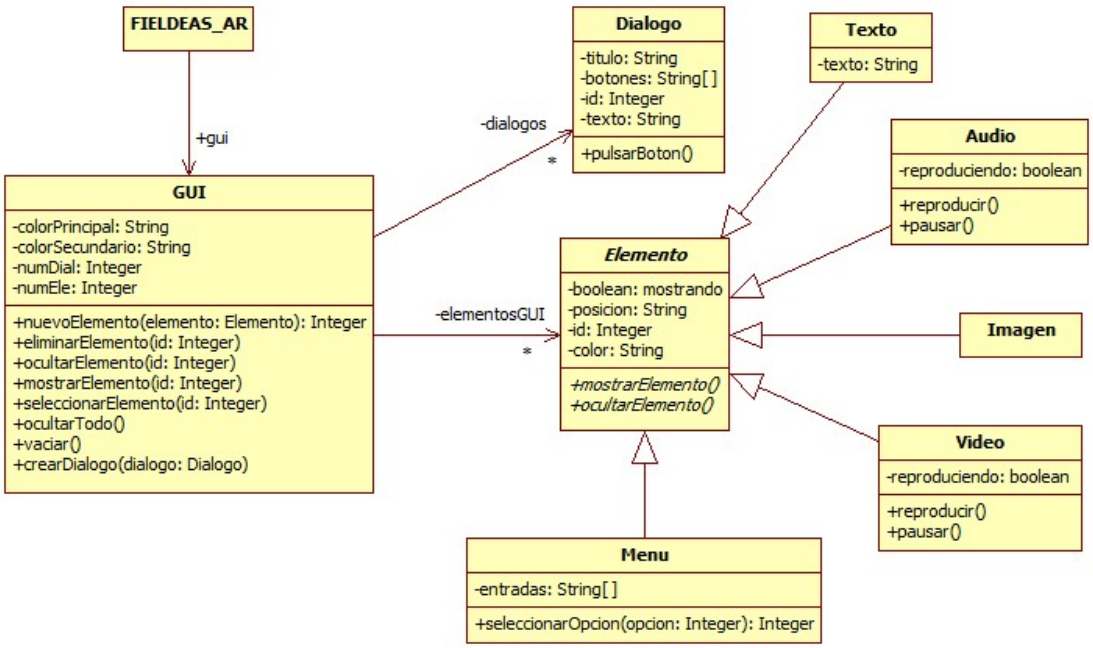

Figura 16: Módulo de la interfaz gráfica

Comenzando por la clase principal, GUI, vemos que hay varios atributos como son el colorPrincipal y el colorSecundario, que se inicializan durante la realización de la conexión.

Los otros dos atributos son para tener un recuento del número de diálogos y elementos que están en la UI, tanto mostrados como ocultos. Con esto podremos recorrer mejor los elementos al tener bien definidos los límites de cada categoría. Los métodos nuevoElemento(), eliminarElemento(), mostrarElemento() y ocultarElemento() ya han sido descritos durante la fase de captura de requisitos, y son los métodos básicos para la creación y modificación de elementos de interfaz.

El método vaciar(), como indica su nombre, reinicia la interfaz a su estado inicial. Mientras que ocultarTodo(), no elimina los elementos si no que les oculta de la vista del operario. Y finalmente, análogamente a nuevoElemento(), está nuevoDialogo() que tiene un comportamiento similar.

Pasamos a definir la clase Dialogo, en la que tenemos varios atributos que componen la interfaz del diálogo que aparecerá por pantalla, entre los que están, el título del mensaje y su texto y las etiquetas de los botones si los hubiera. Y finalmente el id autoasignado por GUI. Solo tiene un método, pulsarBoton(), que indicará en la respuesta de la creación del diálogo el índice del botón pulsado.

Ahora tenemos los elementos, todos ellos derivan de la clase Elemento, dicha clase contiene el atributo que guarda su posición, si se está mostrando el elemento o no, y el identificador de dicho elemento. Todos los elementos tienen las operaciones básicas de mostrar y ocultarse, y cada tipo de elemento propio tiene unas propiedades diferentes.

El elemento más básico, Texto, solo tiene un atributo adicional, el valor del texto que se muestra. Para los de tipo Audio y Video, como tienen una duración, también pueden ser pausados y continuar su reproducción, para ello se utilizan los dos métodos reproducir() y pausar(), que son comunes a los dos tipos y ambos métodos modifican su atributo reproduciendo. Las imágenes no tienen métodos ni atributos adicionales, se contempló la posibilidad de rotar la imagen, pero se desestimó por no ser tan aplicable.

En el caso de los elementos de tipo Menú, tenemos un array de Strings en el que están las entradas del menú, así como también disponemos del método seleccionarOpcion() que nos permite seleccionar una opción para indicárselo a la aplicación de FIELDEAS.

5.3. Implementación de la solución

Durante la realización del proyecto se encontraron diversos problemas, la mayoría de estos fueron durante la realización del modulo de conexión.

Al principio se siguió un tutorial básico proporcionado por Android [17] en el que se realizaba un chat a través de Bluetooth, el primer problema que nos encontramos fue el de adaptar dicho chat a nuestras necesidades para la comunicación entre el dispositivo móvil y las smartGlasses.

En principio el dispositivo móvil hacia de cliente y las smartGlasses de servidor, donde el servidor respondía cuando recibía correctamente una petición, pero esta relación cliente-servidor no era viable para el proyecto ya que los procesos realizados en las smartGlasses en su mayoría son asíncronos y hacia falta conocer en el dispositivo móvil cuando terminaba dichos procesos y en muchos casos su resultado, así que ambos dispositivos se convirtieron en cliente-servidor con la consecuente modificación que eso supuso en el proceso de conexión y comunicación.

Cuando se fueron a implementar funciones como la de capturar una foto en las smartGlasses y devolverla al dispositivo móvil, o viceversa, enviar una foto o vídeo desde el dispositivo móvil hacia las smartGlasses nos encontramos con varias dificultades. La primera, al cargar las imágenes en memoria se superaba el máximo tamaño permitido para una aplicación, con lo que se tuvo que realizar un proceso de reescalado, que no conllevó una gran perdida de calidad, ya que la resolución de las smartGlasses es mucho menor a la de los dispositivos móviles actuales.

Como consecuencia de lo anterior, el envío de archivos pesados superaba en la mayoría de los casos el tamaño máximo permitido en Bluetooth y el del buffer implementado en nuestro "protocolo". Cuando detectamos este problema se propuso dejar de usar Bluetooth para empezar a utilizar el Wi-Fi, pero esta idea se rechazó para no tener que modificar todos los procesos que intervenían en la comunicación.

Una vez desestimado el cambio, se trabajó en separar los envíos en "paquetes"

más pequeños, y a raíz de esto también había que conseguir juntar dichos paquetes en su recepción, e identificar cuando una transmisión había concluido. Para esto se añadieron unos bits de control fijos al final de cada transmisión indicando el cambio, dichos bits eran una secuencia de unos y ceros aleatoria con 50 dígitos para que fuera prácticamente imposible que una secuencia transmitida coincidiera con dichos bits de control.

Al aumentar el número de envíos se encontraron fallos de transmisiones, que se tuvieron que corregir guardando siempre el último paquete enviado para reenviarle en caso de pérdida.

También se encontraron problemas con otras funciones implementadas aparte de la comunicación, pero se solventaron gracias a la documentación y a los compañeros de FIELDEAS que trabajan con el lenguaje Android.

Los problemas de comunicación fueron más complicados de resolver ya que el Bluetooth no es una tecnología muy extendida en el intercambio de mensajes, y las implementaciones que lo utilizan como los dispositivos de sonido inalámbrico u otros dispositivos tienen frameworks de desarrollo que abstraen dicho trabajo.

5.4. Diseño e implementación de funciones avanzadas

Como se indicó en la sección de objetivos se han implementado dos funciones más complejas que permiten realizar procesos completos.

La primera, una función con la que combinando el escaneo masivo de códigos de barras o QR y la visualización de imágenes en las smartGlasses hemos conseguido realizar un proceso de recogida de paquetes, o "picking" como lo llaman las empresas de transporte, que servirá a operarios de los almacenes de empresas de transporte para empaquetar los diferentes objetos de una carga de forma sencilla y guiada.

Para la realización de esta función se modeló primero un *wireframe* para visualizar que es lo que se tenía que implementar.

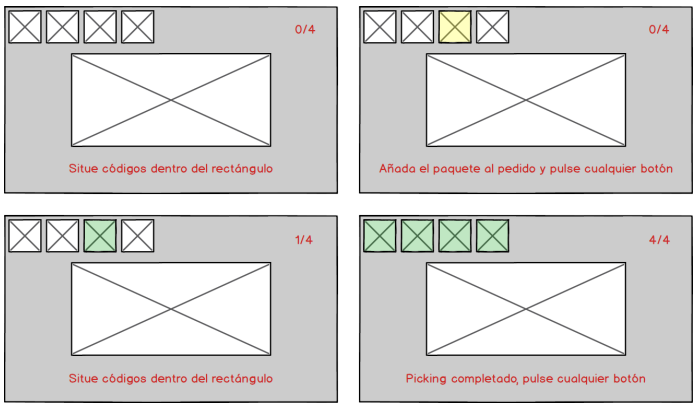

Figura 17: Wireframe de picking

A consecuencia de esto, se añade un nuevo método a la API, realizarPicking(), al que se le pasa como parámetro una lista de objetos, con una imagen y un código de barras asociado y guía al usuario en todo el proceso.

La segunda función implementada, también relacionada con los códigos de barras o QR, fue la realización de un inventariado. A modo de cajero de tienda se registran los objetos uno a uno, con la posibilidad de registrar varias veces el mismo en el caso en el que haya varios objetos del mismo tipo y sin la necesidad de escanear uno a uno, simplemente pulsando tantas veces el botón de las smartGlasses como sea necesario.

Al igual que con realizarPicking(), la función realizarInventario() también se añade a la API, en este caso no tiene parámetros y devuelve la lista de códigos de barras escaneados.

Ambas funciones fueron desarrolladas para poder tener una demostración de procesos más complejos y útiles dentro del entorno industrial. Se escogieron esos dos procesos porque se identificaron como los más útiles en el entorno de las empresas de transporte a las que da servicio FIELDEAS, y porque en dichos procesos se observa claramente el potencial de tener libres las manos para realizar tareas, ya bien sea el picking o el inventariado.

5.5. Pruebas

Para probar las diversas funciones se realizó una aplicación que contenía solamente un listado con botones que invocaban las diferentes funciones implementadas, como se muestra en la figura 18. Dicha aplicación estaba realizada en Java por lo que no se realizaba el proceso de convertir los parámetros desde JSON a objetos Java, solamente se invocaban los métodos directamente.

Figura 18: Capturas de pantalla de la aplicación de prueba

Al terminar el proyecto, una vez integrado dentro de la plataforma de FIELDEAS, se creó una aplicación web con la que se realizó una prueba de concepto. Dicha aplicación usa los métodos finales de la interfaz desarrollada ya que está realizada para que sea utilizada por programadores JavaScript. Algunas capturas de pantalla de la aplicación se incluyen en la figura 19.

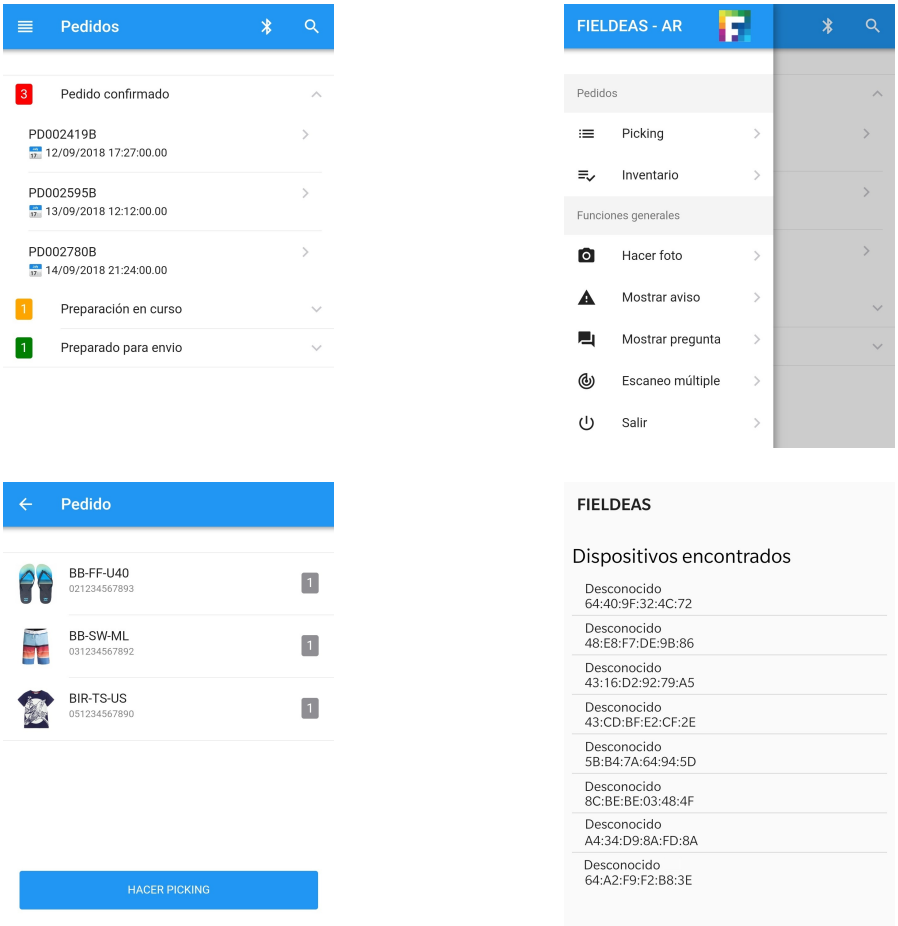

Figura 19: Capturas de pantalla de la aplicación de prueba de concepto

La pantalla de inicio de la aplicación de prueba es la de la esquina superior izquierda de la figura, en ella encontramos una lista de pedidos separados por el estado en el que se encuentren. A la derecha arriba vemos el menú lateral abierto en el que encontramos las dos funciones avanzadas, picking e inventario y debajo acciones de ejemplo más comunes. Al pulsar sobre uno de los pedidos de la pantalla de inicio se abre la pantalla inferior izquierda de la figura, en ella salen los productos del pedido y un botón para lanzar el picking. En la pantalla de abajo a la derecha se muestra la lista de dispositivos Bluetooth disponibles para la conexión.

No se estableció ningún plan de pruebas muy exhaustivo para la interfaz y en algunos momentos esa falta de pruebas complicó el desarrollo, que se podía haber solventado teniendo unas pruebas bien definidas.

6. Líneas futuras y conclusiones

Como se introdujo al principio del trabajo, la realidad aumentada es una tecnología por explotar, ahora mismo se encuentra en sus inicios, y como tal, las expectativas creadas son bastante altas.

En lo que se refiere al proyecto en sí, las empresas de mensajería buscan innovar para mantenerse por delante de sus competidores, y la introducción de la realidad aumentada en el sector está provocando una diferenciación entre las que se están adaptando a esta nueva visión y las que no.

El proyecto en su estado actual cubre las necesidades de guiado desatendido y tiene capacidades para la realización de tareas como el picking o el inventariado. Pero para que fuera completo tendría que añadirse la posibilidad de realizar una asistencia técnica remota.

La realización del trabajo de fin de grado en una empresa ha sido una experiencia que me ha dado la posibilidad de ver como funcionan las empresas de desarrollo de software. Si bien es cierto que durante el grado ya había realizado practicas en empresa, la realización de mi propio proyecto me ha descubierto los problemas a los que se enfrentan los trabajadores, desde los problemas técnicos que imposibilitan la continuación del desarrollo, como las entregas mal fijadas que dan lugar a software desarrollado de una forma rápida y no tan probado como se debería.

En general, la realización del TFG ha sido una experiencia bastante buena, y recomendaría a todos los estudiantes de Ingeniería Informática que realicen al menos un periodo de prácticas antes de terminar su formación. Como en la mayoría de las profesiones no se asemejan los entornos de trabajo durante los estudios y el mundo laboral. La experiencia que proporcionan unas practicas es esencial para tener una visión de la utilidad de lo estudiado.

Referencias

[1] FIELDEAS, "La Industria 4.0 liderando el futuro", 08-03-2018, https://www.fieldeas.com/la-industria-4-0-liderando-futuro

[2] Digi-Capital, "Augmented Reality report and foresights", 2018, https://www.digi-capital.com/news/2018/01/ubiquitous-90-billion-ar-to-dominate-focused-15-billion-vr-by-2022

[3] FIELDEAS, "FIELDEAS", 2019, https://www.fieldeas.com

[4] Ministerio de Administraciones Públicas, "Métrica v.3", 2018, https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html

[5] Atlassian, Dan Radigan, "Kanban", 2018, https://www.atlassian.com/agile/kanban

[6] Atlassian, "Jira", 2018, https://www.atlassian.com/software/jira

[7] Google, "Android Studio", 2019, https://developer.android.com/studio

[8] Google, "Android", 2018, https://developer.android.com

[9] Sublime, "Sublime Text", 2019, https://www.sublimetext.com

[10] w3schools, "JavaScript", 2019, https://www.w3schools.com/jS/default.asp

[11] Evan You, "Vue", 2018, https://vuejs.org

[12] Vladimir Kharlampidi, "Framework7", 2019, https://framework7.io/vue

[13] Vuzix, "Modelo m100", 2018, https://www.vuzix.eu/Products/m100-smart-glasses

[14] Epson, "Modelo Moverio-BT-350", 2018, https://epson.com/For-Work/Wearables/Smart-Glasses/Moverio-BT-350-Smart-Glasses/p/V11H837020

[15] Sony, "Modelo SED-E1", 2018, https://developer.sony.com/develop/smarteyeglass-sed-e1

[16] Vuzix, "Modelo m300", 2018, https://www.vuzix.eu/Products/m300-smart-glasses

[17] Developers, "Bluetooth", 08-10-2018, https://developer.android.com/guide/topics/connectivity/bluetooth