MobDE
Una aplicación móvil de captura de datos.

Luís Pardo Martí
Máster Universitario de Desarrollo de Aplicaciones para Dispositivos Móviles

Profesores:
Carles Garrigues Olivella
Pau Dominkovics Coll

Junio de 2018

Dedicat a:
La meva filla Mavi, pel temps que no li he pogut dedicar.
La meva dona Yanki, pel suport donat.

FICHA DEL TRABAJO FINAL

Título del trabajo:	MobDE Una aplicación móvil de captura de datos
Nombre del autor:	Luís Pardo Martí
Nombre del consultor:	Carles Garrigues Olivella Pau Dominkovics Coll
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Resumen del Trabajo:

Creación de una aplicación móvil Android nativa orientada a la captura de datos *en el lugar* y su comunicación a un servidor central.

La definición de las estructuras a capturar se descarga del servidor central y los datos enviados posteriormente a este para su consulta y proceso unificados.

El proceso de captura de datos puede ser offline y se definirá un sistema de resolución de conflictos para la sincronización de datos.

Abstract:

Development of a native Android mobile app for the *on-site* data capture and its communication to a central server.

The data structure definition is downloaded from central server where data will be send later for centralized processing.

The data capture process may be offline, and a conflict resolution process will be defined for data synchronization.

Palabras clave:

Captura, conflictos, diseño, datos

Índice

Lista de ilustraciones

Lista de tablas

Lista de diagramas

1 Introducción

1.1 Contexto y justificación del Trabajo

En mi vida profesional trabajo con una base de datos (Universe) con un número de instalaciones muy inferior a las grandes (SQL Server, MySQL, Oracle, Informix…), y con unas características propias muy diferentes a las de estas, eso hace que el número de herramientas "del gran público" disponibles para este entorno sea muy reducido.

Entre las carencias que tiene ese entorno actualmente es que apenas hay herramientas para comunicar con el entorno móvil, de ahí la idea de crear una aplicación que permitiera dotar de movilidad al entorno.

A partir de esa premisa nace este proyecto, aunque realmente no se limita a esta base de datos, sino que sería válido para cualquiera adaptando el lado servidor (de hecho, finalmente el desarrollo del lado servidor ha sido contra una base de datos SQL Server).

Así pues, el **objetivo del proyecto es conseguir una herramienta móvil que sea capaz de mostrar datos obtenidos de un servidor** y disponer de una herramienta de captura de datos (con unos mínimos de validación) sin que sea un desarrollo estático, sino que esté **basado en metadatos obtenidos también del servidor**, de manera que tengamos un producto **totalmente integrable** en cualquier organización sea cual sea su base de datos subyacente.

1.2 Objetivos del Trabajo

El objetivo del trabajo es construir una aplicación que demuestre que es posible generar pantallas de modo dinámico en base a metadatos con enlace a datos almacenados localmente con sincronización contra un servidor central de datos y metadatos.

El objetivo final no es disponer de una versión "vendible" de la aplicación sino más bien una prueba de concepto, de modo que demuestre la viabilidad de una solución de este tipo. La implementación final y completa de la solución requiere de mucho más tiempo tanto de análisis como de desarrollo para cubrir un abanico de situaciones lo suficientemente amplio.

1.2.1 Requisitos funcionales

- Cargar los formularios diseñados remotamente.
- Registrar información en base de datos local sobre dichos formularios.
- Comunicar con el servidor central para cargar y enviar datos.
- Gestionar los conflictos de sincronización.

1.2.2 Requisitos no funcionales

- Deberá gestionar correctamente las situaciones de falta de cobertura. Esto se consigue registrando siempre en una base de datos local con unas banderas de estado de comunicación y un sistema de reenvío.

Eso sí, es necesaria la cobertura en el arranque de la aplicación para descargar la versión actualizada del entorno.

- Debe funcionar sobre el máximo número de dispositivos Android posible. Esto se consigue seleccionando un nivel de API lo suficientemente bajo como para cubrir un 90% del parque móvil.

1.2.3 Plataforma de destino

Generalmente consideramos la API 16 como la API que mayor rango de dispositivos permite cubrir, pero en este caso seleccionaremos la API 19 como API de destino puesto que la cantidad de dispositivos que quedan fuera de la criba es mínima.

Version	Codename	API	Distribución	Acumulada
8.1	Oreo	27	0,30%	0,30%
8.0		26	0,80%	1,10%
7.1	Nougat	25	6,20%	7,30%
7.0		24	22,30%	29,60%
6.0	Marshmallow	23	28,10%	57,70%
5.1	Lollipop	22	19,20%	76,90%
5.0		21	5,40%	82,30%
4.4	KitKat	19	12,00%	94,30%
4.3	Jelly Bean	18	0,70%	95,00%
4.2.x		17	2,60%	97,60%
4.1.x		16	1,70%	99,30%
4.0.4	Ice Cream Sandwich	15	0,40%	99,70%
4.0.3 -				99,70%
2.3.7	Gingerbread	10	0,30%	100,00%
2.3.3 -				100,00%

Tabla 1. Distribución de las versiones de Android[1]

1.3 Enfoque y método seguido

Aunque pueden existir soluciones “parecidas” en el mundo empresarial lo cierto es que no he localizado nada en esta línea. Cabe tener en cuenta que desde el principio el objetivo de esta solución es comunicar directamente con un servidor propio de la organización, así que, por diseño, es una solución que siempre deberá ser personalizada para cada organización (ni que sea únicamente a nivel de instalación del servidor).

De algún modo, las Microsoft powerapps[2] van en la línea del producto que busco, pero, lógicamente orientados a un ecosistema Microsoft.

[1] https://developer.android.com/about/dashboards/index.html

[2] https://powerapps.microsoft.com/es-es/

Por otra parte, Field Data Integrators con su producto naturalForms[3] es un ejemplo que seguir por su capacidad de incorporar controles fuera de los clásicos textbox.

Se ha decidido realizar una aplicación nativa, en este caso en entorno Android, esto permite aprovechar las características de los dispositivos a fondo sin, a priori, depender de librerías de terceros. En un futuro, la versión comercial podría tener su versión iOS también nativa por el mismo motivo.

Las ventajas de la programación nativa nunca deben ser subestimadas en una aplicación de empresa, dado que, por ejemplo, puede ser interesante conectar con lectores de código de barras, escáners, lectores/grabadores rfid y otros cuya conexión desde aplicación híbrida sería muy complejo o imposible.

Por otra parte, la generación dinámica de contenido permite disponer de una aplicación que se instala una vez y se mantiene actualizada de modo totalmente dinámico sin necesidad de una actualización de entorno como tal. En una aplicación convencional, si necesitamos agregar un único campo en una única pantalla nos veremos obligados a actualizar la aplicación completa, en este entorno basta con corregir los metadatos de la tabla y en la siguiente conexión a la aplicación estará disponible.

El back-end consta de una capa muy ligera desarrollada con una librería muy reducida de servicios web que proporcionan los servicios necesarios para el frontal y el enlace con el back-end empresarial del cliente (fuera del ámbito de este proyecto).

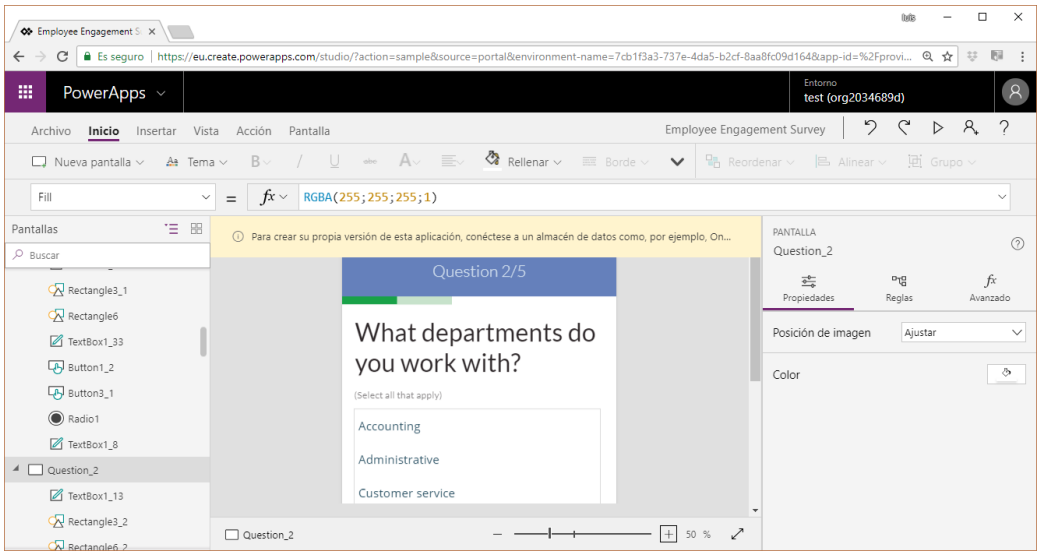
La comunicación entre el back-end y el front-end siempre es a través de json que siguen el formato establecido, en esta versión, incluso el diseño de las tablas y formularios está guardado en json en la base de datos, esto permite trabajar con una base de datos muy simple y estable puesto que futuras iteraciones en el diseño de los json no requieren modificar la base de datos sino simplemente incorporar los atributos necesarios en las cadenas que lo requieran.

1.3.1 Microsoft powerapps

Microsoft presenta un entorno de generación rápida de aplicaciones que en cierto modo asemeja lo que pretende conseguir Mobde. Pero el enfoque de Microsoft está orientado a su ecosistema (como es lógico) y a aplicaciones web, no nativas.

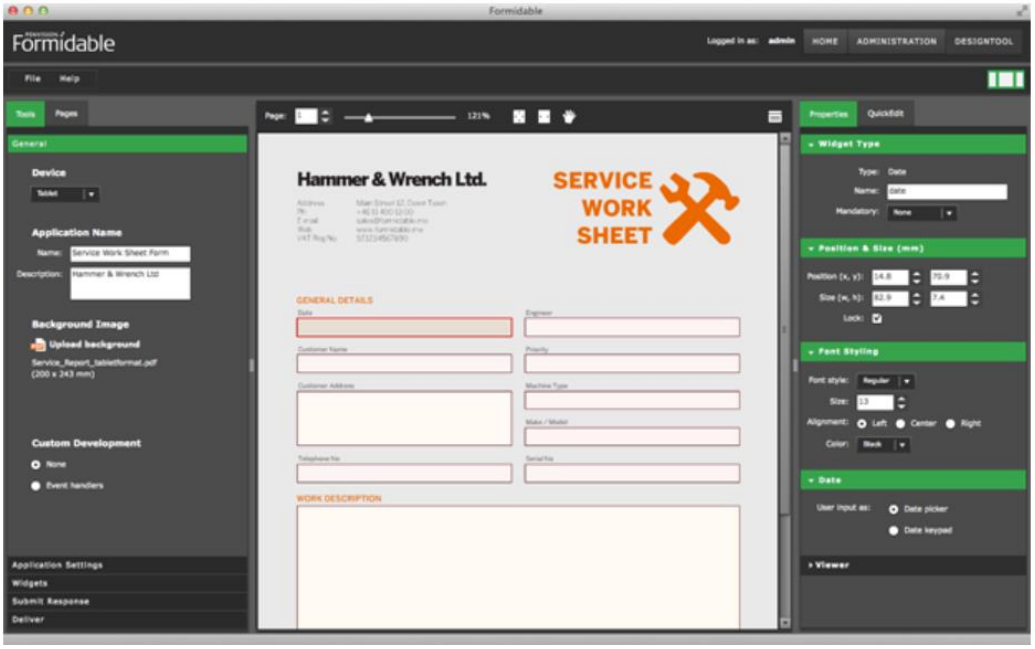
Evidentemente se trata de un entorno de diseño muy evolucionado con multitud de opciones y plantillas predefinidas.

[3] https://fdiforms.zendesk.com/hc/en-us/articles/214933417-Tablet-forms-for-iOS-and-Android



1.3.2 Naturalforms

Este producto aparentemente sí es un desarrollo nativo, no web, y permite trabajar offline, con versiones para Android e IOS.



Resulta atractivo el hecho de que permite diseñar formularios con controles “especiales”, en la imagen de ejemplo, se puede observar un termómetro como control para registrar una temperatura de un paciente, este es el tipo de evolución que debería seguir Mobde.

1.4 Planificación del Trabajo

En este caso la planificación viene muy marcada por los entregables que marcan unos hitos inamovibles.

Por otra parte, la disponibilidad de tiempo es muy inferior a la de un proyecto convencional por la necesidad de coordinarlo con la vida laboral (y la familiar, porque no decirlo), así que la disponibilidad queda reducida a una hora diaria y unas 10 en fin de semana. Esta disponibilidad se refiere a los tiempos dedicados en exclusiva a este proyecto, aunque a lo largo de la jornada se puedan ir dedicando otros momentos para pruebas determinadas o pequeñas intervenciones.

En cuanto a recursos *hardware-software* se dispone del siguiente entorno:

- Hardware
  - Portátil HP con Windows 10
  - MacBook Pro con macOS High Sierra (10.13.3)
  - Teléfono móvil Sony Xperia ZX1 con Android 8.0.0
  - Teléfono móvil Sony Xperia Z3 con Android 6.0.0
  - Servidor HP Proliant G8 con Windows 2012 R2, SQL Server 2014 e IIS 8.5 para el *back-end*.
- Software

- Android Studio 3.1
- SQLite
- Microsoft Office 365
- Microsoft Visio 2016
- Microsoft Project
- Justinmind
- SQL Server 2014
- DB Browser for SQLite

En resumen, las tareas a realizar corresponden al ciclo completo de un desarrollo, resumiendo:

- Diseño, que además corresponderá a la segunda entrega a realizar en fecha 4 de abril. En este caso lo que se plantea es el diseño a nivel de estructura a crear, en el sentido de que como se ha comentado anteriormente en algunos puntos nos quedaremos en la prueba de concepto en lugar de una implementación completa. Por la naturaleza de la aplicación a desarrollar la fase de análisis realmente será iterativa debido a que cada incorporación de una nueva funcionalidad en el entorno requerirá del análisis completo para no alterar la funcionalidad preexistente.
- Implementación, que además corresponderá a la tercera entrega a realizar en fecha 16 de mayo. En este caso se realizará el desarrollo del entorno, tanto servidor como aplicación cliente. Nuevamente, al tratarse de una prueba de concepto y dado que la implementación final debería ir ligada a un entorno empresarial, el desarrollo de la parte servidora será muy ligero, orientado únicamente a demostrar la funcionalidad, sin mayores aspiraciones.
- Entrega final, que corresponde con la cuarta entrega a realizar en fecha 6 de junio. Consiste en la revisión de la documentación que se habrá ido preparando durante el resto de las entregas, así como la preparación de un video de presentación.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**MobDE**	**260 días?**	**dom 11/03/18**	**vie 22/06/18**	
2	**Diseño**	**53 días**	**dom 11/03/18**	**mié 04/04/18**	
3	Usuarios y contextos	6 horas	jue 15/03/18	sáb 17/03/18	
4	Diseño conceptual	11 horas	lun 19/03/18	dom 25/03/18	3
5	Prototipado	9 horas	dom 25/03/18	vie 30/03/18	4
6	Evaluación	7 horas	dom 01/04/18	mié 04/04/18	5
7	Redacción memoria	4 horas	dom 11/03/18	dom 11/03/18	
8	**Implementación**	**102 días**	**jue 05/04/18**	**vie 04/05/18**	**6**
9	**Entorno servidor**	**20 días**	**jue 05/04/18**	**sáb 14/04/18**	
10	Creación base de datos proyectos	6 horas	jue 05/04/18	sáb 07/04/18	
11	Creación modelos y contratos	6 horas	sáb 07/04/18	dom 08/04/18	10
12	Creación servicios	8 horas	lun 09/04/18	sáb 14/04/18	11
13	**Entorno cliente general**	**20 días**	**sáb 14/04/18**	**dom 22/04/18**	
14	Proceso de configuración (pantalla, llamada servicio)	6 horas	sáb 14/04/18	dom 15/04/18	12
15	Proceso de Login (pantalla, llamada servicios)	6 horas	dom 15/04/18	vie 20/04/18	14
16	Menú lateral, pantallas auxiliares	8 horas	sáb 21/04/18	dom 22/04/18	15
17	**Entorno cliente en base a metadatos**	**50 días**	**dom 22/04/18**	**mié 02/05/18**	
18	Investigación creación controles en tiempo de ejecucion	10 horas	dom 22/04/18	sáb 28/04/18	16
19	Entorno de carga de formularios	8 horas	sáb 28/04/18	dom 29/04/18	18
20	Entorno de búsqueda	8 horas	dom 29/04/18	dom 29/04/18	19
21	Entorno de resultados	8 horas	dom 29/04/18	lun 30/04/18	20
22	Entorno de edición	8 horas	lun 30/04/18	mar 01/05/18	21
23	Entorno de conflictos	8 horas	mar 01/05/18	mié 02/05/18	22
24	Redacción memoria	12 horas	mié 02/05/18	vie 04/05/18	23
25	**Final**	**44 días**	**jue 17/05/18**	**mar 05/06/18**	
26	Correcciones finales	20 horas	jue 17/05/18	sáb 26/05/18	
27	Revisión memoria	12 horas	sáb 26/05/18	vie 01/06/18	26
28	Preparación presentación	12 horas	sáb 02/06/18	mar 05/06/18	27
29	Tribunal	5 días?	lun 18/06/18	vie 22/06/18	

[4]

Diagrama. 1, secuencia de tareas y horas asignadas

[4] Hay un problema con mi versión de Project y aunque las tareas las indico en horas hace sumatorios erróneos como si cada hora fuera un día (aunque he definido la jornada laboral correctamente, así como la semana laboral de 12 horas).

1.5 Breve sumario de productos obtenidos

Al finalizar el trabajo se entregarán los siguientes productos:

- Memoria del trabajo (este documento).
- Aplicaciones, en particular:
  - APK Android objeto del proyecto
  - Servidor web de comunicación con el *back-end*
  - Base de datos SQL Server del *back-end*.
- Presentación en video explicativa de la aplicación y el trabajo realizado.
- Archivos de código fuente.

1.6 Descripción del resto de capítulos de la memoria

1.6.1 Análisis

El apartado de análisis del documento final contendrá toda la información referente a los usuarios de la aplicación, contexto en que la utilizan e interacción entre los diferentes usuarios y procesos.

Esta información se concretará en forma de fichas para identificar a los usuarios y su interacción con el sistema, así como la lista de tareas que los usuarios necesitan y los elementos requeridos en la interfaz de la aplicación.

1.6.2 Diseño

En el apartado de diseño incluiremos la información relativa a los escenarios de uso contemplados, así como los flujos de interacción.

Este apartador también incluirá la definición del prototipo de la aplicación, tanto las primeras versiones a mano alzada como la versión final de alta fidelidad realizada ya con el lenguaje de Desarrollo.

Finalmente se incluirá información de diseño técnico como la definición de los casos de uso y el diseño de la arquitectura.

1.6.3 Pruebas

Explicación del método de realización de pruebas y resultado de estas.

1.6.4 Interfaz de usuario

Definición de los aspectos gráficos de la aplicación. De algún modo es la evolución de los diseños realizados a mano alzada y con la herramienta de prototipado.

1.6.5 Implementación

Detalles de implementación de la aplicación. Entre otras cuestiones, detalla las decisiones de diseño de clases y comunicación entre procesos tomadas a lo largo del proceso de implementación, así como detalles técnicos que se consideren interesantes o relevantes.

1.6.6 Conclusiones

Detalle de los objetivos alcanzados, objetivos pendientes, mejoras a realizar, etc.

También se hace hincapié en aquellos aspectos necesarios para convertir esta aplicación en una aplicación comercial.

1.6.7 Glosario

Relación de términos y definiciones.

1.6.8 Bibliografía

Relación de todas las fuentes externas al material de la asignatura utilizadas en la realización del proyecto.

1.6.9 Anexos

Relación de otros documentos que por su especificidad o volumen no se incluyen dentro de la memoria, como el manual de usuario y el de instalación.

2 Análisis

2.1 Usuarios y contextos de uso

En el caso particular de este desarrollo y habida cuenta que se trata de una prueba de concepto más que una aplicación “final” el origen de las necesidades del proyecto, así como los requisitos a cumplir los establece el mismo desarrollador, de manera que no hay entrevistas formales con usuarios sino más bien una exteriorización de unas necesidades vistas a lo largo del tiempo.

Eso no implica que no se pueda simular un entorno convencional con usuarios finales y unos requisitos establecidos por un “cliente”, simplemente en este caso el cliente y el proveedor son dos facetas del desarrollador (entendiendo además que en este contexto desarrollador agrupa todas las funciones de analista, diseñador, implementador, etc.).

Una vez realizada esta prueba de concepto es posible que esto se transforme en un producto “real” y en ese caso sería necesario realizar una nueva iteración completa, identificando usuarios reales y potenciales y realizando entrevistas con ellos para localizar nuevas necesidades.

2.1.1 Usuarios

Como ya se ha indicado en el punto anterior, el usuario que genera esta necesidad es el mismo que realiza el proyecto, así que podríamos considerar que la metodología utilizada ha consistido en la observación e investigación contextual observando el entorno habitual y detectando las carencias a lo largo del tiempo.

MobDE debe considerar dos tipos de usuario, el usuario de *back-end* que es el que genera los metadatos que la aplicación consumirá y el usuario de la aplicación móvil que básicamente introducirá información.

Por el concepto de la aplicación el usuario final no es tan importante como en otros desarrollos por tratarse de una prueba de concepto, necesitamos cubrir unas mínimas necesidades del usuario, pero lo realmente importante es demostrar que es posible.

Pese a esta puntualización, lo que está claro es que el usuario de la aplicación móvil necesitará realizar una serie de acciones que podríamos resumir en:

- Identificarse.
- Cargar la lista de plantillas disponibles
- Realizar operaciones CRUD sobre dichas plantillas.
- En aquellos casos en que se estime oportuno, participar en la resolución de conflictos de sincronización.

Por otra parte, el usuario de *back-end*, que podemos considerar como el administrador de la solución, realizará las operaciones equivalentes, pero sobre las entidades y metadatos. Estas funciones realmente quedaran fuera del ámbito de este proyecto, puesto que bastará con que la aplicación móvil reciba metadatos y sea capaz de tratarlos, sin importar como se hayan generado.

2.1.1.1 Fichas de usuario

Hay que diferenciar claramente los dos entornos que componen esta solución, por una parte, en el back-end (fuera del ámbito) habrá un desarrollador de software construyendo los servicios y metadatos dado que no se ha contemplado por el momento disponer de una herramienta para ello.

Por otra parte, en la aplicación móvil final tendremos diferentes perfiles de usuario, por ejemplo:

- Personal de planta de una fábrica que registra información acerca de órdenes de trabajo o consulta listados de dichas órdenes.
- Personal de captura de datos de imagen de marca en supermercados (registro de posición de los productos de la marca a nivel de altura en los expositores, metros de exposición, etc.).
- Encuestadores a pie de calle realizando sondeos.
- …

Como se puede ver la idea puede cubrir muchas posibilidades, y aún se podrían cubrir más si se considera como un módulo que podría incorporarse a otros desarrollos a base de permitir lanzar *activities* como *intents* (no es el objetivo actual, pero seguramente sería un modo de integración muy interesante).

Tipo de usuario	Personal de planta
Necesidades	Consultar fichas de artículos Registrar información de las Ordenes de trabajo que realiza.
Que espera de la aplicación	La consulta en línea de la documentación garantiza que siempre dispone de la versión correcta. El registro de las órdenes de trabajo directamente en el sistema evita una introducción doble, así como la pérdida de información y/o el retraso en el registro.

Tipo de usuario	Personal auxiliar de marca
Necesidades	Realizar encuestas Registrar información de presencia de marca en calle, grandes superficies, …
Que espera de la aplicación	En este caso es especialmente importante la facilidad de distribuir nuevas encuestas o formatos, puesto que las necesidades de registro de un día a otro varían. Generalmente este tipo de usuario trabaja para empresas de control que dan servicio a múltiples marcas, y cada

	una de ellas puede solicitar información diferente.

2.1.2 Contextos de uso

Nuevamente nos encontramos con una cuestión de difícil respuesta, eventualmente el producto debería ser utilizable en cualquier contexto, entendiendo "cualquier contexto" como:

- Entornos con conectividad indefinida, de modo que debe ser capaz de almacenar información local para comunicar con el servidor cuando sea posible.
- Ubicaciones no fijas, de hecho, puede ser necesario tratar la ubicación como un dato a registrar, de modo que deberá tratar geolocalización.
- En relación con el punto anterior hasta se podría dar el caso de querer limitar la comunicación en el caso de redes de pago.

2.2 Diseño conceptual

2.2.1 Escenarios de uso

Para poder ver las posibilidades del concepto planteo los siguientes escenarios, correspondientes a los 3 casos mencionados en las fichas de usuario:

- Personal de planta consultando ordenes de trabajo y completando información sobre estas.
- Personal de imagen de marca registrando KPIs de la presencia de la marca en una gran superficie.
- Encuestador a pie de calle registrando encuestas.

Se ha intentado atomizar al máximo los escenarios de uso debido a que esta aplicación permite realizar acciones muy concretas, difícilmente permitirá cubrir escenarios complejos.

2.2.1.1 Consulta de la planificación de la jornada.

El primer escenario permite que un operario de una planta de producción consulte su lista de órdenes de trabajo para la jornada y acceda al detalle de una orden de trabajo en particular.

2.2.1.2 Registro de KPIs de imagen de marca

Un agente de una marca de leche se desplaza a un establecimiento de una gran superficie y registra la cantidad de marcas de leche que ofrece el establecimiento, el porcentaje de superficie de venta asignado a la marca sobre el total de superficie de lácteos, la altura mínima y máxima desde el suelo a que se encuentran los productos de la marca, y otros valores.

2.2.1.3 Registro de una encuesta

Un entrevistador a pie de calle inicia su jornada conectándose al entorno que le permite descargarse un nuevo formulario de encuesta con el que entrevistar a transeúntes, el sistema puede incluso indicarle unas instrucciones (para indicar el rango de edades o perfil demográfico a localizar) y registrará su ubicación en el momento de realizar la encuesta.

2.2.1.4 Resolución de conflictos

En cualquiera de los escenarios anteriores, al comunicar con el servidor de *back-end* puede que este informe de un conflicto, este conflicto puede ser de varios tipos (que el servidor identificará) y puede soportar una serie de acciones correctivas (que también recibiremos), entre las cuales se pueden encontrar: corregir información, descartar el registro del servidor, descartar el registro local, reintentar.

2.2.2 Flujos de interacción

Las interacciones con la aplicación se consideran "hacia adelante" ya que el propio entorno Android permite cancelar mediante la pulsación del botón (físico o lógico) de "back".

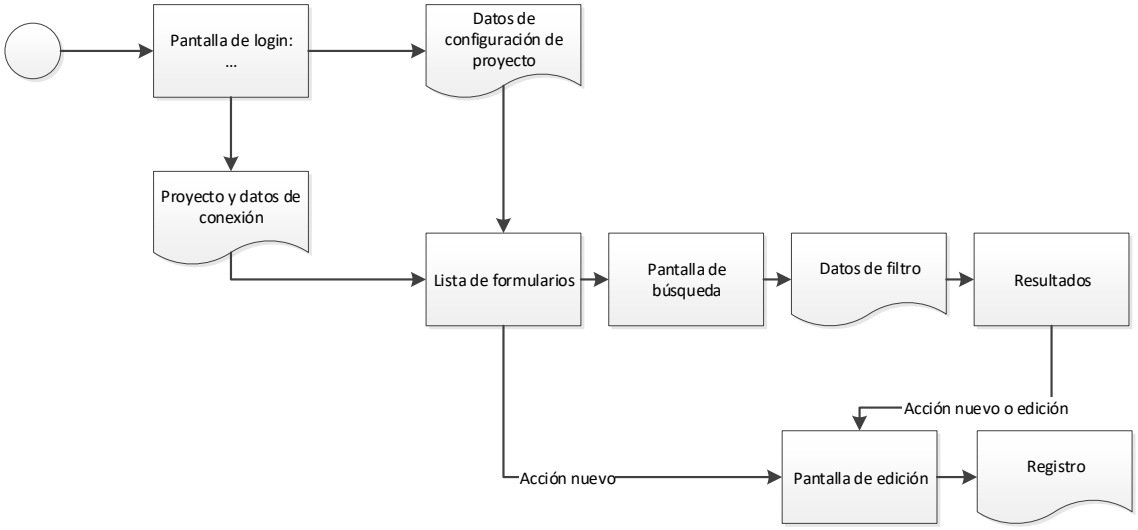
Nuevamente nos encontramos con casos muy genéricos debido a que gran parte del entorno (la parte que nos interesa) será generada de manera dinámica.

Los flujos que se indican a continuación son muy simples, pero corresponden a la interacción básica.

Se incluye un esquema de navegación general para mostrar cómo trabaja la aplicación en conjunto, pero existen otros caminos, navegando desde el menú lateral o bien en el caso de que los formularios no soporten búsqueda o edición, es por ello que a continuación se detallan flujos parciales específicos de cada una de las acciones a realizar.

En general la aplicación en sí es muy lineal: acceder, seleccionar formulario, buscar, editar.

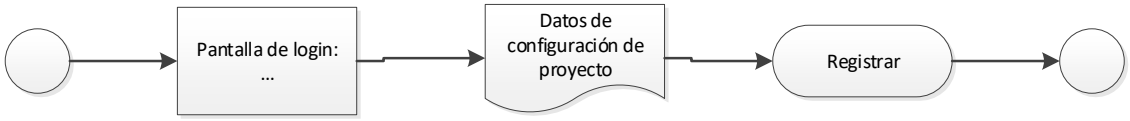
Pero debido a que se puede determinar que los formularios no soporten búsquedas, que se puede tratar de datos remotos en lugar de locales, que puede hacer formularios que no permitan edición, etc. La cantidad de caminos posibles crece de manera considerable. En todos los casos eso sí, existe la posibilidad de volver al menú lateral y desde ahí empezar de cero.



2.2.2.1 Registrar una conexión a un nuevo servidor

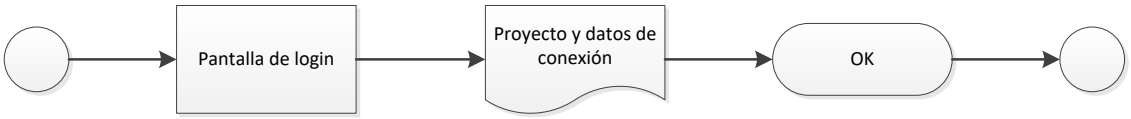
En la pantalla inicial se pulsan los "…" junto al nombre de proyecto con lo que se accede a la pantalla de configuración del proyecto.

La configuración del proyecto puede indicar que el usuario y *password* se guarden para futuras conexiones.



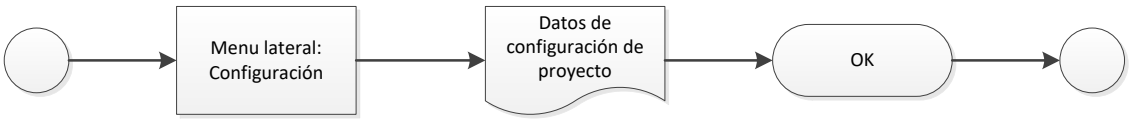
2.2.2.2 *Acceder a un proyecto*

Desde la pantalla inicial se selecciona el proyecto y si este requiere usuario y *password* se completan.



2.2.2.3 *Modificar la configuración de un proyecto*

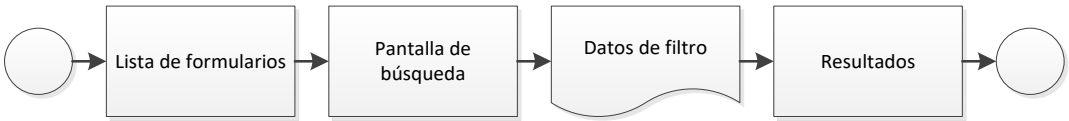
El menú lateral permite acceder a la pantalla de configuración que es la misma que en la configuración inicial.



2.2.2.4 *Consulta de datos*

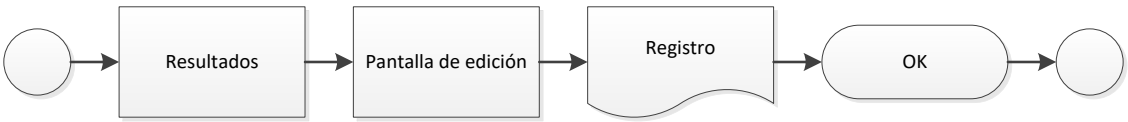
Una vez conectado a un proyecto se accede a la lista de formularios disponibles (es la opción por defecto al conectarse o bien es accesible desde el menú lateral).

Desde la lista de formularios se selecciona uno que nos lleva a la pantalla de búsqueda, esta pantalla se genera desde metadatos y permite acceder a la pantalla de resultados que también es generada desde metadatos.



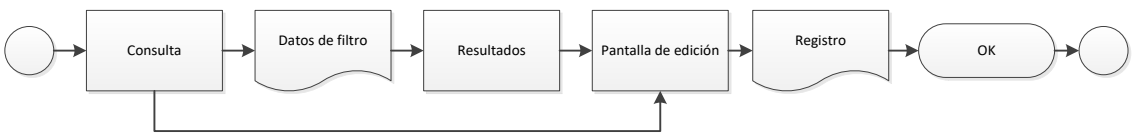
2.2.2.5 *Modificar un registro*

Desde la pantalla de resultados (punto 2.2.2.4), si los metadatos lo permiten se puede acceder a la pantalla de edición también generada desde metadatos.



2.2.2.6 *Editar un registro*

En este caso desde la pantalla de consulta o de resultados se accede a la opción “nuevo” que lleva a la pantalla de edición. El flujo es idéntico al anterior.



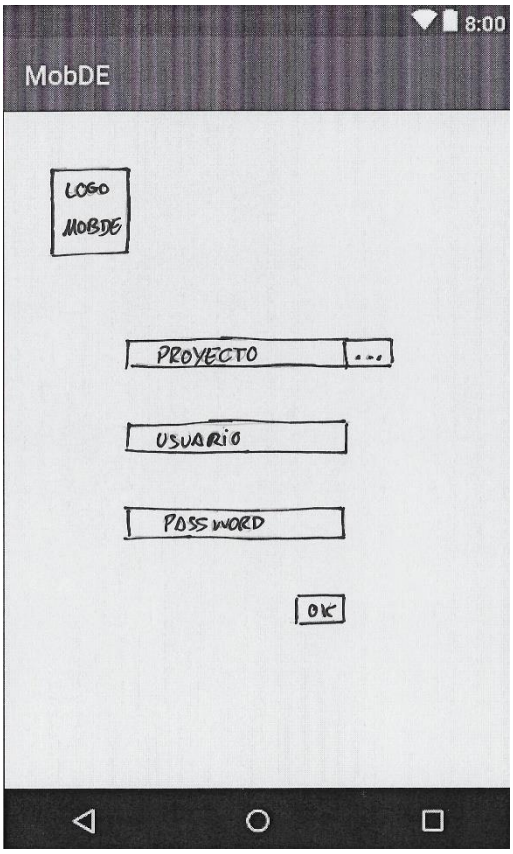
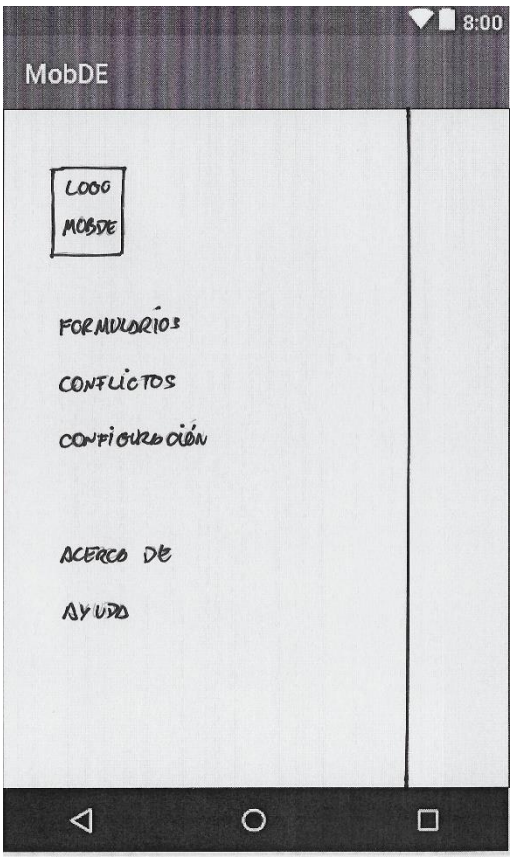
2.3 Prototipaje

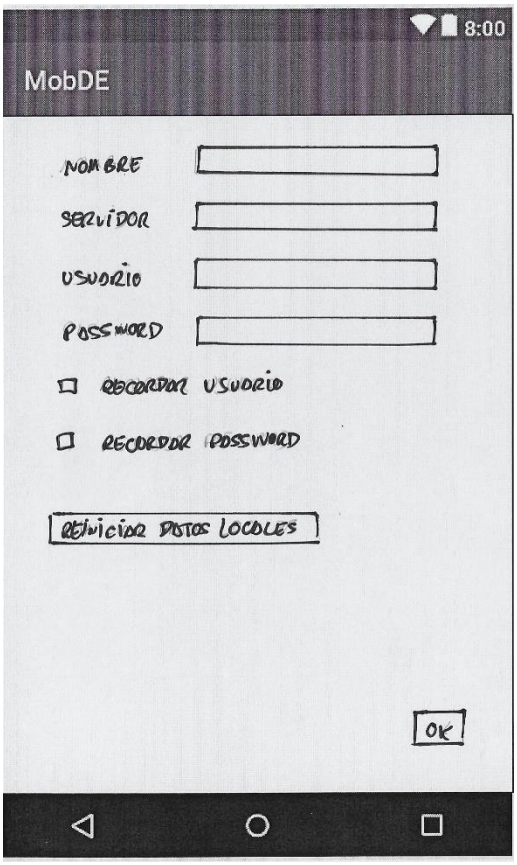
2.3.1 Sketches iniciales

Para la realización de los dibujos a mano alzada se ha partido de una captura de pantalla del modo de diseño de Android Studio de una actividad en blanco con el nombre de la aplicación.

El segundo prototipo ya ha sido realizado con una herramienta de prototipaje, en este caso Justinmind[5].

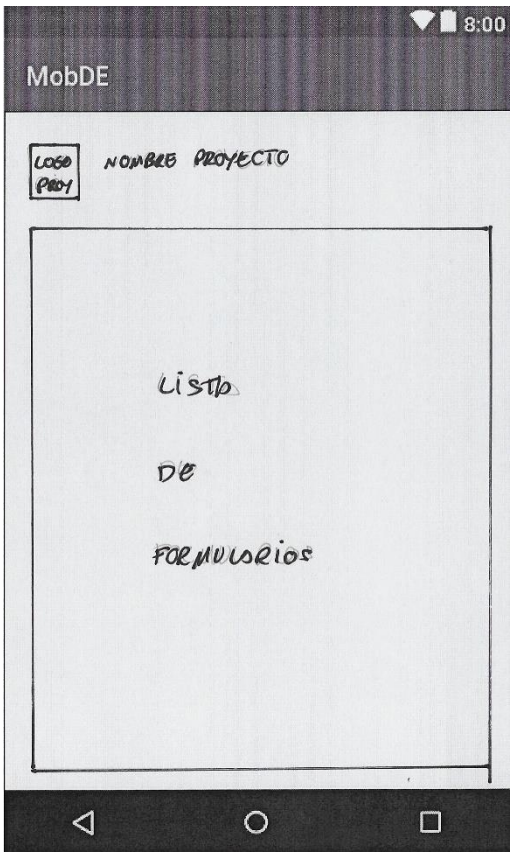
[5] https://www.justinmind.com/

	**Pantalla de login.** Incluye un selector de proyectos debido a que la misma aplicación móvil podrá conectarse a n proyectos de n servidores distintos. Hay un acceso directo a la pantalla de configuración de proyectos para el primer acceso o para modificar la configuración de un proyecto directamente sin llegar a conectar. En el caso de seleccionar un proyecto existente se consulta la configuración para solicitar usuario y password o recuperarlos de los guardados.
	**Menú** Al acceder a la aplicación ya estamos en un proyecto determinado (conectados a un *back-end*), el menú permite acceder a la lista de formularios, la resolución de conflictos y a la pantalla de configuración. También dispone de acceso a la ayuda de la aplicación y a información sobre la misma (acerca de).



Configuración de proyecto

Esta pantalla permite establecer la información de conexión a un *back-end* así como reiniciar (vaciar) las tablas locales de dicho *back-end*.

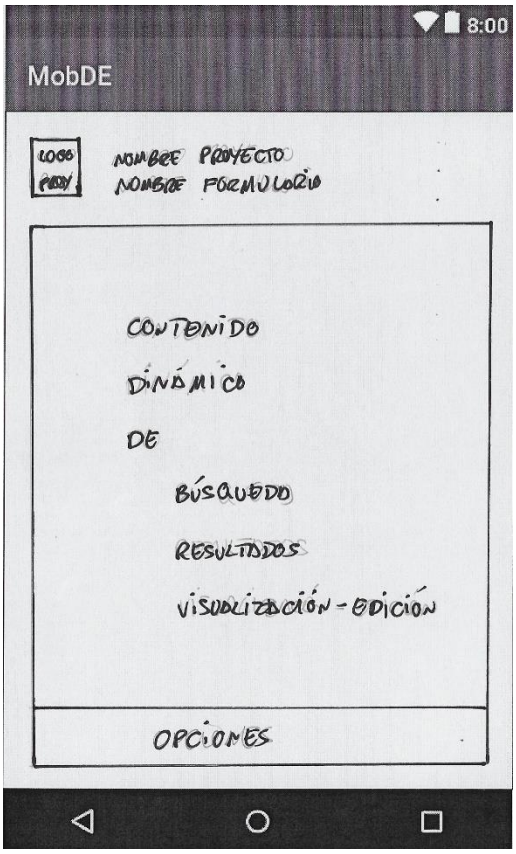


Listado de formularios

A menos que el proyecto conste de un único formulario (en cuyo caso siempre se accede a la pantalla de búsqueda o de edición según corresponda) esta es la pantalla de entrada a la aplicación, desde donde se accede a los diferentes formularios disponibles, sustituye a lo que sería un menú en una aplicación convencional.

Cada formulario realmente consta de 3 vistas:

- Búsqueda
- Resultados
- Edición

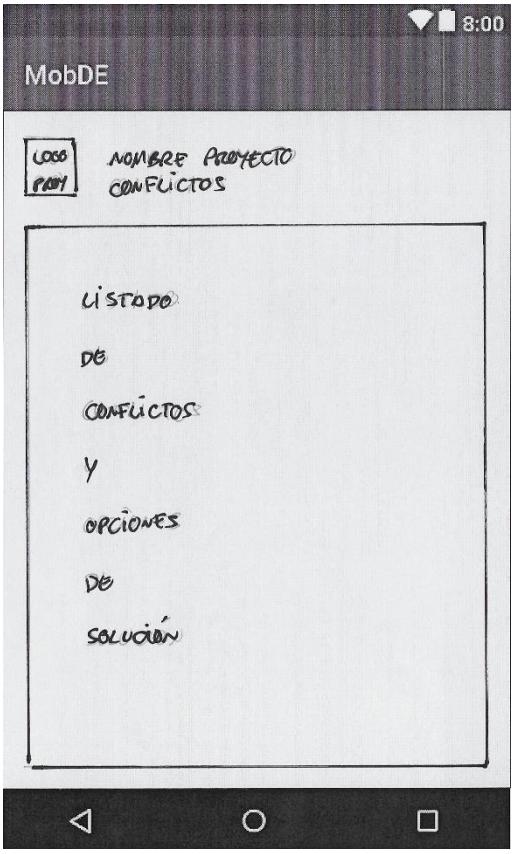


Formulario

En este caso ya estamos accediendo a contenido autogenerado, de modo que el sketch no entra en más detalle.

Como se indica en el punto anterior, se dispone de 3 vistas que se generan desde metadatos.

- Búsqueda, para establecer un filtro.
- Resultados, que muestra los elementos seleccionados en la búsqueda y que, según indiquen los metadatos permitirá acceder a las operaciones CRUD.
- Edición, esta pantalla corresponde al detalle de un elemento y los metadatos establecerán que operaciones son disponibles, así como las validaciones a realizar.



Conflictos

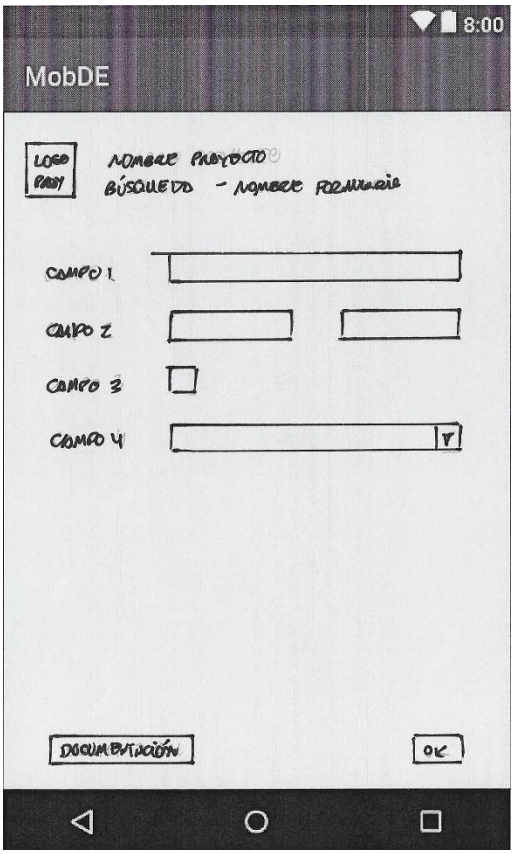
La pantalla de conflictos muestra la lista de conflictos de sincronización y las opciones disponibles para cada uno, por ejemplo:

- Editar
- Descartar
- Forzar (sobrescribe el registro de la base de datos central)

2.3.2 Sketches ejemplo del contenido dinámico

En este caso simplemente son ideas de posibles pantallas que se podrían crear dinámicamente, no pertenecen al *core* de la aplicación sino al contenido que esta es capaz de generan bajo demanda.

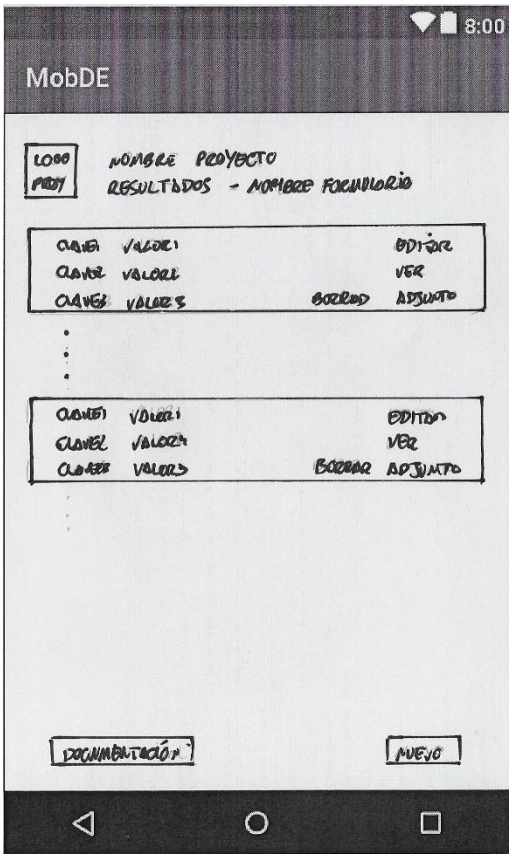
Todas las pantallas dinámicas pueden tener además un acceso a una documentación (posiblemente pdf) que informa al usuario acerca de cómo debe consultar o completar la información.



Criterios de búsqueda

Esta pantalla se construye en base a metadatos y permite consultar la base de datos y muestra el resultado en la pantalla de resultados.

Desde esta pantalla se accede también a un manual (si se dispone de él) y se puede crear un registro accediendo directamente a la pantalla de edición.

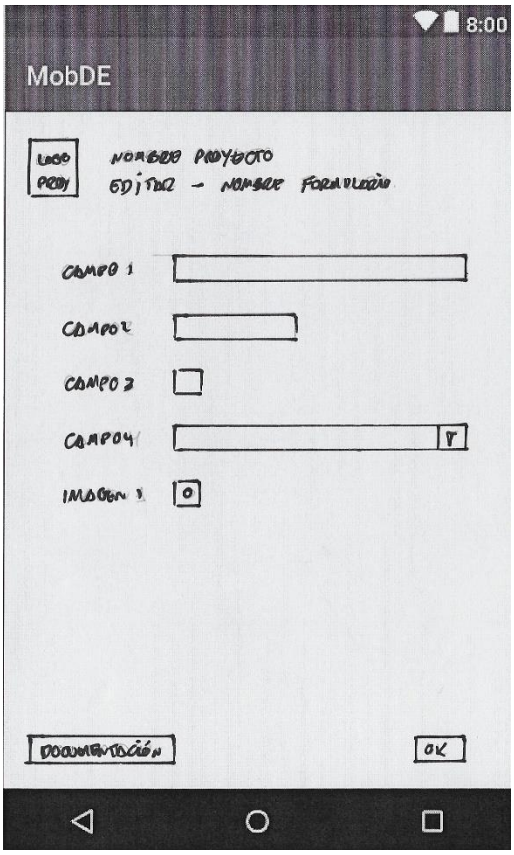


Resultados

Esta pantalla también se genera de modo dinámico, cada registro localizado muestra las opciones disponibles para este, por ejemplo:

- Editar
- Ver
- Borrar
- Consultar documentación adjunta (así puede servir no solo para *data entry* sino también para consulta).

También se dispone de accesos directos a la documentación si está disponible y a crear un registro.



Edición

La última de las pantallas creadas por metadatos, corresponde a la pantalla de detalle, tanto para edición como consulta.

Nuevamente dispone del acceso a la documentación si está disponible.

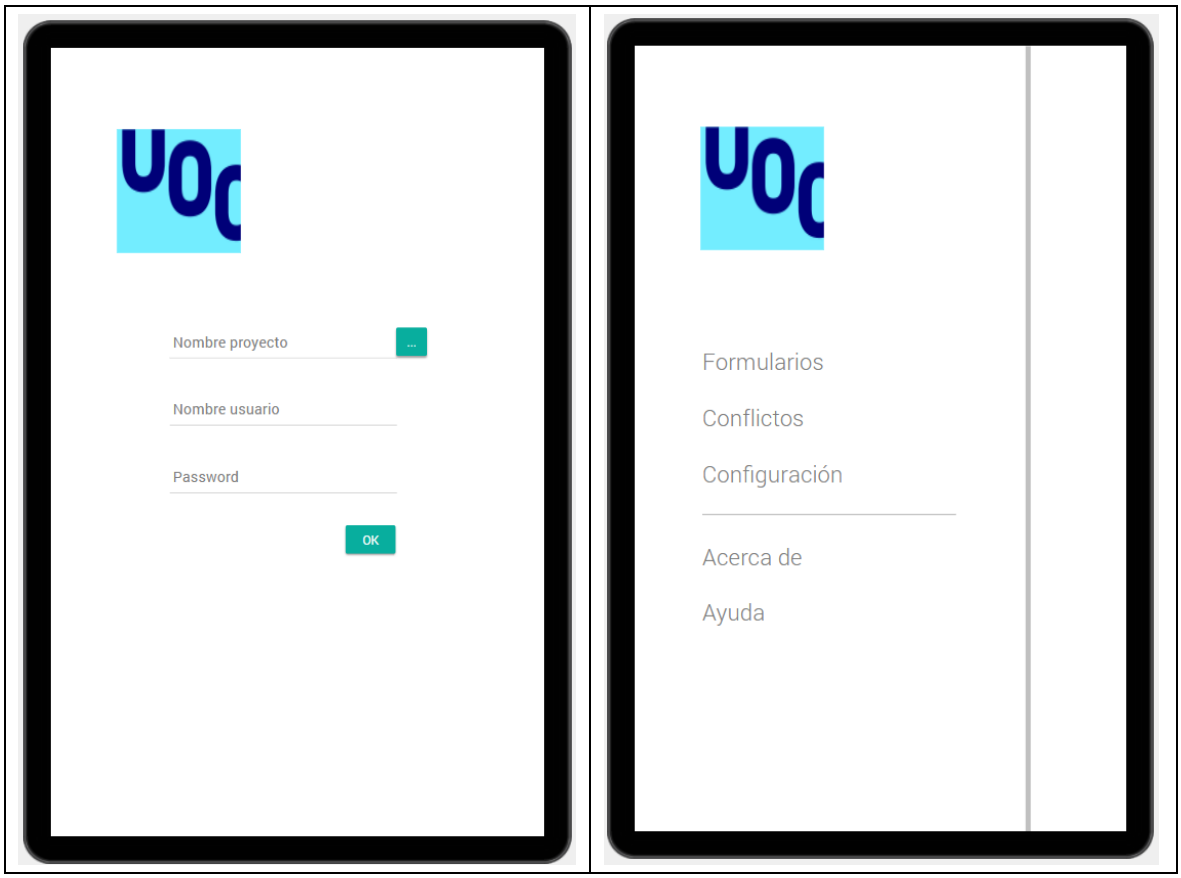
2.3.3 Prototipo de alta fidelidad

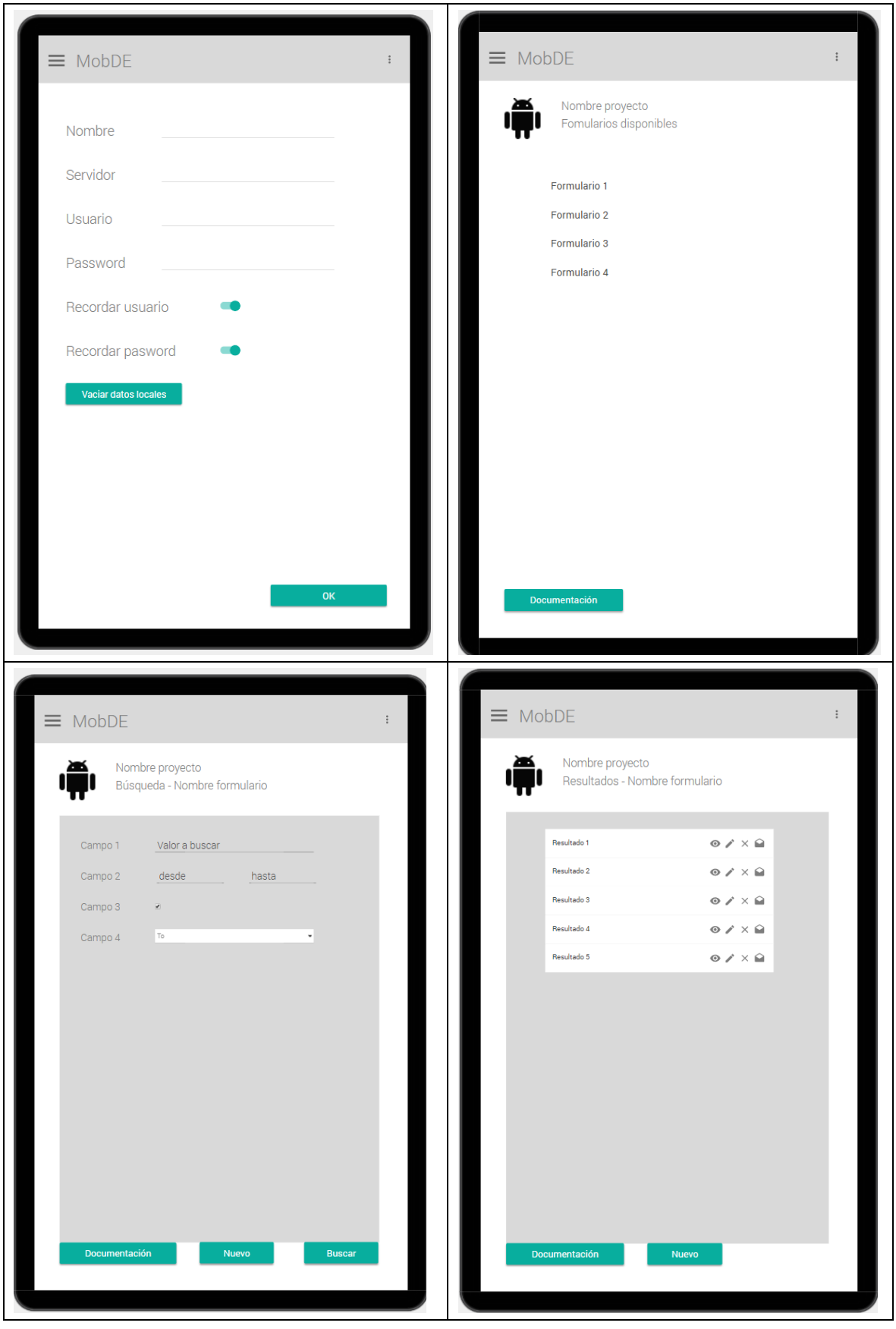
A continuación, se adjuntan imágenes de las pantallas realizadas utilizando Justinmind como herramienta de prototipado.

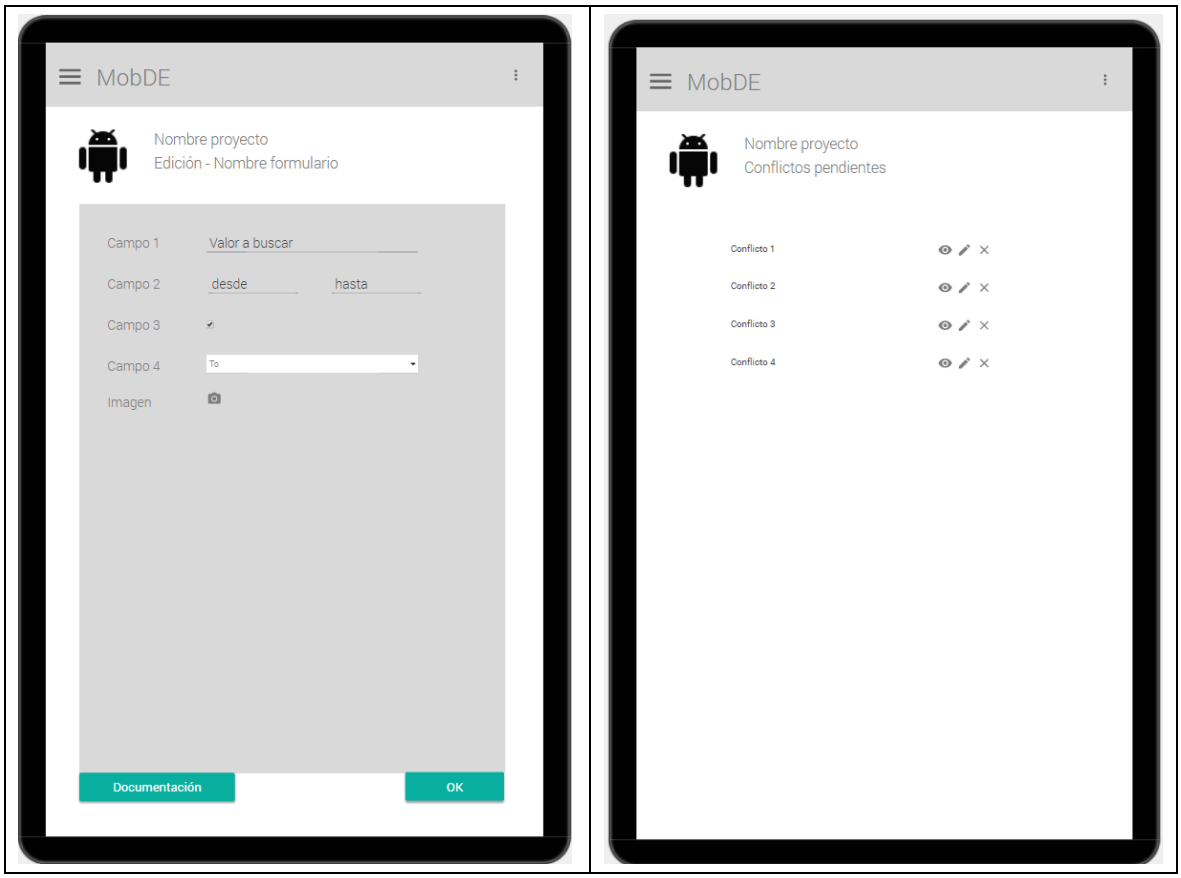
En este caso se ha incluido tanto el contenido estándar de la aplicación como el dinámico de ejemplo.

Se ha procurado que el diseño de la aplicación sea lo más simple posible, sin ornamentos innecesarios, un diseño muy lineal para que sea lo más claro y directo posible.

La única excepción a esta norma es que todas las pantallas incluyen siempre un logotipo del proyecto y un recordatorio textual del proyecto y pantalla en que se encuentra (dado que la misma aplicación da acceso a múltiples entornos y formularios con un aspecto uniforme).




MobDE
Nombre
Servidor
Usuario
Password
Recordar usuario
Recordar pasword
Vaciar datos locales
OK
MobDE
Nombre proyecto
Fomularios disponibles
Formulario 1
Formulario 2
Formulario 3
Formulario 4
Documentación
MobDE
Nombre proyecto
Búsqueda - Nombre formulario
Campo 1
Valor a buscar
Campo 2
desde
hasta
Campo 3
Campo 4
To
Documentación
Nuevo
Buscar
MobDE
Nombre proyecto
Resultados - Nombre formulario
Resultado 1
Resultado 2
Resultado 3
Resultado 4
Resultado 5
Documentación
Nuevo



2.4 Evaluación

Utilizando el prototipo creado con Justinmind y sus enlaces entre pantallas se ha podido comprobar que toda la aplicación es accesible y que la navegación entre pantallas es coherente.

Cabe tener en cuenta que algunos caminos entre pantallas (del apartado dinámico) pueden llegar a desaparecer según metadatos, por ejemplo:

- Si el entorno no tiene más que un formulario, la lista de formularios se obvia, en ese caso la primera pantalla pasa a ser la de búsqueda, resultados o edición según corresponda.
- Si el entorno así lo establece, la pantalla de búsqueda no existe (por ejemplo, en el caso de procesar siempre un número de registros muy limitado), en este caso la pantalla de resultados o la de edición pasan a ser primarias.
- En algunos casos se puede establecer que no se puedan consultar resultados pasados o, por el contrario, que no se puedan crear registros sino únicamente consultar o modificar.
- Todas estas casuísticas se recibirán mediante metadatos del “formulario” pero son tremendamente complejas de probar en un prototipo.

3 Diseño

3.1 Diseño técnico

3.1.1 Definición de casos de uso

Los casos de uso se han atomizado al máximo, las secuencias de utilización consistirán en ir enlazando casos.

Identificador	**CU-001**
Nombre	Registro
Prioridad	Alta
Descripción	Registro de los datos de un proyecto.
Actores	Usuario
Precondiciones	Aplicación instalada
Iniciado por	Usuario
Flujo	Al ejecutar la aplicación y pulsar el botón "…" de configuración, el sistema redirige a la pantalla de configuración de proyectos con todos los campos en blanco (CU-002)
Postcondiciones	Proyecto registrado en la base de datos local
Notas	Esta es la única opción disponible cuando no se ha registrado ningún proyecto y es una acción opcional cuando ya hay proyectos registrados.

Identificador	**CU-002**
Nombre	Configuración de proyecto
Prioridad	Alta
Descripción	Configuración de los datos de acceso a un back-end
Actores	Usuario
Precondiciones	Aplicación instalada
Iniciado por	Usuario
Flujo	La pantalla muestra la información disponible del proyecto, el usuario completa los campos de acceso a un back-end.
Postcondiciones	Proyecto registrado en la base de datos local.
Notas	El flujo es el mismo tanto si se trata de un proyecto nuevo o una reconfiguración de uno preexistente únicamente cambia que se cargan datos iniciales o no. En los diseños de pantalla no se ha tenido en cuenta pero podría ser interesante añadir un botón de "Test" de la configuración.

Identificador	**CU-003**
Nombre	Limpiar datos de un proyecto
Prioridad	Baja
Descripción	Permite vaciar la base de datos local de un proyecto
Actores	Usuario
Precondiciones	Proyecto configurado
Iniciado por	Usuario
Flujo	A través de la pantalla de configuración de un proyecto (CU-002) el usuario pulsa el botón “Reiniciar datos locales” y confirma el diálogo que aparece.
Postcondiciones	Base de datos local vacía.
Notas	Esta función es de baja prioridad ya que realmente no es crítica ni mucho menos, simplemente permite eliminar registros antiguos.

Identificador	**CU-004**
Nombre	Consulta lista de formularios disponibles
Prioridad	Alta
Descripción	Muestra los formularios asociados a un proyecto
Actores	Usuario
Precondiciones	Proyecto configurado, datos de proyecto descargados (CU-005)
Iniciado por	Usuario
Flujo	Se accede a esta lista al iniciar la aplicación o mediante el menú lateral.
Postcondiciones	Se muestra la lista de formularios
Notas	No se trata de un caso de uso como tal, solo es un fragmento, pero permite enlazar con otros fragmentos.

Identificador	**CU-005**
Nombre	Descarga de configuración
Prioridad	Alta
Descripción	Conecta con el *back-end* utilizando la configuración establecida y descarga la información disponible (formularios, tablas auxiliares, etc.)
Actores	Sistema
Precondiciones	Proyecto configurado
Iniciado por	Usuario - Sistema
Flujo	Esta tarea se inicia al grabar la configuración de un proyecto, así como cada vez que se realiza la conexión.
Postcondiciones	Información de proyecto actualizada.
Notas	Cabe la posibilidad de que algunos proyectos definan un intervalo de actualización de la configuración más frecuente.

Identificador	**CU-006**
Nombre	Consulta documentación proyecto
Prioridad	Baja
Descripción	En el caso que el proyecto indique que tiene un manual de uso, permite acceder a dicho manual
Actores	Usuario
Precondiciones	Proyecto configurado Proyecto con manual de uso Conexión a internet si el documento no ha sido descargado ya.
Iniciado por	Usuario
Flujo	Este caso está disponible desde diversos puntos - Listado de formularios - Pantalla de búsqueda - Pantalla de resultados - Pantalla de edición El sistema descarga el documento si aún no ha sido descargado y lo muestra en pantalla.
Postcondiciones	Muestra la documentación del proyecto.
Notas	

Identificador	**CU-007**
Nombre	Búsqueda de registros
Prioridad	Alta
Descripción	El sistema crea dinámicamente una pantalla de criterios de búsqueda y permite ejecutar la búsqueda.
Actores	Usuario
Precondiciones	Proyecto configurado Formulario seleccionado Formulario con búsqueda habilitada (ver notas)
Iniciado por	Usuario – Sistema
Flujo	El sistema consulta los metadatos para construir la pantalla de consulta, el usuario completa los datos y pulsa el botón “búsqueda”
Postcondiciones	El sistema redirige a resultados
Notas	Normalmente se accede a este caso desde la pantalla de lista de formularios, pero se puede llegar a ella directamente en el caso que el proyecto solo conste de un formulario.

Identificador	**CU-008**
Nombre	Lista de resultados
Prioridad	Alta
Descripción	El sistema presenta la lista de registros que cumplen las condiciones de búsqueda (ver notas) y presenta las acciones disponibles para cada registro
Actores	Usuario
Precondiciones	Proyecto configurado Conexión a base de datos del proyecto
Iniciado por	Usuario - Sistema
Flujo	El sistema consulta los metadatos para construir la lista de resultados, así como las acciones disponibles para cada registro que cumple las condiciones. Por otra parte, se consultan los datos para obtener los registros que cumplen los requisitos y se pintan según los metadatos. El usuario tiene disponibles las acciones a realizar sobre cada registro, por ejemplo: - Ver (CU-009) - Editar (CU-010) - Eliminar (CU-011)

	- Consultar documentación adjunta (CU-012)
Postcondiciones	La lista de resultados se presenta en pantalla.
Notas	Normalmente se accede a este caso desde CU-007 pero se puede llegar directamente desde CU-004 en el caso de que el formulario no soporte búsqueda (en el caso más extremo se llega aquí desde el login si solo hay un formulario y dicho formulario no soporta búsqueda), en estos casos la lista carga toda la información disponible.

Identificador	**CU-009**
Nombre	Ver
Prioridad	Media
Descripción	El sistema consulta los metadatos para construir la pantalla de detalle de un registro. Por otra parte, se consultan los datos para obtener el detalle del registro a mostrar.
Actores	Usuario
Precondiciones	Lista de resultados cargada
Iniciado por	Usuario
Flujo	Desde la lista de resultados se pulsa el icono de “ver”. El sistema lanza el proceso de sincronización (CU-013)
Postcondiciones	Se muestra la información de un registro en pantalla.
Notas	Los casos CU-009, CU-010 y CU-014 comparten la mayor parte de funcionalidad, cargando o no datos y actualizando o no la base de datos, inicialmente se consideraba CU-014 parte de CU-010.

Identificador	**CU-010**
Nombre	Editar
Prioridad	Alta
Descripción	El sistema consulta los metadatos para construir la pantalla de detalle de un registro. Por otra parte, se consultan los datos para obtener el detalle del registro a mostrar.
Actores	Usuario
Precondiciones	Lista de resultados cargada Metadatos del formulario autorizan la edición.
Iniciado por	Usuario
Flujo	Desde la lista de resultados se pulsa el icono de “editar”.

	El sistema lanza el proceso de sincronización (CU-013)
Postcondiciones	Base de datos local actualizada
Notas	Los casos CU-009, CU-010 y CU-014 comparten la mayor parte de funcionalidad, cargando o no datos y actualizando o no la base de datos, inicialmente se consideraba CU-014 parte de CU-010. Hay un modo alternativo de llegar a este punto y es desde la lista de conflictos.

Identificador	**CU-011**
Nombre	Eliminar
Prioridad	Media
Descripción	Permite eliminar un registro de la base de datos local.
Actores	Usuario
Precondiciones	Lista de resultados cargada Metadatos del formulario autorizan la eliminación.
Iniciado por	Usuario
Flujo	Desde la lista de resultados se pulsa el icono de “eliminar”. El sistema lanza el proceso de sincronización (CU-013)
Postcondiciones	Base de datos local actualizada
Notas	Hay un modo alternativo de llegar a este punto y es desde la lista de conflictos.

Identificador	**CU-012**
Nombre	Consultar documentación adjunta
Prioridad	Baja
Descripción	
Actores	Usuario
Precondiciones	Lista de resultados cargada Metadatos del formulario indican que hay adjuntos.
Iniciado por	Usuario
Flujo	Desde la lista de resultados se pulsa el icono de consultar documentación.
Postcondiciones	Se muestra la información solicitada en pantalla
Notas	Este caso es ya un “futurible”, se plantea la posibilidad de que determinados formularios no tengan como objetivo editar información sino únicamente consultarla, y en ese caso los registros podrían tener documentación adjunta y

	ser consultable directamente desde la lista de resultados (en cualquier caso también lo sería desde el detalle, por ello es de baja prioridad).

Identificador	**CU-013**
Nombre	Sincronización
Prioridad	Alta
Descripción	
Actores	Sistema
Precondiciones	
Iniciado por	Sistema
Flujo	
Postcondiciones	
Notas	

Identificador	**CU-014**
Nombre	Nuevo
Prioridad	Alta
Descripción	Permite crear un nuevo registro en la base de datos local.
Actores	Usuario
Precondiciones	Proyecto configurado Conexión a base de datos del proyecto Metadatos del formulario autorizan la eliminación.
Iniciado por	Usuario
Flujo	El sistema consulta los metadatos para construir la pantalla de detalle de un registro. El usuario completa la información del registro y pulsa el botón de grabación. El sistema lanza el proceso de sincronización (CU-013)
Postcondiciones	Base de datos local actualizada
Notas	Los casos CU-009, CU-010 y CU-014 comparten la mayor parte de funcionalidad, cargando o no datos y actualizando o no la base de datos, inicialmente se consideraba CU-014 parte de CU-010.

Con esto concluye la lista de casos de uso principales, quedan algunos casos auxiliares sin mencionar, pero no tienen mayor importancia. Además, futuras versiones de la aplicación incorporarán nuevos casos de

uso, pero no deberían afectar a este núcleo en torno al cual nace el entorno.

3.2 Diseño de la arquitectura

3.2.1 Diagrama UML de la base de datos

La intención es mantener el diseño de la base de datos lo más simple posible, de hecho, toda la complejidad se almacena en campos de la base de datos que contienen cadenas en formato json, esto permite almacenar estructuras muy complejas en una base de datos muy simple.

Separamos el diseño de la base de datos global o de entorno del resto ya que siempre habrá una base de datos general para el entorno mientras que cada proyecto individual tendrá su base de datos propia (descargada desde el servidor como metadatos).

3.2.1.1 UML de la base de datos del entorno

Como se ha comentado en el punto anterior, la intención es mantener la base de datos lo más simple posible.

Existe una tabla de configuración de proyectos y una tabla para los formularios de dichos proyectos. El resto de información dinámica se almacena en json como atributos de la base de datos.

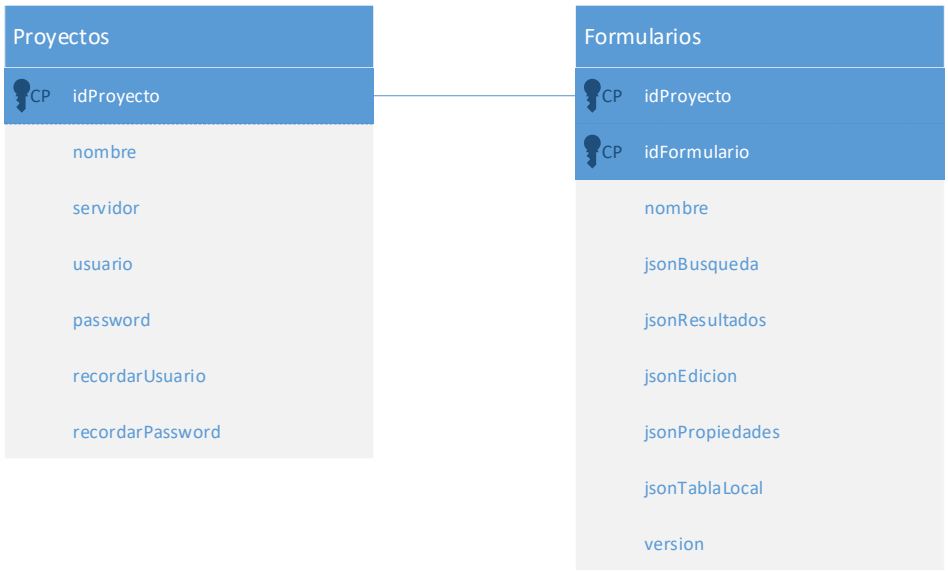


Ilustración 1. UML base entorno

3.2.1.2 UML de la base de datos de los proyectos

En este caso el diagrama no está totalmente definido dado que las bases de datos se crearán bajo demanda, pero siempre considero dos tipos de tablas:

- Datos, estas tablas corresponden a los formularios en sí y tendrán siempre unos atributos fijos además de los que reciban por los metadatos puesto que son necesarios para la gestión de sincronización y conflictos (ver ilustración).
- Datos auxiliares, son las tablas que permitirán llenar listas de selección y similares, en la versión inicial se limitan a únicamente un id y una descripción, pero puede haber tantas como sea necesario.

Ilustración 2. UML base proyecto

3.2.2 Diagrama UML de entidades y clases

Nuevamente hay que diferenciar entre el entorno y los proyectos, para el entorno, solo existe una entidad disponible, que corresponde a proyectos, todo el resto de las entidades corresponden ya al entorno de proyectos.

Por su parte en cada proyecto tenemos las entidades: formularios, registros, conflictos, documentos e imágenes.

Cada una de estas entidades requiere de una clase para gestionarla además de algunas clases auxiliares como el inflador de vistas y el gestor de geolocalización.

A grandes rasgos la estructura de entidades es la siguiente:

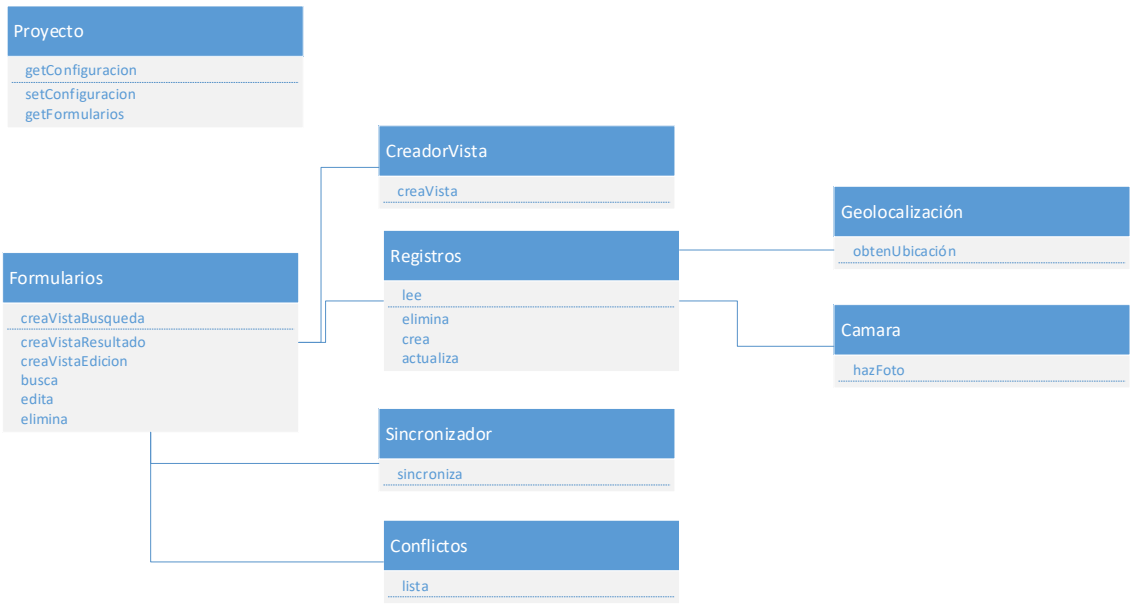


Ilustración 3. UML entidades

3.2.3 Diagrama de arquitectura MVC

Este punto es complejo debido a que el núcleo de la aplicación se genera de modo dinámico en base a unos metadatos.

Considerar que el modelo de implementación de esta aplicación es MVC implica estirar un poco el concepto, pero es asumible si entendemos que solo tenemos un modelo, una vista y un controlador y que los tres se apoyan en una clase auxiliar de metadatos que los complementa, esto implica que aunque no tenemos un modelo, una vista y un controlador como tales, sí que tenemos una clase que actúa como modelo, una clase que actúa como vista y una clase que actúa como controlador.

Por otra parte, lo que realmente establecerá que sea una implementación MVC será el modo en que se dividan las competencias entre las clases, más que el hecho de que todas ellas sean genéricas.

Dada esta genericidad de las clases, el diagrama que se presenta pretende hacer hincapié en ese punto, por ello solo se consideran 3 elementos, precisamente modelo, vista, controlador, todos ellos genéricos (hay una parte de la aplicación, correspondiente al entorno que no utiliza genéricos, pero es la parte menos importante de la aplicación).

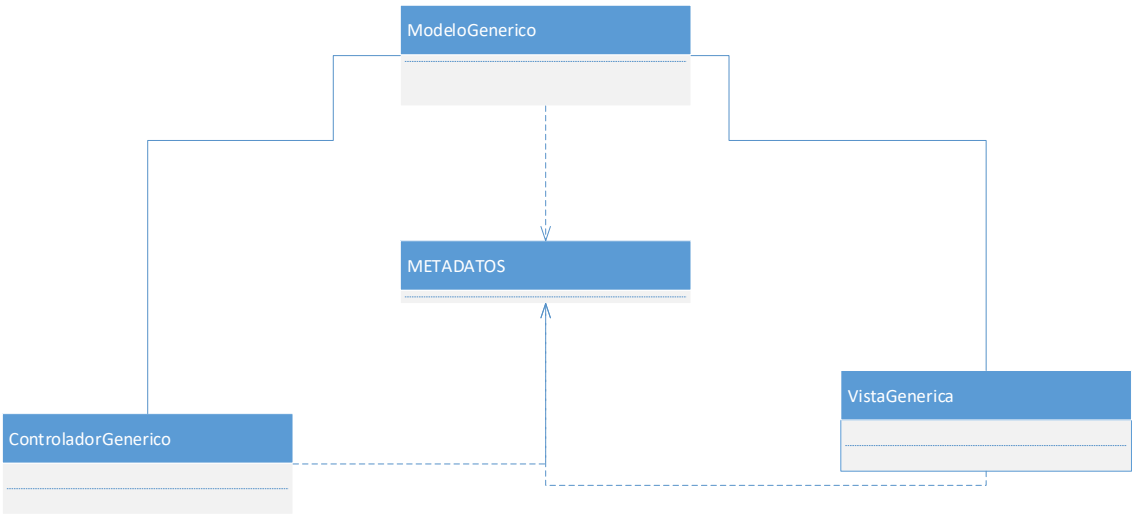


Ilustración 4. MVC genérico

En el momento de la implementación encontraremos con que los tres componentes se alimentaran de los metadatos para realizar sus acciones, así la lista de propiedades que expondrá el modelo será una colección de parejas clave-valor dinámica en lugar de algo fijo, las acciones que el controlador podrá solicitar estarán determinadas no por el código de la aplicación sino por una serie de banderas en los metadatos y la visualización que nos presentará la vista utilizará una clase generadora de la vista y unos metadatos para poder construir dicha presentación, incluyendo los enlaces con las acciones del controlador.

3.2.4 Comunicación con el back-end

Toda la comunicación entre back-end y front-end se realiza a través de una cantidad muy limitada de servicios, utilizando la autenticación básica según RFC-2617 (Network Working Group, s.f.).

3.2.4.1 Lista de métodos del back-end

Metodo	**Tipo de entrada**	**Tipo de salida**
Login	Cabecera de la llamada según RFC-2617	LoginResult
Formularios	Cabecera de la llamada según RFC-2617	FormulariosInfo
Sync	RecordData	UpdateResult
Query	Query	QueryResult
Download	DownloadRequest	DownloadResult

El detalle de las estructuras json implicadas se ve a continuación, cuando el tipo no es un tipo base, su estructura se detalla en su capítulo correspondiente.

Este conjunto de estructuras permite aumentar la complejidad del entorno sin aumentar el número de estructuras, simplemente incorporando nuevos atributos a las estructuras json traspasadas, así, por ejemplo, se podrán incorporar validaciones más complejas a la estructura infoCampo cuando sea necesario.

3.2.4.2 Estructura LoginResult

Elemento	**Descripción**	**Tipo**
codError	Código de retorno del error producido, 0 para éxito	int
token	Token identificador de la sesión, este token se debe enviar en el resto de llamadas para identificar la sesión activa.	String

3.2.4.3 Estructura FormulariosInfo

Elemento	**Descripción**	**Tipo**
formularios	Lista de formularios pertenecientes a este proyecto.	Formulario[]
auxiliares	Lista de tablas auxiliares del proyecto.	Auxiliar[]

3.2.4.4 Estructura RecordData

Elemento	**Descripción**	**Tipo**
tabla	Identificador de la tabla de destino	String
forzar	-1 indica que se desea sobreescribir el registro del back-end como respuesta a una incidencia de sincronización.	Int
registro	Contenido del registro enviado. No tiene una estructura fija sino que es distinta para cada tabla	Campo[]

	puesto que contiene parejas clave-valor para cada campo.	

3.2.4.5 *Estructura UpdateResult*

Elemento	Descripción	Tipo
result	Código de retorno del error producido: 0 para éxito y conservar 1 para éxito y eliminar registro local 2 para problema de sincronización Eventualmente se podrían definir nuevos códigos para otras acciones.	int
sync_flags	Banderas de opciones disponibles para corregir el problema de sincronización	String
sync_error	Explicación del problema de sincronización, corresponde al texto que se presenta al usuario.	String

3.2.4.6 *Estructura Query*

Elemento	Descripción	Tipo
tabla	Tabla sobre la que realizar la consulta	String
condiciones	Lista de filtros a aplicar	Condicion[]

3.2.4.7 *Estructura QueryResult*

Elemento	Descripción	Tipo
registros	Conjunto de resultados	Record[]

3.2.4.8 *Estructura DownloadRequest*

Elemento	Descripción	Tipo
ruta	Ruta del fichero a descargar. En futuras versiones podría ser un id	String

3.2.4.9 *Estructura DownloadResult*

Elemento	Descripción	Tipo
ficheroBase64	Cadena que contiene el fichero codificado en base64.	String

3.2.4.10 *Otras estructuras*

3.2.4.10.1 Estructura Formulario

Elemento	Descripción	Tipo
id	Identificador del formulario	Guid
nombre	Nombre del formulario	String
jsonBusqueda	Json que define la pantalla de búsqueda	Json
jsonResultados	Json que define cada ítem de la pantalla de resultados de búsqueda / incidencias	Json

jsonEdicion	Json que define la pantalla de edición	Json
jsonPropiedades	Json que define otras propiedades del formulario como puede ser si soporta búsqueda o no.	Json
version	Número de versión del formulario, básicamente permite al entorno saber si se trata de un formulario ya conocido o bien si debe reconstruir la tabla local.	int
jsonPropiedadesTabla	Json que define la tabla de datos (ya sea para crear una tabla local o para conocer los tipos de datos para la búsqueda/edición remota.	Json

3.2.4.10.2 Estructura Auxiliar

Elemento	**Descripción**	**Tipo**
nombre	Nombre de la tabla auxiliar, utilizado como referencia	String
Registros	Lista de registros que componen la tabla.	RegAux[]

3.2.4.10.3 Estructura RegAux

Elemento	**Descripción**	**Tipo**
codigo	Identificador utilizado en las relaciones	String
nombre	Texto a mostrar	String

3.2.4.10.4 Estructura jsonBusqueda, jsonEdicion, jsonResultados

Se definen 3 propiedades diferentes para tener mayor control sobre el diseño de las pantallas, pero las 3 siguen la misma estructura.

Esta estructura debe evolucionar con futuras versiones para permitir mayor control del pintado de las pantallas

Elemento	**Descripción**	**Tipo**
campos	Lista de campos	Campo[]

3.2.4.10.5 Estructura campo

Elemento	**Descripción**	**Tipo**
titulo	Título a mostrar	String
campo	Campo relacionado de la tabla de propiedades	String

3.2.4.10.6 Estructura jsonPropiedades

Esta estructura registra las propiedades del formulario.

Elemento	**Descripción**	**Tipo**
permiteBusqueda	Habilita la pantalla de búsqueda	bool
permiteResultados	Habilita la pantalla de resultados	bool
permiteAñadir	Habilita la acción de agregar	bool

permiteEdicion	Habilita la acción de editar	bool
permiteVisualización	Habilita la acción de ver	bool
permiteBorrado	Habilita la acción de eliminar	bool
Remoto	Indica si es una tabla remota o local	bool

3.2.4.10.7 Estructura jsonPropiedadesTabla

Esta estructura permite definir los campos de la tabla, que se utilizan tanto para crear la tabla local en el dispositivo como para establecer las propiedades de los controles.

Elemento	**Descripción**	**Tipo**
Nombre	Nombre de la tabla	String
Campos	Lista de campos que componen la tabla, esta estructura incluye la definición de tipos así como alguna validación (min/max) y relaciones con tablas auxiliares.	InfoCampo

3.2.4.10.8 Estructura infoCampo

Elemento	**Descripción**	**Tipo**
Nombre	Nombre del campo	String
tipo	Tipo de datos: text email phone date time datetime integer pdf El tipo de datos establece si la pantalla de búsqueda define desde-hasta, así como el tipo de entrada que se solicita a Android. En el caso de los tipos de fecha/hora además da acceso a los selectores correspondientes. En el caso de datos remotos, los tipos pdf, phone y email además lanzan acciones (abrir pdf, llamar o escribir correo).	String
aux	Tabla auxiliar (FK)	String
min / max	Valores límite para campos numéricos, fecha, hora.	--

3.2.4.10.9 Estructura condición

Elemento	**Descripción**	**Tipo**
campo	Nombre del campo	String

operador	Operador de la condición (“=”, “>=”, …)	String
valor	Valor a comparar	String

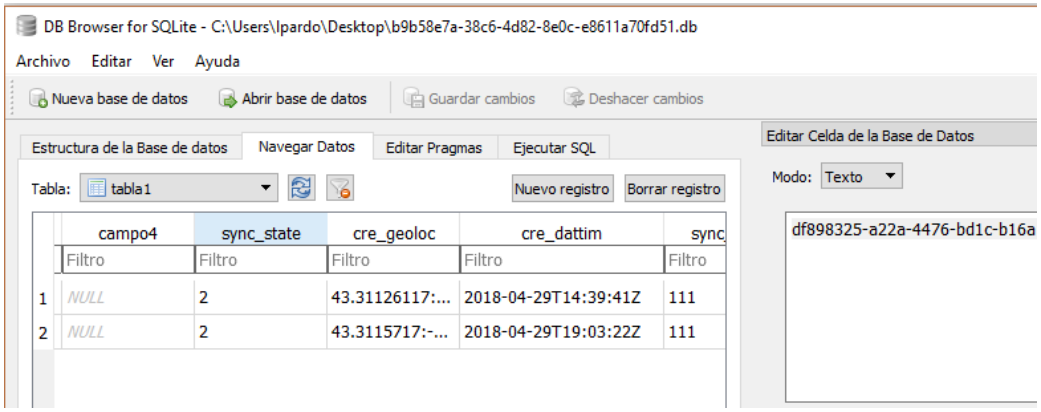
3.2.4.10.10 Estructura record

Elemento	**Descripción**	**Tipo**
campos	Colección de campos del registro.	Campo[]

4 Pruebas

No se han realizado pruebas automatizadas del entorno, sino que se han realizado únicamente pruebas de usuario, de conjunto.

Es evidente que disponer de una batería de pruebas automáticas sería interesante, pero debido a que la mayor complejidad de esta aplicación consiste en la generación del entorno gráfico y su reflejo en una base de datos creada sobre la marcha ha resultado mucho más simple probar los formularios manualmente y copiar las bases de datos del dispositivo sobre el ordenador de desarrollo para revisar el contenido mediante una herramienta de consulta SQLite.



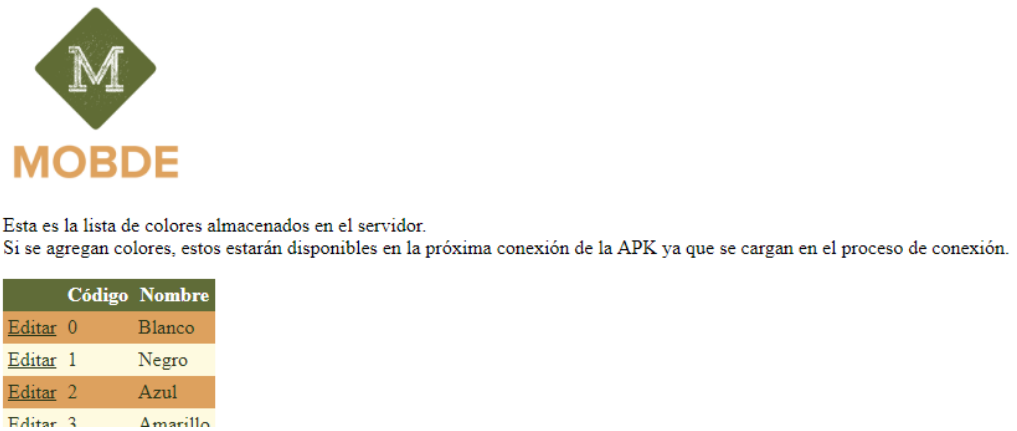
Esta herramienta ha permitido ir analizando en todo momento el contenido volcado a la base de datos del proyecto independientemente de que el flujo de la aplicación ya estuviera funcionando correctamente o no. Esto ha permitido separar la complejidad de las pruebas en dos ámbitos diferentes:

- Frontal de usuario.
- Tratamiento de los datos.

Por su parte, el back-end también registraba las peticiones recibidas mediante un log y/o envío de correos electrónicos.

El back-end ligero desarrollado, por su parte, ha permitido comprobar que el sistema carga las tablas auxiliares como se esperaba, así como la recepción de datos desde la aplicación móvil.

Este back-end ligero ha sido desarrollado como una serie de páginas aspx con enlace directo a la base de datos, sin mayor complejidad.

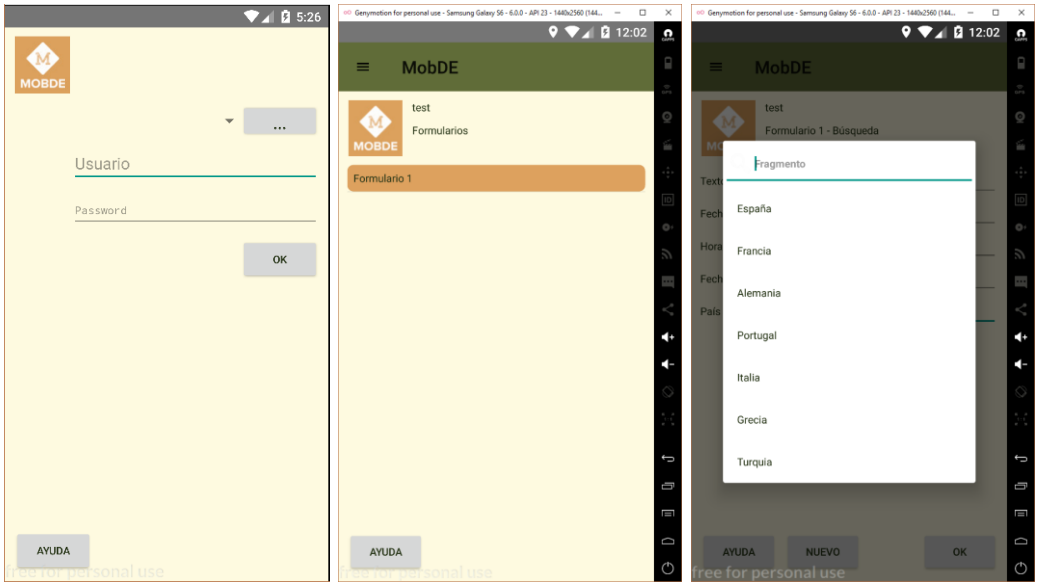


5 Interfaz de usuario

En general la interfaz de usuario respeta bastante el diseño inicial, se ha incorporado una cierta nota de color para huir de la monotonía monocroma del diseño original y se han redondeado los elementos de las listas, poco cambio más.

Se ha adquirido un logo para la aplicación respetando los colores seleccionados con la intención de hacer un poco más agradable el entorno.

Se incluyen a continuación unas capturas de pantalla obtenidas de emulador.



En la actualidad se dispone únicamente de controles de tipo texto, en el futuro la librería de controles debería crecer, para incorporar otros controles más gráficos, tanto para valores lógico, numéricos o incluso miniaturas de imagen, por ejemplo.

6 Implementación

6.1 Convenciones de nomenclatura de variables

En el código se ha procurado seguir las convenciones establecidas en los requisitos que establece Google para contribuir en el *Android Open Source Project* (Google, s.f.) ya que han sido las únicas guías de estilo/buenas prácticas localizadas específicas para Android.

También se ha recurrido a

http://www.oracle.com/technetwork/java/codeconventions-135099.html

en caso de duda o ausencia.

En resumen, las convenciones aplicadas son las siguientes:

- Para las actividades:
  - Nombre de clase, *camelcase* con la primera letra en mayúscula y utiliza el sufijo Activity.
  - Layout, todo en minúsculas, utiliza el sufijo _activity.
  - Componentes del layout, utilizan el prefijo correspondiente a su nombre de layout, un afijo del tipo de control y finalmente el nombre del control, todo ello separado por _ (guion bajo).
  - Variables correspondientes a los componentes del layout dentro de la clase correspondiente *camelcase* con la primera letra minúscula y obvian el prefijo correspondiente al nombre de la actividad.
  - Ejemplo
    - ConfiguracionProyectoActivity.class
    - configuracion_proyecto_activity.xml
    - configuracion_proyecto_activity_btn_ok
    - btnOk
- Métodos
  - Infinitivos, en *camelcase,* con la primera letra en minúscula.

6.2 Implementación basada en fragmentos

Inicialmente tenía previsto hacer el desarrollo basado en activities en lugar de fragments, pero he acabado cambiando de opinión y derivando a trabajar por fragments puesto que creo que esta aplicación es un caso claro de uso de fragmentos (e incluso el diseño inicial de los sketches parece que iba en esa dirección, aunque fuera de modo inconsciente).

Esta decisión me ha permitido aprender bastante sobre el funcionamiento de los fragmentos y la comunicación entre ellos y su actividad principal.

6.3 Implementación basada en metadatos

Aunque en todo el documento se hace referencia a esto, cabe insistir en este momento en que el desarrollo está condicionado en todo momento a que se están trabajando estructuras y datos desconocidos en el momento de la implementación, es por ello que se dispone de clases que realizan las funciones de Modelo, Vista y Controlador pero en todo momento realizan consultas a los metadatos para establecer cómo deben realizar sus funciones. Esto implica, por ejemplo, que el pintado de la pantalla, por

ejemplo (que correspondería a la vista) se realiza consultando los metadatos para establecer los controles a utilizar, así como los títulos y botones de acción a presentar.

El modelo por su parte apenas existe como clase ya que de algún modo su función queda absorbida por la cadena json que contiene los metadatos.

Finalmente el controlador genérico se ve obligado a consumir continuamente los metadatos (que actúan, insisto, como modelo) para recorrer la vista y poder consumir y alimentar la base de datos.

6.4 Controles utilizados

He desarrollado dos controles que extienden controles estándar de Android añadiéndoles una cierta funcionalidad extra para simplificar el pintado dinámico de pantallas.

Esto es algo que no había hecho nunca y creo que ha sido muy interesante, extender los controles me ha permitido eliminar mucho código de las pantallas que los utilizan y delegar en ellos mucha carga.

En particular he desarrollado dos controles que implementan una interfaz que deberían implementar el resto de controles que, con el tiempo, se incorporen a Mobde.

- EditText (derivados en MobdeEditText) para presentar-registrar datos, con un cuadro de dialogo vinculado en el caso de tablas auxiliares y selectores de fecha-hora si corresponde. Estas capacidades se podrían ampliar en el futuro para indicar que se deben registrar códigos de barras, imágenes, etc.
- TextView (derivado en MobdeTextView), de hecho, es un control compuesto, que modifica su presentación según el tipo de dato:
    - Presenta un icono de “pdf” en el caso de tratarse de un campo de fichero (en el futuro, jpg y otros) e incorpora el código responsable de descargar/presentar dicho fichero.
    - Presenta un icono de teléfono y permite llamar en el caso de datos de tipo “phone"
    - Presenta un icono de correo y lanza la aplicación de correo electrónico en el caso de campos de tipo “email”.
    - Esta lista de tipos se puede ampliar en el futuro.

Como se indica, se ha definido una interfaz con los métodos necesarios para el pintado de los controles y su relación con los campos, esto permite incorporar nuevos controles que implementen dicha interfaz, este paso es necesario de cara a la versión 2 de la aplicación para aumentar las posibilidades de diseño de las pantallas.

Esta decisión me ha permitido aprender a extender los controles estándar de Android e incorporar lógica en ellos (como la llamada a los diálogos de búsqueda para los campos codificados).

6.5 Implementación del back-end

El back-end se ha desarrollado en c# como una serie de servicios (detallados en el capítulo correspondiente 3.2.4.1), se ha realizado la implementación mínima necesaria para que el frontal pueda funcionar,

pero no existe entorno de diseño ni control sobre los metadatos, en una versión comercial faltaría:

- Enlace con el resto de entorno de la empresa.
- Frontal de diseño de formularios para no tener que escribir los json a mano.

6.6 Implementación del frontal web de consulta

Se han desarrollado una serie de páginas aspx que permiten ver como la información llega hasta el servidor. Esta información es consultable desde http://lpardo.selfip.net

Este frontal web incluso permite alterar los json de definición del entorno, pero hay que tener en cuenta que el servidor no dispone de un control del entorno, así, si se realizan alteraciones en los metadatos de formularios estos podrían dejar de funcionar, por ello existe un botón de “restaurar situación inicial”.

6.7 Problemas encontrados

El mayor problema es que mi rendimiento en Android aún es bajo comparado con otros lenguajes en que tengo mayor experiencia, esto hace que haya tenido que aumentar la dedicación y pese a ello no haya llegado hasta donde quería llegar.

Inicialmente definí las consultas para utilizar el operador “igual” en los campos de texto, pero por recomendación de Pau Dominkovics he pasado a utilizar el operador “contiene” para que resulte más claro.

6.7.1 Controles de usuario y su estado

No estoy seguro de que el sistema utilizado para guardar el estado de los controles sea el correcto, me parece demasiado complicado (el fragmento le pide al control que guarde el estado y este utiliza una clase interna, tengo la sensación de que la clase interna debería ser suficiente, pero por algún motivo no es llamada).

He intentado utilizar el código de https://github.com/IntertechInc/android-custom-view-tutorial/blob/master/customviews/src/main/java/com/intertech/customviews/ValueBar.java para hacer que mis controles guarden su estado de manera autónoma, pero no funciona como esperaba, por ello hay código creo que redundante de guardar estado en los fragmentos y el control.

6.7.2 Control de rutas de archivos

No había trabajado con ficheros distribuidos con la propia aplicación (como el manual de usuario) y poder hacerlo accesible ha dado unos ciertos problemas con la gestión de permisos y sandbox de Android. He necesitado unas cuantas consultas a developer.android.com y stackoverflow para conseguir que funcionase.

6.8 Fallos conocidos

Todos los fallos localizados han sido corregidos.

6.9 Puntos para versiones futuras

El botón documentación lanzará un servicio web específico para obtener la versión actual. Si el servicio falla abrirá el documento que tenga (si tiene alguno). Si el servicio funciona, abrirá el documento actual o descargará el nuevo según corresponda.

Actualmente todos los campos de las tablas locales están definidos para soportar valores null, se puede incorporar en el futuro una bandera para indicar si se soportan o no.

En este momento no se utilizan los valores de retorno de las inserciones en las tablas ni se presenta información al usuario acerca del progreso de envío a back-end, se podría incorporar esa información.

La aplicación está diseñada pensando en que una única instalación pueda conectar a *n* entornos diferentes, pero la configuración de estos entornos es manual, futuras versiones podrían incorporar una opción para configurar mediante un QR-code, así bastaría con acceder a la pestaña de configuración y fotografiar el código del entorno al que queremos conectarnos.

6.10 Ejemplos código consultados

- La base del código para tratar json se obtuvo en su momento de http://www.androidhive.info/2012/01/android-json-parsing-tutorial/ (Diversos, AndroidHive, s.f.)
- La información sobre el tratamiento de la geolocalización se ha obtenido de http://developer.android.com/intl/es/guide/topics/location/strategies.html (Google, s.f.)
- El 'nuevo' tratamiento de permisos a partir de Android 6 se consulto de http://stackoverflow.com/questions/31928868/how-do-we-distinguish-never-asked-from-stop-asking-in-android-ms-runtime-permis/35495372#35495372 (Diversos, StackOverflow, s.f.)
- Finalment, el método para mostrar errores en campos de texto se ha copiado de http://www.technotalkative.com/android-show-error-in-edittext/ (Mayani, s.f.)

6.11 Instrucciones para la ejecución del entorno de pruebas

Por simplicidad, se ha establecido que este entorno ya va preconfigurado en la aplicación.

Datos de conexión:

Proyecto: cualquier nombre

Servidor: lpardo.selfip.net

Usuario: mobde

Password: mobde

7 Conclusiones

Puede ser interesante plantearse la posibilidad de permitir que otras aplicaciones lancen *activities* de esta aplicación para completar sus funcionalidades.

Aunque la aplicación hace lo que se quería que hiciera queda claro que una versión comercial debería permitir construir interfaces de usuario más complejas. Para ello bastará con rediseñar el contenido de los json de definición de cada una de las pantallas.

Sería interesante incorporar validaciones de datos más complejas, por ejemplo: que el valor de un campo no pueda ser inferior al de otro campo.

Asimismo, se podrán incorporar visibilidades/habilitaciones condicionales, de modo que determinados campos solo sean visibles/editables para determinados valores de otro campo.

8 Glosario

La mayor parte del vocabulario utilizado en el presente documento, así como en el proyecto asociado, es de uso común y habitual, pero a continuación se listan los términos más específicos, así como los anglicismos para evitar posibles confusiones.

Activity	Cada una de las pantallas de una aplicación Android.
Android	Sistema operativo basado en Linux sobre el que funcionan los dispositivos móviles objeto de este desarrollo.
Android Studio	Entorno de desarrollo.
API	Application Programming Interface, conjunto de funciones y procedimientos proporcionados por una librería de programes.
APK	Nombre con el que se conocen las aplicaciones para Android (Android Application Package) derivado del formato jar.
GPS	Global Positioning System, es un sistema de geolocalización por Satélite. En el contexto de este documento se utiliza como sinónimo de geolocalización sea cual sea el origen de la información.
Intent	Mecanismo de Android para describir una operación, como puede ser la llamada a una activity (formulario).
SQLite	Base de dades disponible en los dispositivos Android

9 Bibliografía

Diversos. (s.f.). *AndroidHive*. Obtenido de www.androidhive.info

Diversos. (s.f.). *StackOverflow*. Obtenido de www.stackoverflow.com

Google. (s.f.). *developer.android.com*. Obtenido de developer.android.com

Google. (s.f.). *developers.google.com*. Obtenido de developers.google.com

Google. (s.f.). *Source*. Obtenido de https://source.android.com/setup/contribute/code-style

Mayani, P. (s.f.). *technotalkative*. Obtenido de http://www.technotalkative.com/android-show-error-in-edittext/

Microsoft powerapps. (s.f.). Obtenido de https://powerapps.microsoft.com/es-es/

Network Working Group. (s.f.). Obtenido de https://www.ietf.org/rfc/rfc2617.txt

Vogella. (s.f.). *Vogella Tutorials*. Obtenido de http://www.vogella.com/tutorials/AndroidTesting/article.html