Aplicación móvil para la configuración de variadores

Víctor Siles Martínez
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Consultor responsable
Roger Montserrat Ribes

07/06/2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	Aplicación móvil para la configuración de variadores
Nombre del autor:	Víctor Siles Martínez
Nombre del consultor:	Roger Montserrat Ribes
Fecha de entrega (mm/aaaa):	06/2017
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*

Resumen del Trabajo (máximo 250 palabras):

Los variadores de frecuencia son un sistema para el control de la velocidad rotacional de un motor de corriente alterna por medio del control de la frecuencia de alimentación suministrada al motor.

Su uso puede realizarse desde líneas de producción de fábricas a ascensores. Para la configuración es necesario disponer de un PC con un software específico (distribuido por la misma compañía que el variador) o realizarlo desde la pantalla del variador.

Para agilizar el proceso de esta configuración, se utilizará una aplicación móvil que sea capaz de comunicarse vía Bluetooth con el variador. Para ello es necesario:

- Desarrollar una aplicación móvil capaz de conectarse a un variador e intercambiar datos.
- Instalar en el variador y actualizar su firmware para ser capaz de comunicar vía Bluetooth.

Este proyecto se centrará en el diseño y desarrollo de la aplicación.

Abstract (in English, 250 words or less):

Frequency inverters are a system for controlling the rotational speed of an AC motor by controlling the supply frequency supplied to the motor.

They can be used from factories (Conveyor belts) to elevators. For the configuration of these inverters, it is necessary to have a PC with a specific software (distributed by the same company that the inverter) or from the screen of the inverter.

To improve the process of this configuration, a mobile application that is able of communicating via Bluetooth with the inverter will be used. For this matter, it is necessary to:

- Develop a mobile application capable of connecting to a drive and exchanging data.
- Install in the drive and update its firmware to be able to communicate via Bluetooth.

This project will focus the design and development of the application.

Palabras clave (entre 4 y 8):
Variador de frecuencia, Operador Digital, Smartphone, App.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

En la automatización industrial, se conocen los variadores como:
"Un sistema para el control de la velocidad rotacional de un motor de corriente alterna (AC) por medio del control de la frecuencia de alimentación suministrada al motor". [1]

Este dispositivo electrónico contiene unos objetos (también llamados parámetros), los cuales configuran o informan sobre el estado y funcionamiento actual. Para realizar esta operación, un operario o ingeniero debe hacer una de las siguientes acciones:

- Conectarse desde un PC (cable entre PC y variador o a través de una red LAN u otro protocolo) con un software determinado. [2]
- Hacer uso de un Operador Digital o KeyPad conectándose al variador. [3]
- Hacer uso del operador integrado en la pantalla.

A partir de aquí, imaginemos el siguiente caso:

Un aeropuerto utiliza variadores para el motor que mueve la cinta mecánica donde se realiza la facturación de las maletas. Uno de ellos falla, y da la alerta. El operario, con las dos opciones antes expuestas debe:

- O bien hacer uso del PC, y cablear el PC con el variador.
- O bien hacer uso del operador digital conectándolo al variador.

Cabe destacar que normalmente estos variadores están cubiertos o protegidos en paneles con puerta o incluso cercano al mecanismo de la cinta.

Con el desarrollo de esta aplicación se quiere conseguir que:

- El operario no dependa de un PC y/o cableado.
- Obtenga, de una forma más cómoda y rápida, información que respecto al uso de un operador digital.

Otro valor añadido a la aplicación es que el usuario final no tiene que desembolsar un gasto en el operador digital, el cual se vende por separado al variador.

Actualmente, en el campo de la automatización industrial, como en casi cualquier otro ámbito de la industria, el mundo de los smartphones se va abriendo paso. Tanto es así, que muchas empresas las cuales no están apostando fuerte, están perdiendo fidelización de sus clientes. Gran parte de esta aplicación pretende que los actuales clientes no cambien de proveedor y a la vez, conseguir clientes nuevos.

En el mercado se puede encontrar aplicaciones que permiten:

- Saber cuántos componentes de variadores de frecuencia son necesarios para una determinada línea. [4]
- Configuración y estado de un controlador. [5]

Ninguna de ellas cubre el funcionamiento de la aplicación a desarrollar.

1.2 Objetivos del Trabajo

El objetivo de este trabajo final de master debe cumplir lo anteriormente planteado.

La idea de la empresa que vende y gestiona los variadores es desarrollar la aplicación a la vez que el equipo de hardware añade el equipamiento necesario para que el variador sea capaz de comunicarse con el móvil.

La realización de este proyecto estará centrada en la aplicación móvil, dejando la parte de hardware del variador aparte. Para ello, se desarrollará una aplicación de PC que simulará un variador y su conectividad

Aplicación Móvil:

- Mostrar la información de los variadores (Objetos, descripción, ayuda…)
- Acceder a un servidor donde se dispone de la información de los diferentes variadores.
- Conectarse de manera inalámbrica al variador.
- Leer los objetos del variador y mostrar en la aplicación los datos actuales.
- Escribir los objetos desde el móvil y que sean efectivos en el variador.
- Copiar datos de un variador en memoria para poder ser “pegados” en otro dispositivo.

Simulador de variador (Aplicación PC):

- Conectividad Bluetooth
- Responder a los comandos de lectura y escritura.

1.3 Enfoque y método seguido

La aplicación facilitará al operario a configurar el variador, el cual, como se mencionó antes, es el target de la empresa. Debido a esto, cubrir el máximo de rango posible de usuarios hará que la aplicación tenga más buena acogida, ya que no se obligará al usuario final que tenga un dispositivo en concreto a parte del variador. La mejor opción en este caso es realizar una aplicación **Multiplataforma**.

Hay diferentes maneras de desarrollar una aplicación multiplataforma con diferentes herramientas: Web, Phonegap, Titanium…
Debido al conocimiento de .NET y al reto que supone aprender una tecnología nueva no realizada durante el máster, la aplicación se desarrollará haciendo uso de Xamarin. (iOS, Android, Windows Universal Apps).

El desarrollo de esta aplicación debería ir desarrollado en paralelo por el de un hardware. El objetivo de este trabajo no es éste, pero es necesario para el funcionamiento de la aplicación. Para ello, se realizará una aplicación .NET donde simule el funcionamiento y las comunicaciones del variador.

En cuanto a la tecnología inalámbrica se ha decidido hacer uso e implementar la comunicación de Bluetooth debido a:

- No todos los variadores están conectados a la red.
- NFC tiene un rango demasiado corto para el uso que se le quiere dar a la aplicación.

1.4 Planificación del Trabajo

A continuación, se detallará la planificación para la realización del proyecto. Para ello, cabe tener en cuenta los distintos datos:

- Dedicación diaria: 1 hora día laborable, 5 horas días festivos.
- Dedicación semanal: 15 horas.

- PEC 2: Fecha límite **05/04/17** (Diseño)
- PEC 3: Fecha límite **17/05/17** (Implementación)
- Entrega final **07/06/17**

La metodología a seguir será una metodología ágil (SCRUM), así el desarrollo será incremental. Los roles de SCRUM Master, desarollador y tester, será la misma persona. El product owner será el tutor del máster.

En la metodología SCRUM, se realiza el trabajo en sprints, donde al finalizar cada uno de ellos el equipo debe demostrar al product owner el trabajo realizado.

Para agilizar el desarrollo y la comunicación del SCRUM master, se realizarán sprints de dos semanas, contando a partir del día 15/03. Dado que la dedicación máxima de un sprint será de 18 horas, cualquier tarea que su duración sea superior, habría que dividirla en otras menores. En caso de que el esfuerzo sobrepase el tiempo total, esa semana se asume un esfuerzo extra.

A continuación, se muestran las tareas a realizar y el tiempo de dedicación requerido. Más adelante, se definirá su distribución en sprints:

Fase de diseño:

- Usuarios y contexto de uso: 5 horas
- Diseño conceptual: 5 horas.
- Prototipado: 15 horas.
- Evaluación: 21 horas.
    - Casos de uso: 7 horas.
    - Diseño de la arquitectura: 14 horas.

Fase de implementación (Esta fase puede diferir dependiendo del resultado del diseño). En esta fase, el desarrollo de cada tarea es, además del desarrollo, también el testeo (Tanto unitario como funcional).

- Xamarin Learning (Realización de tutoriales): 10 horas
- Creación de los proyectos y vista con fake data: 12 horas.
- Creación del modelo de datos del variador: 6 horas.
- Consumir datos y sustituir fake data: 10 horas.

- Creación de aplicación en PC para comunicación Bluetooth con funcionamiento para lectura 10 horas.
- Añadir lectura desde aplicación móvil (Uso de Bluetooth desde el móvil) 20 horas
- Añadir escritura. 15 horas
- Reconocimiento de modelo. 15 horas
- Más funcionalidad…

A continuación, se desglosan los sprints:

Sprint 1 (15/03 – 29/03) Total: 30 horas:

- Usuarios y conecto de uso (5 horas)
- Diseño conceptual (5 horas)
- Prototipado (15 horas)
- Casos de uso (5 horas)

  Entrega: Usuarios y conecto de uso, Diseño conceptual y prototipado.

Sprint 2 (29/03 – 05/04) (Más corto debido a entrega de PEC1, así que más dedicación) Total: 20 horas.

- Casos de uso (2 horas)
- Diseño de arquitectura (14 horas)
- Redacción documento (4 horas)
- **Entrega**: PEC1 (Sprint 1 y Sprint 2)

Sprint 3 (05/04 – 19/04) Total: 30 horas:

- Xamaring Learning (0 horas)
- Creación de los proyectos y vista con fake data (12 horas)
- Creación de modelo de datos (reales) del variador (6 horas)
- Actualización memoria (2 horas)

  Entrega: Aplicación con vista fake data. Datos y memoria actualizada

Sprint 4 (19/04 – 03/05) Total: 30 horas:

- Aplicación de PC con Bluetooth (10 horas)
- Añadir lectura desde el móvil (20 horas)
- **Entrega**: Aplicación PC con lectura y desarrollo lectura desde aplicación móvil.

Sprint 5 (03/05 – 17/05) Total: 30 horas

- Añadir lectura (15 horas)
- Reconocimiento de modelo (15 horas)
- **Entrega**: Soporte a lectura y reconocimiento de modelo.

Sprint 6 (03/05 – 17/05) Total: 30 horas

- Añadir lectura (15 horas)
- Reconocimiento de modelo (15 horas)
- **Entrega**: PEC 2 (Soporte a lectura y reconocimiento de modelo y previos Sprints)

Sprint 7 (17/05 – 31/05) Total: 30 horas.

- Completar memoria, presentación, videos, etc... (15 horas)
- **Entrega**: Entrega Final.

1.5 Breve sumario de productos obtenidos

Al final del proyecto se entregará:

- Código fuente de aplicación móvil
- Aplicación móvil
- Simulador de variador (Aplicación PC)
- Memoria
- Presentación
- Vídeo demostrativo
- Manual

1.6 Breve descripción de los otros capítulos de la memoria

La memoria se ha dividido en capítulos para explicar el proceso de diseño y creación de la aplicación, así como el cumplimiento de la planificación propuesta. Esta es la organización de los capítulos:

- Diseño: Capítulo con toma de decisiones para diseñar y coger forma de la aplicación. En ella, se detalla los contextos de uso, sus usuarios potenciales, la lista de funcionalidades, el árbol de navegación, prototipado y arquitectura.
- Remotinv: Capítulo centrado en el aspecto final de la aplicación.

- Simulador de Variador: Se muestra en que consiste la aplicación desarrollada para simular un variador.
- Pruebas y verificación: Capítulo para mostrar las pruebas realizadas a la aplicación y las correcciones a partir de éstas.
- Procedimiento y resultado: Resumen del cumplimiento de la planificación y el resultado de la aplicación.
- Conclusiones: Centrado en explicar lo que se ha aprendido durante el proyecto, así como las dificultades y como han sido superadas.
- Glosario: Conjunto de nomenclatura usada durante el proyecto.
- Bibliografía: Lista con toda la bibliografía usada durante el proyecto.
- Anexos: Manual de usuario de la aplicación para la configuración de variadores y el simulador.

2. Diseño

2.1 Usuarios y contexto de uso

La aplicación, al ser atada a un producto (variador), es fácilmente saber quiénes serán sus usuarios finales.

Se han hablado con algunos usuarios finales para ver sus necesidades, como ofrecerles una solución o incluso mejorando su actual uso de los variadores.

Características de los usuarios:

- Personas adultas.
- Operarios o Ingenieros de aplicación.
- Conocimiento sobre variadores.
- Hacen uso de un operador digital o PC para la configuración de los variadores.

Una vez hecha la fase de análisis, se muestran fichas de usuarios que fueron entrevistados y un escenario de uso para cada uno de ellos.

Usuario 1

Nombre: *Roberto*
Edad: *47 años*
Profesión: *Operario en un aeropuerto*

Roberto siempre actúa de la misma manera cada vez que en el recibe un aviso de que una cinta va mal. Para ello, coge su portátil, y se dirige al variador. Algunos de ellos están tapados o no son accesibles, así que debe parar alguna maquinaria, conectar el ordenador al variador, diagnosticar el problema y solucionarlo. Una vez acabado, debe dejar todo como estaba.

Escenario

Roberto recibe una llamada por su Walkie de que algo va mal en una cinta que se encarga de mover el equipaje por el aeropuerto. Para solucionarlo, se dirige al lugar donde se sitúa el variador. Una vez allí, sincronizando su móvil con el variador por Bluetooth, diagnostica el problema y lo soluciona, evitando parar otras líneas y sin tener que llevar el portátil encima.

Usuario 2

Nombre: *José*
Edad: *31 años*
Profesión: *Operario en una fábrica*

José lleva poco tiempo trabajando en la fábrica. En ella, se empaquetan productos medicinales. La línea donde se ayuda a cerrar las cajas, cambia cada cierto tiempo en el tamaño de las cajas. Para configurar el variador, José dispone de un operador digital para configurarlo y adecuarlo al nuevo empaquetamiento. Al no conocer suficiente el variador y la poca información que ofrece el operador digital, debe acercarse con el manual para saber qué hace cada parámetro y que valores puede guardar.

Escenario

José debe cambiar la configuración de un variador para adaptarlo al nuevo producto a empaquetar. Para ello, usando la aplicación móvil y sincronizándose al variador, dispone de la información de cada objeto, con ayuda de su uso y el rango de valores que acepta.

Usuario 3

Nombre: *Carlos*
Edad: *40 años*
Profesión: *Ingeniero de aplicaciones*

Carlos es ingeniero de aplicación que trabaja para la empresa que está desarrollando la aplicación móvil. Él se encarga de visitar a los clientes y ayudarles a poner en marcha el sistema. Para ello, suele hacer uso de un operador digital. Una vez configurado uno, hacia una copia y aplica esa copia uno por uno, conectando el operador digital.

Escenario

Carlos, utilizando el propio móvil de empresa, es capaz de acercarse al cliente sin operador digital. Solamente utilizando su aplicación, puede configurar el sistema del cliente a partir de un variador, copiar su configuración, y copiarla a los demás.

A partir de estas fichas de usuario, se puede definir también el contexto de uso de la aplicación:

- Uso de la aplicación exclusiva en el trabajo.
- Normalmente, solo una persona mirará la pantalla.
- Se utilizará en localizaciones con luz artificial.
- Disponibilidad de wifi en la mayoría de los casos.

2.2 Lista de funciones

Una vez vistos todos los usuarios potenciales y sus escenarios de uso, podemos definir una lista de funciones que compondrán la aplicación. Para ello, se nombrarán y se hará una breve descripción haciendo especial énfasis en su finalidad.

Lista de funcionalidades:

- **Detección de modelo**: La aplicación debe ser capaz de detectar el modelo a través de Bluetooth y descargar, desde el catalogo del servidor, los datos correspondientes con el variador: Parámetros, alarmas, etc…
- **Visualización de parámetros**: Poder ver la lista de parámetros, con su información: Nombre, descripción, ayuda, rango y valor actual (Leyendo el valor del variador).
- **Edición de parámetros:** Edición y seteo de parámetros al variador.
- **Estado del drive:** Permite ver el estado del drive, y en caso de alarmas, mostrarlas con información sobre estas mismas.
- **Start**: Comienza el movimiento del motor.
- **Stop**: Para el movimiento del motor.
- **Inicialización:** Envía comando de inicialización al variador.
- **Copiar:** Adquiere todos los valores del drive y realiza una copia en el móvil.
- **Aplicar copia:** Aplica los valores de una copia previamente hecha.
- **Información del dispositivo**: El modelo, firmware del variador, etc…

2.3 Árbol de navegación

Con toda la funcionalidad requerida ya listada, el siguiente paso es formar el árbol de navegación de la aplicación. En este árbol, se nombrarán las pantallas, su contenido y su navegación hacia otras.

Para hacer una aplicación más responsiveness y con una navegación más ágil, se ha tenido en mente diferentes tipos de navegación:

- Hamburguer Menu
- Tabs views.

Finalmente, el uso de las tabs views resulta más cómoda para el usuario, pudiendo navegar fácilmente entre diferentes funcionalidades sin necesidad de pulsar más de la

cuenta. Este cambio de tab se detallará en el diagrama como una línea discontinua. Las líneas continúan representan un cambio de ventana.

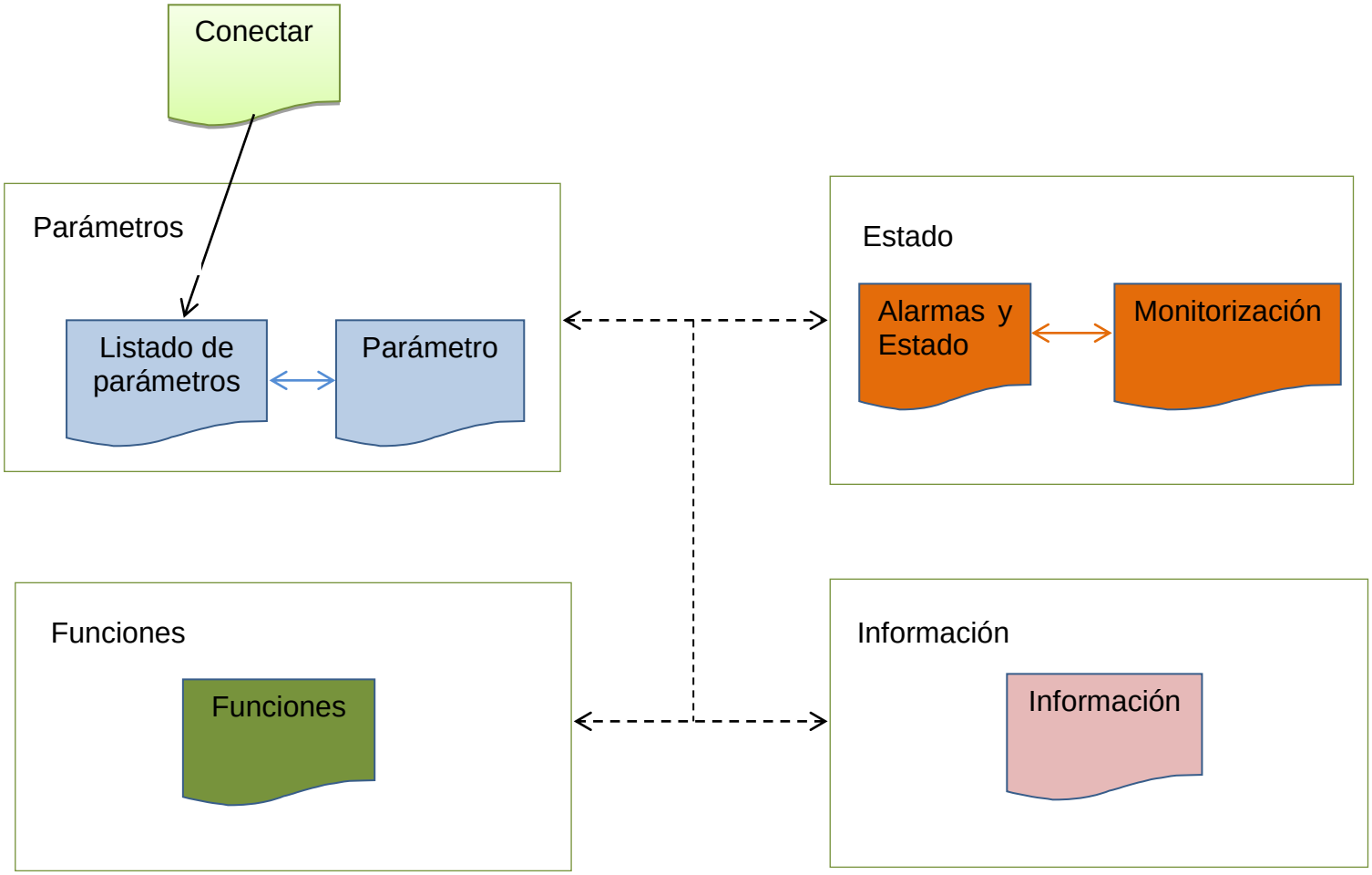


Ilustración 1: Árbol de navegación

A continuación, se detalla el contenido de cada una de ellas y su navegación:

- **Conectar**: Pantalla para permitir al usuario conectar con el variador previamente sincronizado desde el propio sistema (Android o iOS). La aplicación detectará el modelo y buscará en el servidor sus características, o si no dispone de conexión, en la memoria interna del teléfono. En caso de que el modelo sea encontrado, pasará a **Listado de Parámetros**.
- **Listado de Parámetros**. Primera pantalla mostrada al usuario una vez el modelo ha sido detectado. En ella se mostrará una lista de todos los parámetros configurables. Pulsando sobre un parámetro accederá a la pantalla

Parámetro. El usuario también se puede desplazar a **Estado**, **Funciones** y/o **Información**.

- **Parámetro**. Muestra al usuario información del parámetro: Nombre, Descripción, Rango y valor actual. Puede editar el valor actual y enviarlo al variador. El usuario puede volver hacia atrás al Listado de Parámetros y cambiar de tab a **Estado**, **Funciones** y/o **Información**.
- **Alarmas y Estado**: Muestra el estado actual del drive: No Alarma, Alarma y su información… También consta de un Botón para navegar a **Monitorización**. El usuario también se puede desplazar a **Parámetros**, **Funciones** y/o **Información**.
- **Monitorización**: Muestra una lista con los objetos de estado y su valor actual. El usuario puede volver atrás a **Alarmas y Estado** y también navegar a **Parámetros**, **Funciones** y/o **Información**.
- **Funciones**: Pantalla que consta con diferentes botones para realizar funciones. Estas son: Copiar, Aplicar e Inicializar. El usuario también se puede desplazar a **Estado**, **Parámetros** y/o **Información**.
- **Información**: Muestra información del drive: Modelo y firmware. El usuario también se puede desplazar a **Estado**, **Parámetros** y/o **Información**.

2.4 Prototipo

Con el árbol de navegación definido, ya se puede realizar un prototipo de alto nivel definiendo la composición visual de las pantallas. En las siguientes figuras se mostrará el contenido y distribución de éstas.

2.4.1 Parámetros

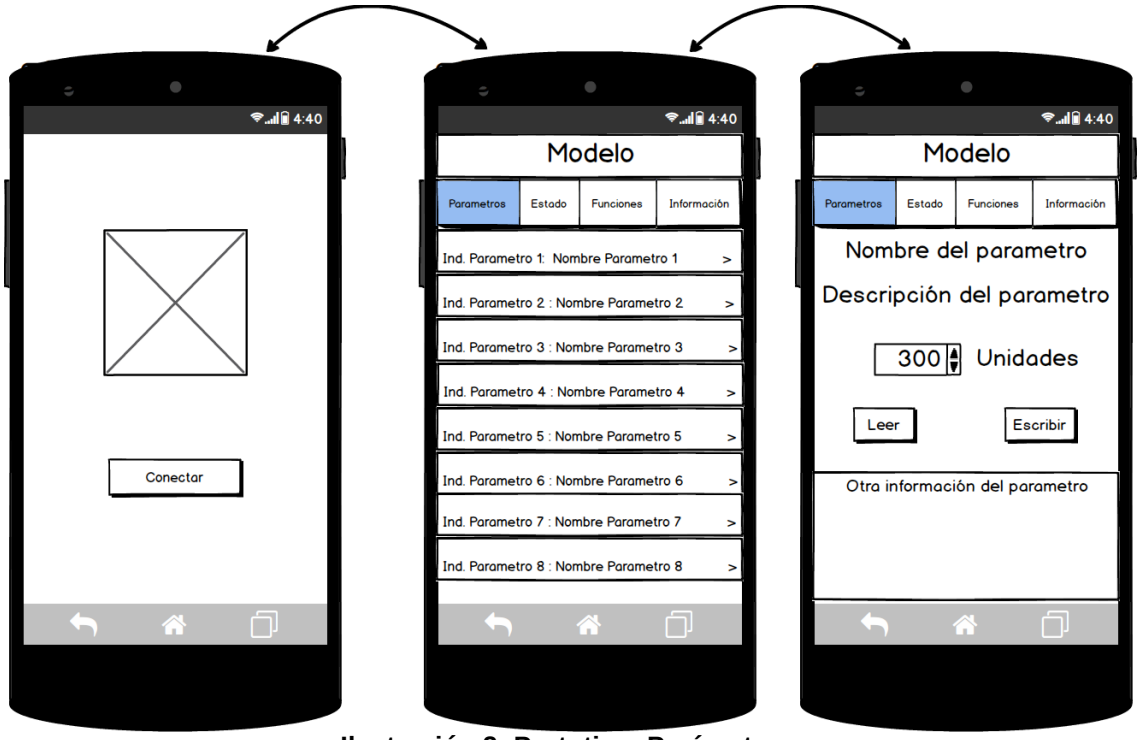


Ilustración 2: Prototipo, Parámetros.

- **Pantalla 1**: Muestra el icono de la aplicación y un botón para conectar con el variador. En caso de conexión correcta, saltará a pantalla dos. En caso de conexión incorrecta, notificará al usuario y se quedará en la pantalla.
- **Pantalla 2**: Muestra la lista de parámetros. Por cada uno de ellos, muestra el índice y su nombre. Pulsando sobre uno de ellos, se navega a la pantalla 3.
- **Pantalla 3**: Muestra el nombre y la descripción. Ofrece al usuario editar el valor del parámetro. A su lado, aparecerán las unidades. En cualquier momento, el usuario puede pulsar sobre leer, el cual obtendrá el valor del variador, sobrescribiendo cualquier cambio que el usuario haya hecho en ese parámetro. También puede pulsar sobre escribir, donde se enviará el drive el valor elegido en la aplicación. Por último, mostrará más información sobre el parámetro: Valor de fábrica, rango y ayuda. El usuario puede pulsar atrás para volver a la pantalla 2.

2.4.2 Estado

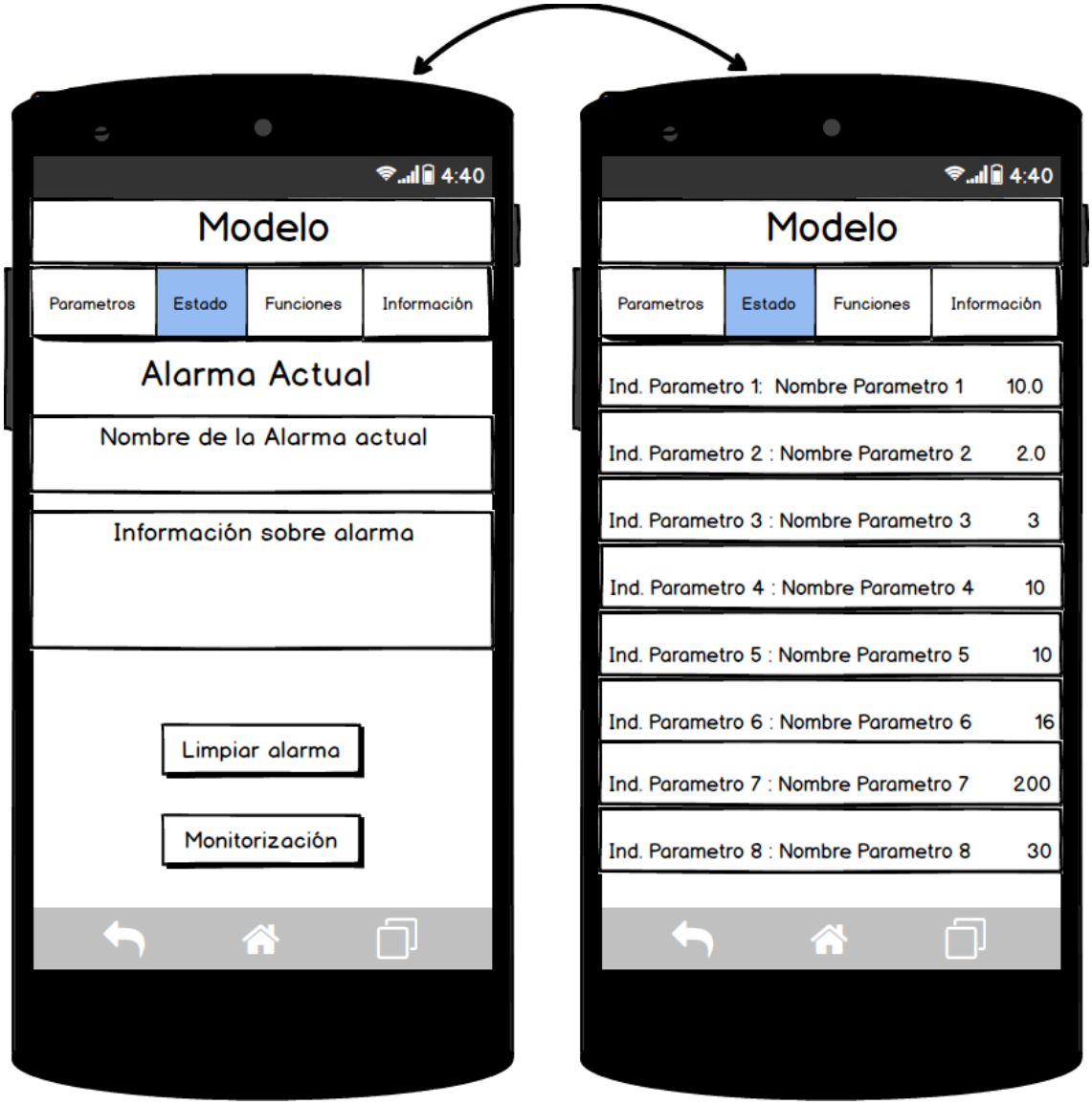


Ilustración 3: Prototipo, Alarmas

- **Pantalla 1**: Muestra la alarma actual, si la hay. También muestra información disponible sobre la alarma. Botón para Limpiar alarma, donde sí se pulsa, se enviará al variador el comando para limpiar la alarma. Pulsando sobre el botón de monitorización, se navegará a la pantalla 2.
- **Pantalla 2**: Muestra una lista de parámetros de monitorización. Por cada uno de ellos, muestra el índice, el nombre actual y el valor. El valor se actualiza cada x tiempo automáticamente.

2.4.3 Funciones

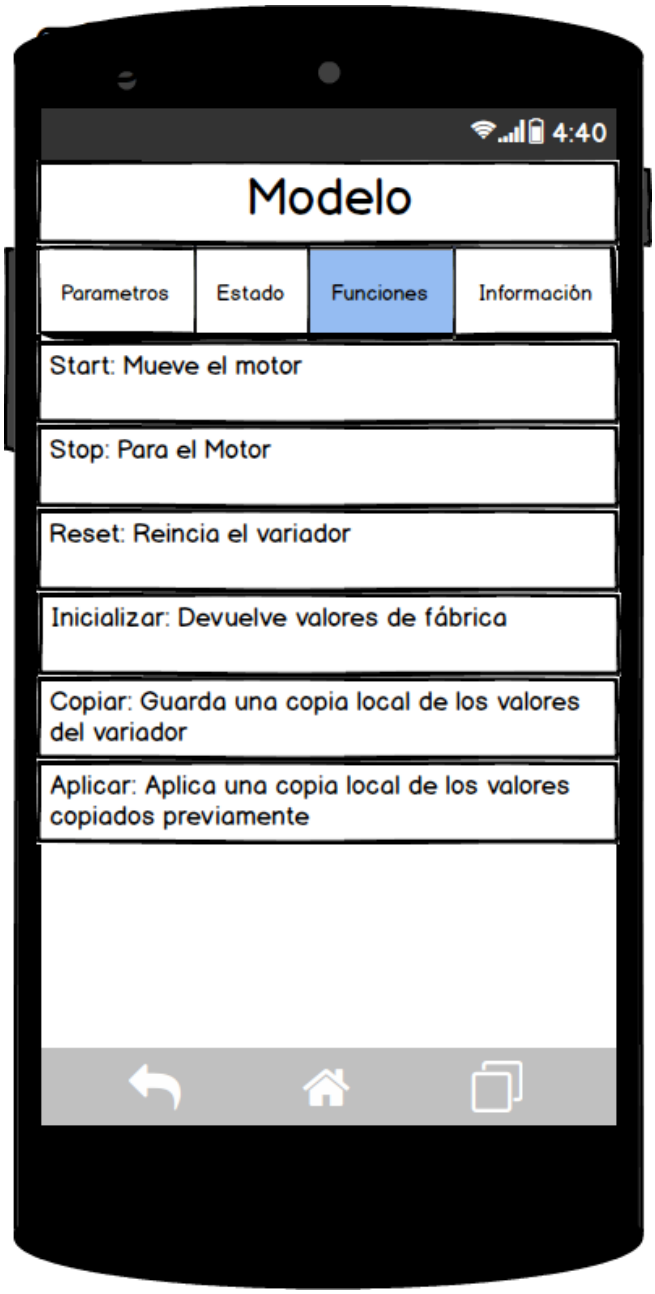


Ilustración 4: Prototipo, Funciones

- **Pantalla 1**: Lista de botones donde cada uno de ellos ejecuta una funcionalidad: Start, Stop, Reset, Inicializar, Copiar y Aplicar.

2.4.3 Información

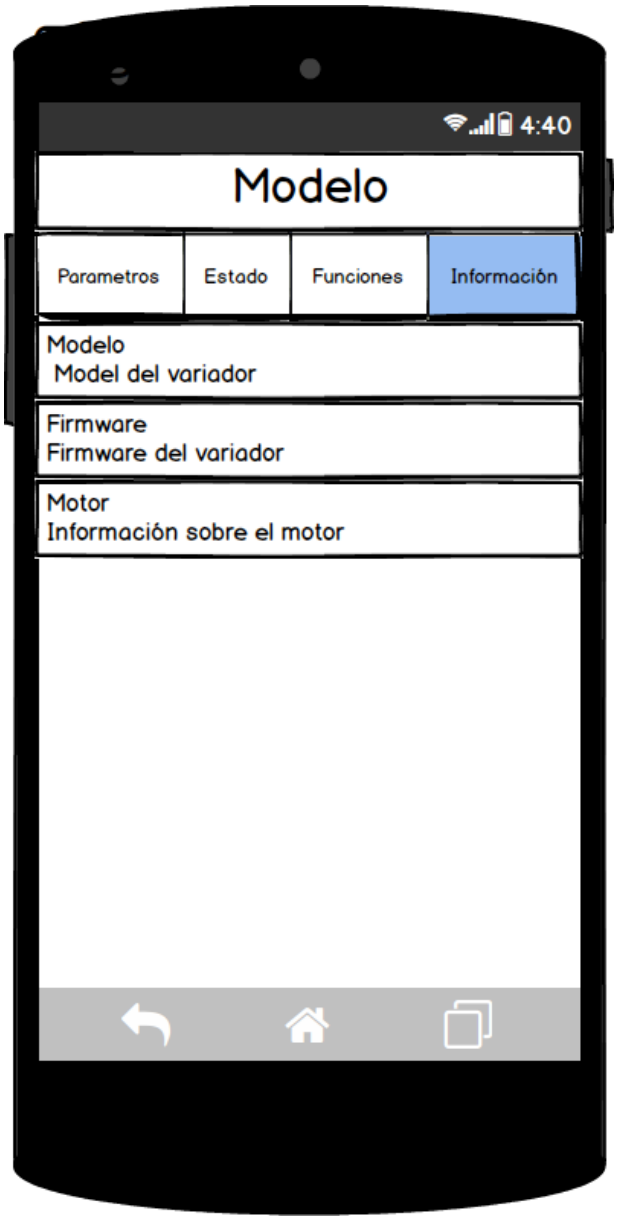


Ilustración 5: Prototipo, Información

- **Pantalla 1**: Lista de información sobre el variador: Modelo, firmware y motor.

2.5 Arquitectura

2.5.1 Tecnología

Como bien se ha comentado anteriormente en este documento, la aplicación a realizar se desarrollará utilizando Xamarin. Con Xamarin, se puede desarrollar aplicaciones Android, iOS o Windows utilizando el lenguaje C#. Una de las posibilidades que Xamarin ofrece es la de crear una aplicación con Xamarin.Forms. **[6]**

Xamarin.Forms permite desarrollar una aplicación multiplataforma compartiendo código. La ventaja de esta herramienta es que, aunque la aplicación sea multiplataforma, en tiempo de compilación, interpreta el lenguaje y genera una aplicación nativa.

Las plataformas para las que se desarrollará la aplicación son:

- Android
- iOS

En el futuro, si para la aplicación se quisiera soportar una nueva plataforma, como, por ejemplo, Windows, sería posible.

Teniendo en cuenta las plataformas, se pueden definir los diferentes módulos en los cuales la aplicación será dividida:

- **PCL (Portable Class Library)**: Éste módulo contiene código común entre plataformas: Vistas, viewModels, modelos, XML readers, generador y traductor de tramas a enviar por bluetooth… **[7]**
  Para hacer uso de una plataforma específica, se utiliza el "DependencyService": Definiendo una interfaz en el proyecto compartido e implementación en cada una de las plataformas.
- **Android**: Código específico de plataforma. En el caso de la aplicación, aquí se desarrollará la lógica para comunicar con el variador por Bluetooth.
- **iOS**: Igual que Android, pero para un dispositivo iOS.

2.5.2 PCL

A continuación, se definen las clases y sus usos definidos en **Ilustración 6.**

App
Clase donde comienza la ejecución en una aplicación desarrollada en Xamarin.Forms. Al iniciarse, asigna como página principal la vista **ConnectPage.** También se encarga de cambiar la página principal a **MainPage** cuando es necesario. Hereda de **IConnectManager** para realizar esta acción.

ConnectPage
Primera vista que se muestra de la aplicación. En ella el usuario puede conectarse a un dispositivo ya emparejado utilizando el DependencyService. Si la conexión ha sido correcta, carga toda la información del modelo para no generar bajo rendimiento más adelante haciendo uso del **InverterModelFactory**

ConnectPage
Primera vista que se muestra de la aplicación. En ella el usuario puede conectarse a un dispositivo ya emparejado. Si la conexión ha sido correcta, carga toda la información del modelo para no generar bajo rendimiento más adelante haciendo uso del **InverterModelFactory**

ConnectViewModel
Define la información a mostrar en la vista **ConnectPage,** como por el ejemplo, el progreso de la conexión.

InverterModelFactory
Singleton que se encarga de leer los datos de un XML usando **XMLReader** si no han sido leídos con anterioridad.

XMLReader
Parsea y genera los Modelos a través de la información leída del XML que define el variador.

IInverterTransferor
Interfaz que define la manera de transferir información con el variador. Cada variador puede tener una implementación diferente, aunque en el alcance de este proyecto solo se utilizará uno. Utiliza el *DependencyService* **IBluetoothManager** para la comunicación.

MainPage
Vista principal. Es una vista tabulada que contiene otras vistas: **ParameterListPage**, **StatusPage** y **FunctionsPage**.

ParameterListPage
Vista que muestra la lista de parámetros de configuración. Controla la pulsación sobre un elemento para lanzar la vista de **ParameterPage.**

ParameterListViewModel
ViewModel que contiene una lista de **ParameterViewModel**

ParameterPage
Vista que muestra la información de un parámetro y la posibilidad de leer el valor o escribirlo al variador.

ParameterViewModel
ViewModel que contiene toda la información a mostrar. Hace uso del modelo **Parameter**.

Parameter
Clase que modela un parámetro del variador en la aplicación.

StatusPage
Vista que muestra el estado actual del variador (si hay alarma, y si hay, cual es), botón para eliminar la alarma y otro para acceder a la lista de parámetros de monitorización. Si se pulsa, se navega a **MonitorListPage.**

AlarmViewModel
ViewModel que contiene la información a mostrar. Hace uso del modelo **Alarm.**

Alarm
Clase que modela una alarma del variador en la aplicación.

MontiorListPage
Vista que muestra la lista de parámetros de monitorización.

MonitorParameterListViewModel
ViewModel que contiene una lista de **MonitorParameterViewModel**

MonitorParameterViewModel
ViewModel que contiene toda la información a mostrar. Hace uso del modelo **MonitorParameter**.

MonitorParameter
Clase que modela un parámetro de monitorización del variador en la aplicación.

FunctionsPage
Vista que muestra las funciones a realizar por el variador.

FunctionsPageViewModel
ViewModel que sabe el progreso de las funciones.

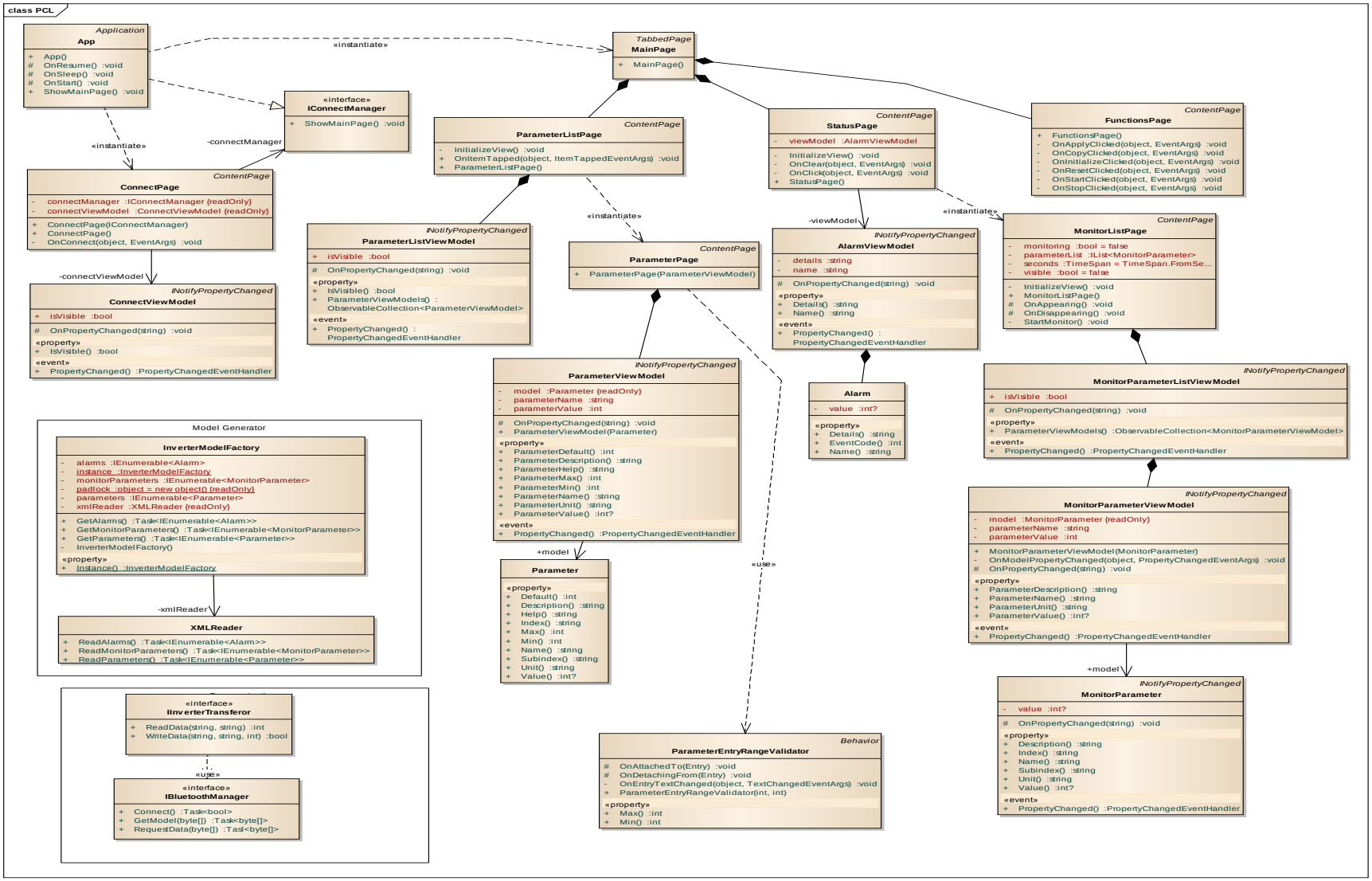


Ilustración 6: Diagrama de clases de librería PCL

2.5.3 Android

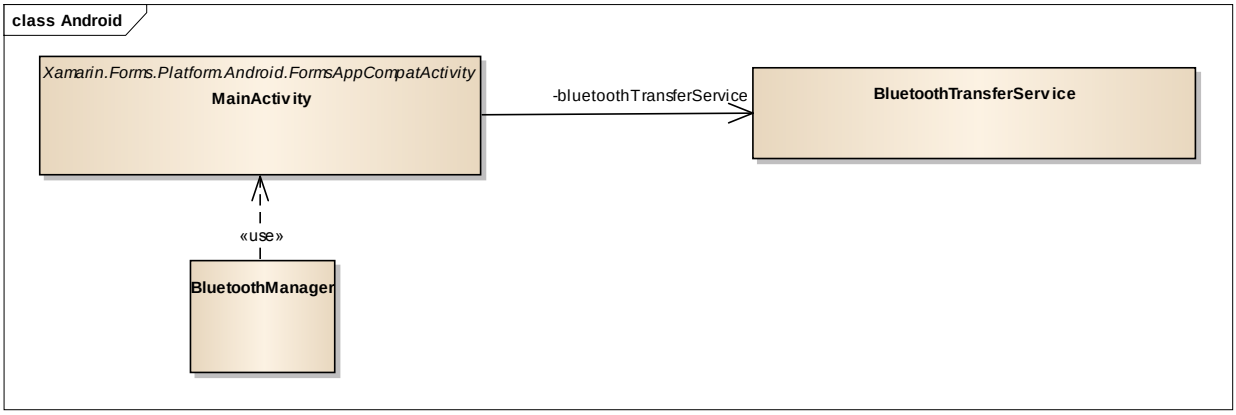


Ilustración 7: Diagrama de clases librearia Droid.

MainActivity
Android Activity que se crea al iniciar la aplicación que gestiona Xamarin en el proyecto Droid.

BluetoothManager
Implementación del *DependencyService* para Android.

BluetoothTransferService
Clase que sabe cómo conectar con otros dispositivos por Bluetooth, abrir sockets, escribir, leer…

2.5.5 Modelo de datos

La aplicación, dependiendo del modelo conectado por Bluetooth, leerá su información de un XML. En la siguiente figura se puede observar un ejemplo para poder entender la de definición de este modelo:

```xml
<Inverter>
    <Parameters>
        <Configuration>
            <Parameter Index="1000" Subindex="03" Min="25" Max="500" Default="50" Unit="HZ">
                <Name>1000.03</Name>
                <Description>Maximum output frequency 1</Description>
                <Help> It is the maximum output frequency 1
                Range: 25 to 500
                Default Value: 50</Help>
            </Parameter>
        </Configuration>
        <Monitor>
            <Parameter Index="1100" Subindex="01" Unit="">
                <Name>1100.01</Name>
                <Description>Frequency reference (p.u.) (Final command)</Description>
                <Help>311</Help>
            </Parameter>
        </Monitor>
    </Parameters>
    <Alarms>
        <Alarm EventCode="00001">
            <Name>62.00: Control Right Release Error</Name>
            <Details>The connected motor is different from the motor that was connected the last time.</Details>
        </Alarm>
    </Alarms>
</Inverter>
```

Ilustración 8: Modelo de datos en XML

Esta definición de modelo permite a la aplicación poder leer los diferentes tipos de parámetros (Configuración y Monitorización) y alarmas con toda su información.

Este modelo de datos se ha escogido sobre otro dada la flexibilidad, en un futuro, de añadir más datos y no romper compatibilidad con una aplicación que no esté actualizada, pero descargue el mismo fichero de definición.

2.5.5 Comunicaciones

Bluetooth es la tecnología de comunicación que se utiliza en el proyecto para comunicarse con un variador. La elección de esta tecnología, desde el punto de vista del fabricante, es debido a su coste. Para establecer la conexión y comunicarse, hay diferentes librerías dependiendo del Bluetooth a utilizar:

- **RFCOMM**: Conjunto simple de protocolos de transporte, construido sobre el protocolo L2CAP **[8]**.
- **LE Bluetooth**: Bluetooth de baja energía, utilizado para comunicar dispositivos cercanos sin un gran consumo de batería **[9].**

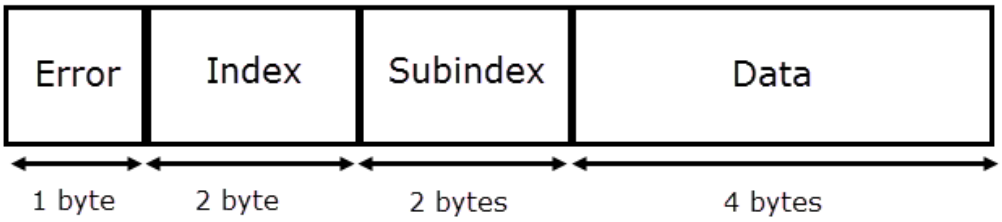
El protocolo utilizado en la aplicación es el RFCOMM debido a que no siempre el usuario va a estar muy cerca del dispositivo.

RFCOMM tiene como estándar unos servicios predefinidos con los cuales empareja e intercambia información entre dispositivos. Entre ellos se encuentra, por ejemplo, la transferencia de archivos o de audio. Estos servicios vienen definidos, entre otras cosas, por un **UUID** (Identificador único universal).

En el caso de la aplicación, se ha definido uno para el intercambio de información: *3b29c7a5-13b5-42f3-8fb7-8cb80f35aa68.*

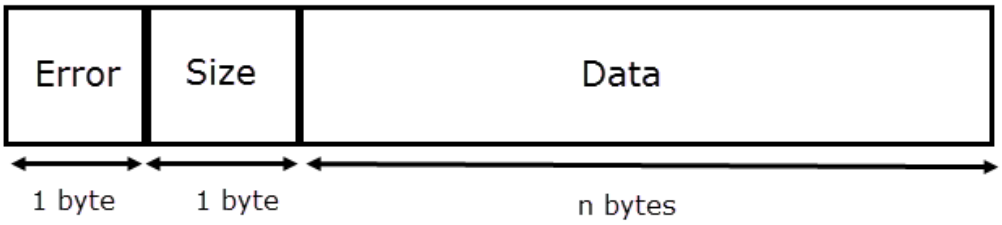
Una vez los dispositivos han sido conectados, comienza el intercambio de información. Para ello, se ha diseñado un pequeño protocolo de comunicación para intercambiar información con el variador.

Estos son los datos y el formato del paquete en el cual se envía la información hacia el variador:



- **Tipo**: Byte que define el tipo de solicitud: Leer, escribir…
- **Index**: El índice al cual se accede en el variador
- **Subindex**: El subíndice al cual se accede en el variador.
- **Data**: Datos a enviar al variador. En el caso de lectura, los datos serán ignorados en el variador, mientras que, en caso de escritura, se tendrán en cuenta.

Por cada paquete enviado, se espera uno de respuesta con el siguiente formato:



- **Error**: Byte que representa si la solicitud previamente enviada era errónea.

- **Size**: Byte que marca el tamaño de la respuesta.
- **Data**: La respuesta a la solicitud. En caso de haber sido escritura, data será 0, en caso de lectura, el tamaño de data vendrá definido por el campo size.

La aplicación ha sido diseñada para tener comunicaciones síncronas. Esto quiere decir, que, por cada solicitud enviada, siempre se esperará una respuesta a esa solicitud.

3. Remotinv

3.1 Aspecto final

En las siguientes figuras, se puede observar el aspecto final de la aplicación.

Ilustración 10: Connect

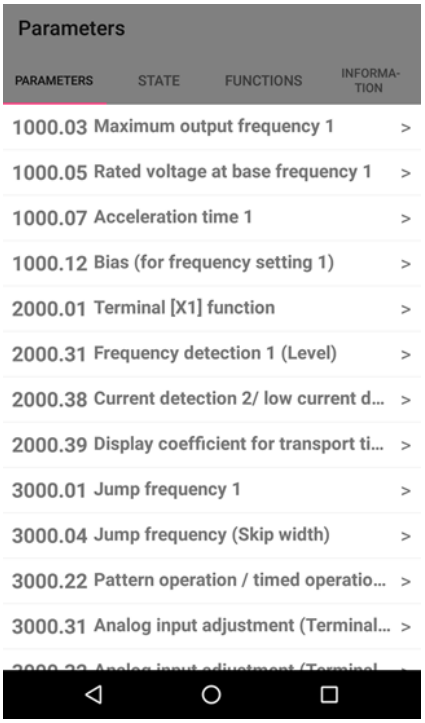


Ilustración 9: Parameters

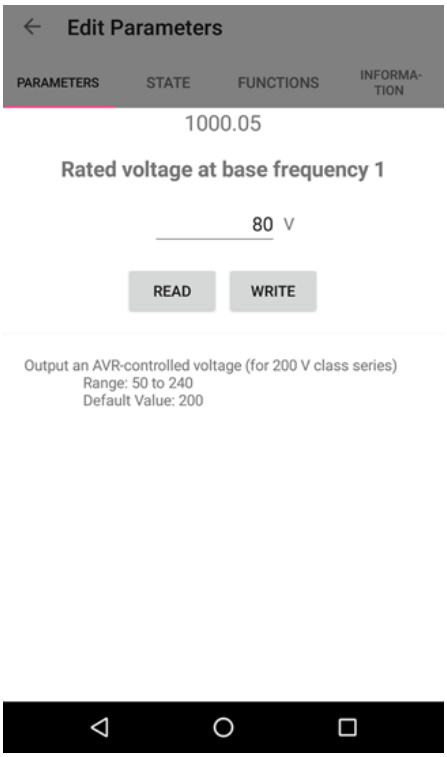


Ilustración 12: Edit Parameters

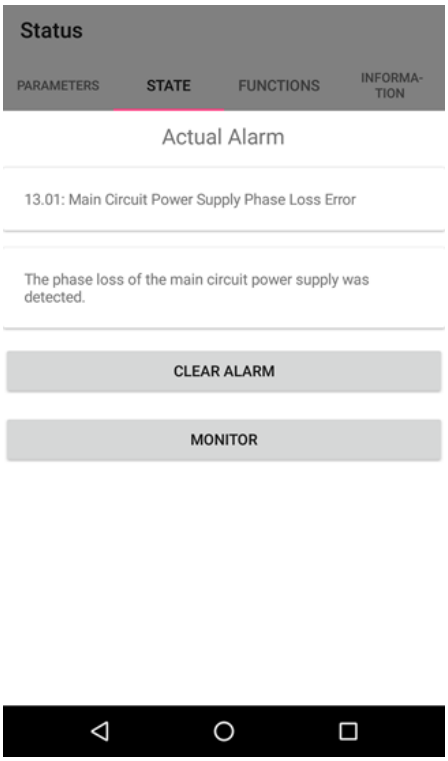


Ilustración 11: Status

Ilustración 14: Monitorization

Ilustración 15: Functions

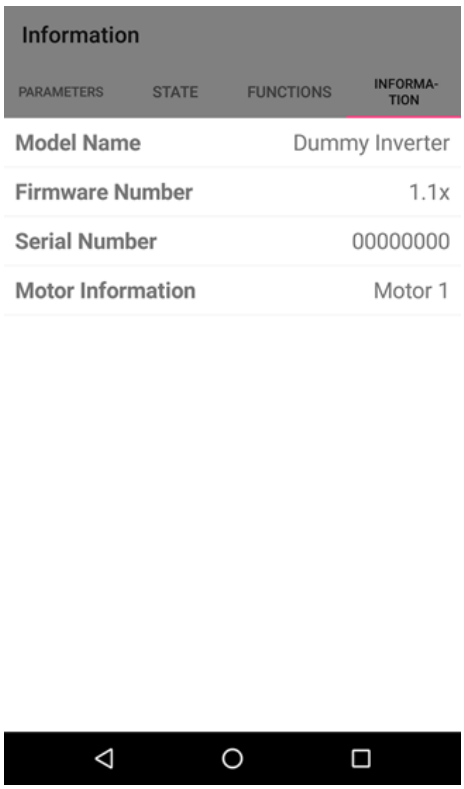


Ilustración 13: Information

3.2 Desarrollo

A continuación, se detallan puntos a destacar en la implementación:

- Uso de *Behavior* para no dejar al usuario introducir campos fuera de rango para los parámetros.
- Uso de sockets para el intercambio de información entre aplicación y el simulador.
- **MVC**: Todas las vistas tienen un *ViewModel* que definen su contenido, accediendo al modelo. Cada una de las ventanas utiliza *Binding* y *INotifyPropertyChanged* para actualizar la vista por cada cambio en el model.
- Uso de interfaces donde puede habas más de una implementación para dotar de una mejor infraestructura. Por ejemplo, *IInverterConnectionManager*: contiene lógica para codificar y decodificar las comunicaciones.
- Crear un fichero xml interno donde guardar los valores actuales del variador para ser aplicados más tarde.

4. Simulador de Variador

Al no disponer de un hardware con la tecnología adecuada, para simular el funcionamiento y las comunicaciones entre móvil y variador, se ha realizado una aplicación paralela.

Ésta aplicación es una UWP (Universal Windows Application). Esto quiere decir que puede correr en cualquier ecosistema de Windows 8 y 10 (Pcs, Tablets, móviles…). Al igual que la aplicación para móvil, hace uso del protocolo RFCOMM, requiriendo tener Bluetooth en el dispositivo para hacerla funcionar.

Cuando se arranca la aplicación, inicializa todos los objetos necesarios para simular el variador y comienza a escuchar cualquier petición exclusiva a su servicio Bluetooth. Una vez cumplida esa petición, el simulador y la aplicación móvil pueden comenzar a comunicarse.

Para una correcta simulación de un variador, el cual puede cambiar en tiempo real, la aplicación permite:

- Visualizar en todo momento los valores de configuración y monitorización.
- Cambiar los valores de configuración.
- Monitorizar la alarma actual y/o configurarla.
- Visualizar el estado de movimiento del motor. Rojo simboliza un motor parado, mientras que verde, motor en movimiento.

Este simulador es necesario para la prueba y verificación del funcionamiento de la aplicación móvil.

A continuación, se detallan los elementos mostrados en la **Ilustración 16**:

1. Estado del variador. Muestra si está desconectado en cambio está compartiendo datos con la aplicación
2. Lista de parámetros de edición. Cambiable por el usuario.
3. Lista de parámetros de monitorización. Cada segundo cambia de valor generado aleatoriamente.
4. Muestra la alarma actual. Puede ser cambiada haciendo uso del combobox.

5. Led que muestra si el motor está en movimiento o no. Rojo simboliza el no movimiento, y verde, movimiento.

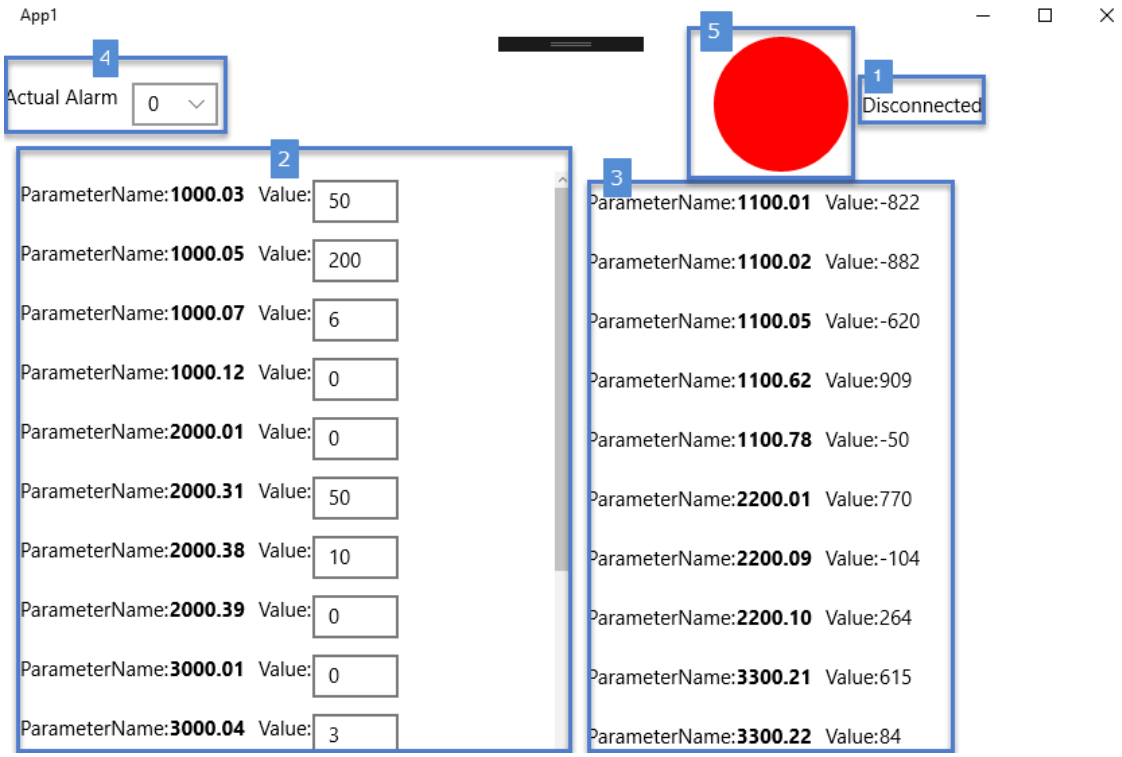


Ilustración 16: Simulador de Variador

5. Pruebas y verificación

Debido al ajustado tiempo, no se han realizado pruebas unitarias. Aun así, se han suplido con pruebas funcionales de la aplicación. Gracias a estas pruebas funcionales, se han detectado y corregido varios errores, para luego, ser subsanados. Algunos otros no han sido subsanados por mayor complejidad y tiempo necesitado.

A continuación, se muestra una tabla con las pruebas realizadas:

Tabla 1: Pruebas y verificación

Caso	Prueba	Resultado esperado	Resultado	Arreglado?	Resultado
Conectar	Conectar teniendo bluetooth desactivado	Toast indicando fallo	OK		
	Conectar con dispositvos emparejados pero simulador apagado	Toast indicando fallo	Error	Si	OK
	Conectar con dispositivos emparejados y simulador funcionando	Pasar a página Principal	OK		
	Una vez conectado el dispositivo, desconectar bluetoth, volver a conectar y continuar.	Seguir transmitiendo e intercambiando datos.	Error	No	
Parámetros	Abrir página de vista de parámetros	La página se muestra como es deseado	OK		
	Pulsar sobre párametro en la lista	Navega a Editar Parámetros	OK		
Editar Parámetro	Datos correctos	Índice, nombre, ayuda, unidades…	OK		
	Pulsar sobre leer	El valor leído es igual que en el simulador	OK		
	Editar valor	El valor es editado	OK		
	Editar valor fuera del rango	Valor fuera del rango no es seteable	OK		
	Escribir valor	El valor es transferido y el simulador cambia de valor	OK		
Estado	Abrir página de estado	La página de estado muestra alarma y botón para monitorizar	OK		
	Cambiar el simulador y ver reflejada la alarma en la aplicación	La alarma es reflejada en la aplicación	OK		

	Limpiar la alarma	La alarma desaparece de la aplicación y del simulador	OK		
	Pulsar sobre parámetros de monitorización	Navega a Monitorizar Parámetros	OK		
Monitorizar parámetros	Datos correctos	La lista de parámetros es correcta, con unidades	OK		
	Cada 2 segundos o menos los valroes se actualizan	Los valores son actualizados	OK		
Funciones	Todas las funciones disponibles	Todo correcto	OK		
	Pulsar sobre Run	El simulador pasa de rojo a verde	OK		
	Pulsar sobre Stop	El simulador pasa de verde a rojo	OK		
	Pulsar sobre initialize	El simulador cambai todo los valores al default	OK		
	Pulsar sobre Copiar	Los datos son copiados en la memoria del teléfono	Error	Si	OK
	Pulsar sobre Aplicar	Los datos son aplciados y los valores en el simuador cambian	OK		
Información	Lista sobre información mostrada	Modelo, Firmware, Numero de serie y Motor	OK		
	Leer Modelo	Leído y mostrado correctamente	Error	Si	OK
	Leer Firmware	Leído y mostrado correctamente	OK		
	Leer Número de Series	Leído y mostrado correctamente	OK		
	Leer Motor	Leído y mostrado correctamente	OK		

6. Procedimiento y resultado

Se ha intentado seguir la planificación inicial durante el máster, pero debido a diferentes factores, no ha sido posible entregar todo lo planeado.

Estos factores han sido diferentes, desde mala estimación en la planificación a imprevistos. A remarcar:

- Semana Santa. Durante la planificación no se tuvo en cuenta y fue una semana en la que no se trabajó. Tiempo estimado perdido: 12 horas.
- Xamarin Studio para Mac: Por alguna razón, dejó de debugar correctamente, haciendo imposible parar la ejecución en breakpoints y dificultando la corrección de errores. Finalmente se solventó instalando Visual Studio para Mac. Tiempo estimado perdido: 4 horas.
- Comunicación Bluetooth: Al principio, la comunicación se intentó establecer entre la UWP utilizada desde una máquina virtual haciendo uso de un USB Bluetooth. Fue imposible que funcionara de éste modo, pensando que la implementación estaba mal y probando muchas alternativas. Finalmente, se cambió de PC con Windows 10 y funcionó. Tiempo estimado perdido: 14 horas.

Todo esto ha hecho que el proyecto se retrasará y no se pudiera acabar toda la funcionalidad.

Funcional entregada:

- Leer y escribir parámetros.
- Monitorizar y limpiar alarma.
- Monitorizar parámetros de Estado.
- Funciones
- Consultar información de modelo, firmware, etc…
- Sistema operativo Android.

Funcionalidad no entregada.

- iOS: Se ha dado prioridad a acabar la parte Android totalmente y sin errores. Se estima que con una semana más de esfuerzo, podría haber sido incluido.
- Errores pendientes por arreglar.
- El modelo de datos se obtiene de los XMLs preinstalados con la aplicación. Si finalmente, la aplicación soporta muchos modelos de variadores, su tamaño

podría ser demasiado grande. Debido a esto, sería conveniente obtener el XML de un servidor una vez el modelo ha sido identificado.

7. Conclusiones

En la realización del proyecto, se han puesto en práctica muchos conceptos aprendidos durante la realización del máster:

- Planificación y diseño de una aplicación móvil
- Desarrollo de una aplicación móvil.
- Conectividad.
- Toma de decisiones en el desarrollo.

También, cabe destacar que desde un principio se ha decidido por tomar un camino para el desarrollo no incluido en la docencia:

- Desarrollo de aplicación móvil nativa multiplataforma con el uso de Xamarin.
- Uso de conectividad Bluetooth.
- Desarrollo de una Universal Window Application.

En cuanto a objetivos, gran parte de ellos han sido cumplidos. El más importante, ofrecer un producto mejor que el operador digital por conexión inalámbrica, ha sido realizado con creces: Permitir al usuario ver la lista de parámetros fácilmente, con edición y sus características, fácil acceso a alarmas, información, etc…

En cambio, por el lado negativo:: El desarrollo de la aplicación para iOS. Por motivos anteriormente explicados, el tiempo se ha hecho corto y no ha podido ser entregado por los problemas de conectividad encontrados.

Otro punto negativo, ha sido el no seguimiento de la metodología SCRUM hacia el exterior. En otras palabras, en el desarrollo sí que se ha seguido haciendo entregas de funcionalidad, pero estas no fueron expuestas al tutor paulatinamente.

Uno de los cambios que se realizó durante el desarrollo fue el cambio de plataforma para el simulador. Fue concebido para ser una aplicación WPF, pero se pensó que siendo una UWP, se ampliaría más el conocimiento del desarrollo par aplicaciones móviles, ya que está aplicación puede ejecutarse en un amplio número de dispositivos móviles con Windows.

La ventaja de la toma de decisión de la arquitectura y su posterior diseño ofrece una amplia ventana para trabajo futuro:

- Añadir iOS o Windows debería ser sencillo ya que solo habría que implementar la sincronización y transmisión de datos por Bluetooth.
- El diseño permite, a través de factorías y estrategias, añadir fácilmente modelos nuevos e implementar nuevos protocolos de comunicación.

8. Glosario

- **Variador o Inverter**: Variador de frecuencia
- **UWP**: Universal Windows Application
- **XML**: Lenguaje de marcado extensible

9. Bibliografía

[1] Variador de frecuencia (08/03/2017)

[2] Manual de variador Fuji Electric (08/03/2017)

[3] Omron Digital Operator for Inverter (08/03/2017)

[4] Siemes g120 drive components (08/03/2017)

[5] Siemens Simatic s7 (08/03/2017)

[6] Xamarin Forms (25/04/2017)

[7] Portable Class Library (25/04/2017)

[8] Protocolo RFCOMM (30/05/2017)

[9] Bluetooth LTE (30/05/2017)

10. Anexos

10.1 Instalar Simulador

Requisitos:

- Windows 10
- Habilitar el dispositivo para desarrollo
- Cambiar nombre del dispositivo para que contenga la palabra INVERTER

Como Instalar:

- Botón derecho sobre Add-AppDevPackage y clickar sobre Ejecutar con PowerShell.
- Afirmar en todos los diálogos.
- Si falla, pulsar sobre Simulator_1.0.0.0_x86_x64_arm.cer e instalar el certificado.
- Repetir primer paso.
- Ejecutar aplicación.

10.2 Instalar Remotinv

Requisitos:

- Minimum Android Version: 4.0.3 (API level 15)
- Target Android Version (API 25)
- Ejecutar Simulador

Como Instalar:

- Ejecutar el fichero com.uoc.tfm.apk.