Guia de desarrollo Android en mSSPA

Contenido

Resumen

Versión: v01r00

Fecha de publicación: 11 de octubre de 2019

Fecha de entrada en vigor: 11 de octubre de 2019

Introducción

La presente guía pretende ser un conjunto de buenas prácticas para el desarrollo de aplicaciones móviles Android que formarán parte del ecosistema del proyecto mSSPA.

Los componentes del proyecto mSSPA, a su vez, se basan en el resto de normativas definidas en la STIC, como por ejemplo, la de arquitectura y desarrollo para los componentes de backend, o la de interoperabilidad para la necesidad de consumo y definición de servicios que utiliza cada App.

A continuación se listan los enlaces a los distintos apartados ya existentes y disponibles públicamente en UNIFICA, para facilitar el acceso directo a los mismos:

Información sobre arquitectura y normativa de desarrollo: Arquitectura

Información sobre interoperabilidad: Interoperabilidad

Información sobre normativa de calidad: Calidad

Información sobre integración con las herramientas de gestión TIC: Herramienta de gestión TIC - Servicios CGES

Versión mínima del Sistema Operativo (SO)

Se establece la versión mínima del SO en donde se podrá ejecutar cualquier App del ecosistema del proyecto mSSPA. Para Android, la mínima versión soportada es la **5.0 Lollipop (API Level 21)**. Según los últimos datos de adopción del SO Android, aproximadamente el 90% de los dispositivos Android existentes tienen instalada al menos dicha versión o una superior, por lo que el rango de dispositivos objetivos de cualquier aplicación es bastante amplio.

Entornos y Versiones

Disponer de entornos de desarrollo separados en las distintas fases del ciclo de vida del software evita datos incongruentes o erróneos en entornos productivos. Para los proyectos de aplicaciones Android se contará con dos entornos de trabajo distintos, uno para la fase de desarrollo al que llamamos entorno PRE y otro para la fase de producción al que llamaremos entorno PRO. Para ello se hará uso de las herramientas que ofrece gradle para la diferenciación de recursos y configuración. Así, en el proyecto se contará con dos flavors distintos, uno llamado “Pre” y otro llamado “Pro” con la configuración necesaria para cada uno de los distintos entornos de desarrollo, tales como URLs de servidores y otras opciones.

Del mismo modo se diferencian las versiones de compilación. De esta forma, como mínimo, tendremos una versión de compilación de “debug” y otra “release”. También se pueden establecer opciones de configuración distintas para cada uno de los modos de compilación, tales como la ofuscación de código.

En este sentido, es la versión “release” la que se seleccionará cuando se quiera crear una versión funcional de la aplicación, que esté lista para la subida a la tienda de aplicaciones. Esto será de obligado cumplimiento para las aplicaciones Android.

Lenguaje de programación

El desarrollo de aplicaciones móviles Android estará basado en código **Kotlin**. A ser posible, estará actualizado a la última versión estable de dicho lenguaje de programación. Kotlin es un lenguaje de programación de reciente creación diseñado por JetBrains. Es un lenguaje funcional con una sintaxis simple y muy eficiente, y rápido en cuando a rendimiento se refiere.

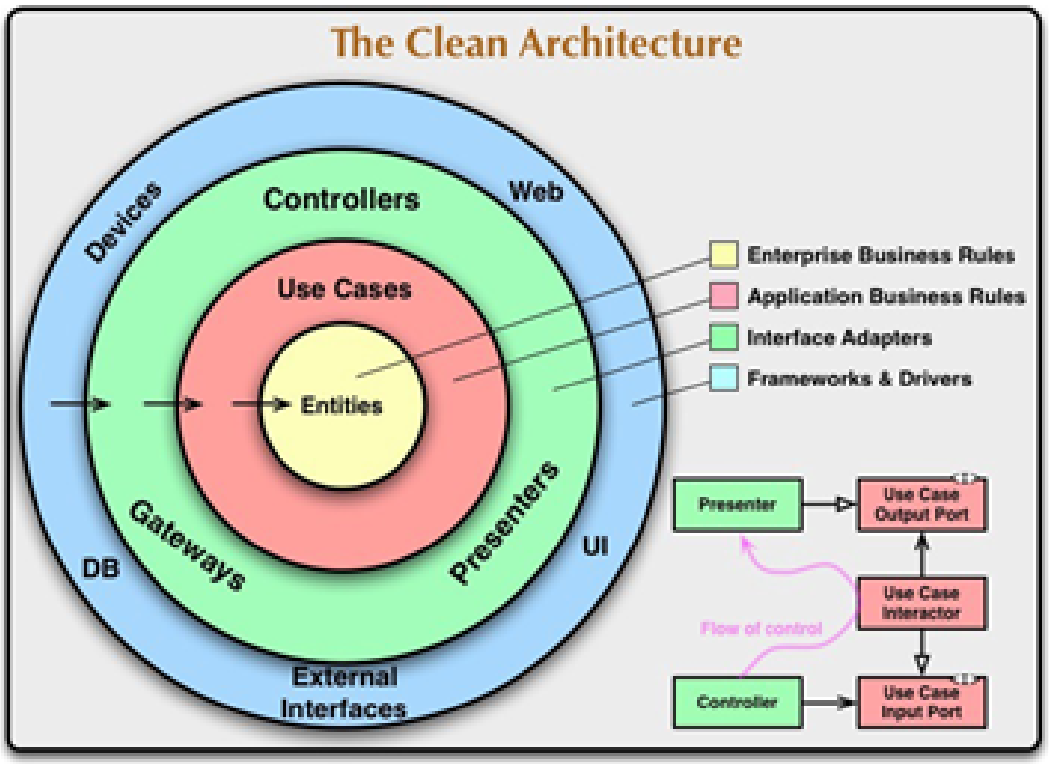
Arquitectura

Con objeto de garantizar la robustez, calidad y mantenibilidad de la aplicación, así como una buena estructura del código de la aplicación, es muy importante definir una buena arquitectura. En este apartado se describe la arquitectura adoptada para las aplicaciones Android.

Clean Architecture

La arquitectura a usar por la apps Android será Clean Architecture, una arquitectura de uso extendido en el desarrollo nativo de aplicaciones móviles. Entre las ventajas de usar **Clean Architecture** podemos citar:

- Separación del código en capas independientes con responsabilidades bien definidas para cada una de las capas.
- Mayor nivel de abstracción.
- Código desacoplado.
- Facilita el uso de tests.

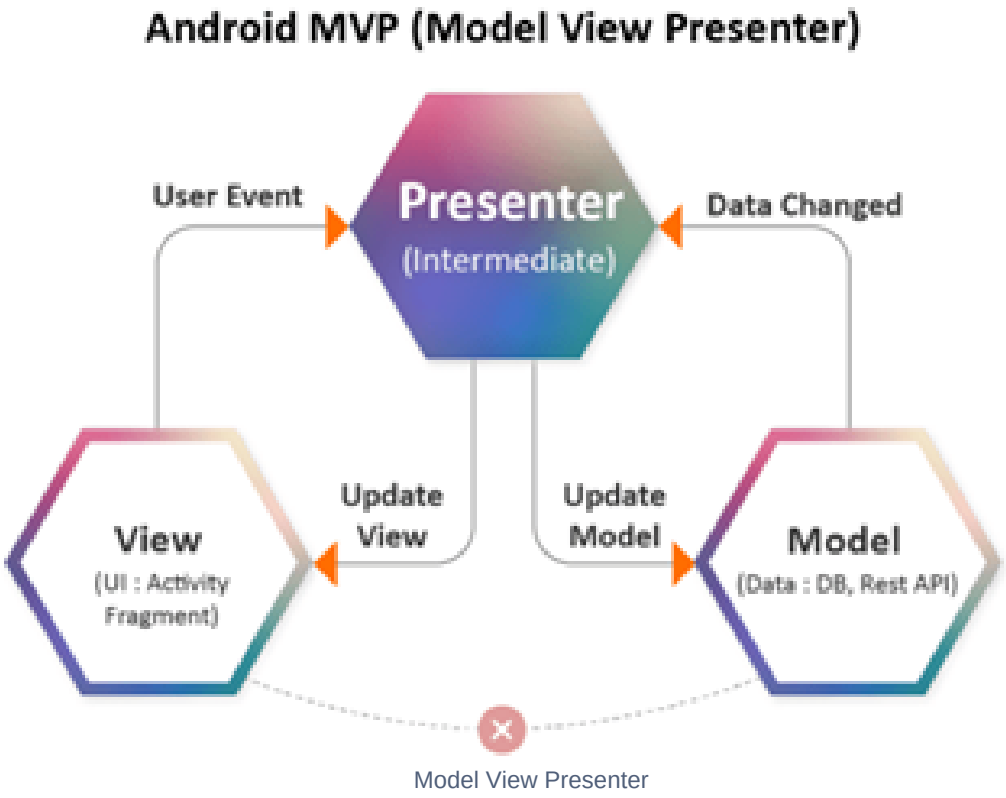


Clean Architecture

Clean Architecture

Patrón de Diseño – MVP

Uso del patrón de diseño **MVP** (Model View Presenter). Patrón de diseño muy utilizado en el sector del desarrollo móvil nativo. Dicho patrón separa la capa de la vista o presentación con la capa de negocio, haciendo de esta manera el código más robusto, mantenible y testeable.



Model View Presenter

Patrón de Diseño - Repository

Una de las ventajas de tener el código de la app Android separado en capas es la independencia de cada una de las capas en cuanto a responsabilidades. En este sentido, en la capa de acceso a datos haremos uso del patrón de diseño Repository. Este patrón nos permite separar la fuente de datos del resto de partes de la aplicación permitiendo en caso necesario interactuar con una o varias fuentes de datos distintas e incluso cambiar la fuente de datos utilizada por la aplicación.

Componentes y Tecnologías

A continuación se describen los componentes y tecnologías más relevantes incluidas en el desarrollo de app Android.

Eventos – Rx (RxJava/RxAndroid/RxKotlin)

La gestión de eventos y tareas es vital para la usabilidad y fluidez de una aplicación Android. Está totalmente desaconsejado realizar operaciones costosas a nivel computacional en el hilo principal de la aplicación, ya que si este tipo de operaciones las hacemos en el hilo principal dejaremos la interfaz de usuario congelada hasta que la operación termine, bloqueando al usuario incluso a poder navegar a otras vistas o realizar otra acción. Para evitar bloquear la interfaz de usuario, haremos uso de ReactiveX (y sus extensiones RxJava, RxAndroid y RxKotlin) como API para la gestión de eventos, tales como el manejo de peticiones al servidor de soporte de la aplicación, almacenado de datos en preferencias de usuario, lectura de configuración por defecto, etc.

Inyección de dependencias – Dagger2

Inyección de dependencias es un patrón de diseño software muy útil para facilitar el uso de instancias de clases. Con el uso de este patrón de diseño evitaremos tener que instanciar clases que comparten ámbito dentro del software, asegurando que siempre se utiliza la misma instancia en cualquier parte que se necesite.

La inyección de dependencias encaja con la arquitectura de la aplicación propuesta, aportando limpieza y simpleza en el código necesario para la ejecución de las distintas funcionalidades.

En este sentido, usaremos Dagger2 como librería para implementar la inyección de dependencias.

Red – Retrofit

Retrofit es una librería de comunicación que se usa para encapsular las peticiones HTTP a nivel de red. Esta librería facilita la comunicación con el servidor de soporte correspondiente vía API, modelando las peticiones y las respuestas para cada uno de los servicios expuestos y utilizados en la app.

AndroidX

AndroidX es la nueva versión de Google para la librería de soporte. La librería de soporte es la librería encargada de asegurar la retrocompatiblidad de los últimos componentes de diseño de Material Design con dispositivos con versiones anteriores del SO. Antes de AndroidX existían varias versiones de la librería de soporte (v4, v7, v8, v13…) cada una de las cuales añadían los nuevos componentes para mantener la retrocompatibilidad con las versiones anteriores. En este sentido, AndroidX es la nueva estrategia para aglutinar todos estos componentes bajo la misma librería de soporte.

Android Navigation

En las aplicaciones Android se hará uso del componente Navigation. Navigation es un componente nativo Android que facilita la navegación entre las distintas partes de la aplicación. Este componente nos permite también de una forma rápida tener una imagen de todo el mapa de navegación completo de la aplicación, haciendo la navegación más intuitiva y sencilla.

Firebase

Firebase es un conjunto de herramientas y servicios proporcionadas por Google que facilitan ciertas tareas de análisis, configuración, comunicación entre otras para con las distintas aplicaciones móviles que lo integren. De todos los servicios que ofrecen, pensamos que los mínimos que deben integrarse dentro de las apps Android son los siguientes:

- Analytics: Es importante conocer el uso que un usuario hace de cualquier aplicación una vez la app ha sido liberada. Es interesante medir tiempos de sesión, uso de ciertas funcionalidades, etc. En este sentido, mientras más datos se conozcan sobre el uso de la aplicación más argumentos de decisión se tendrán para actuar en consecuencia. Analytics permite definir tanto eventos como propiedades de los usuarios que formarán parte de las estadísticas de uso de la aplicación.
- Crashlytics: Al igual que el apartado anterior, es de vital importancia conocer los errores de la aplicación en tiempo real. Crashlytics proporciona información sobre los errores que se han producido en la ejecución de la aplicación por parte de los usuarios, aportando información técnica muy valiosa que permite disponer de una traza de en dónde y qué ha provocado el error, permitiendo así solventar esos posibles errores en el menor tiempo posible.
- Remote Config: Cambiar el comportamiento de la aplicación en tiempo real es útil debido a que permite redirigir el flujo de comportamiento de los usuarios dentro de la aplicación de forma transparente y rápida en función de determinados parámetros, tales como versión del SO, localización, audiencias, etc.
- Cloud Messaging: Aunque las notificaciones push son el objetivo de otro hito dentro del proyecto mSSPA (Servidor de Notificaciones), una app Android contará con una gestión mínima de notificaciones push, es decir, la app será capaz de recibir y mostrar notificaciones push cuando el proyecto se esté ejecutando en segundo plano y queramos establecer un canal de comunicación con el usuario de la aplicación.

WebViews

Por motivos de seguridad es desaconsejable el uso de WebViews para mostrar cualquier contenido web de terceros dentro de la aplicación. Partiendo de esa premisa, hay veces en las que es útil cargar contenido web dentro de la aplicación.

Las aplicaciones Android controlarán esta integración para hacerlo de forma segura y minimizando el impacto de la misma. En este sentido se adoptan las siguientes medidas entre otras:

- Sólo se permite la carga de contenido web que provengan de servidores de confianza y securizados.

- Se deberá deshabilitar por defecto poder cargar el contenido JavaScript asociado al HTML de la web que se quiere cargar dentro del WebView.
- El WebView no tendrá acceso a ningún recurso de la aplicación
- La aplicación no cacheará ningún dato proporcionado por la WebView

Integración con MAG (Mobile API Gateway)

Las aplicaciones Android incorporarán la última versión del SDK para CA Mobile API Gateway para poder aprovechar las funcionalidades que dicha plataforma proporciona.

Funcionalidades

En este apartado se describen las funcionalidades mínimas comunes para todas las aplicaciones Android que formarán parte del ecosistema del proyecto mSSPA.

Versiones

Tal y como se ha detallado en apartados anteriores, se contará con dos versiones distintas, la versión “debug” y la versión “releáse”, además de contar también con dos entornos de desarrollo distintos, PRE y PRO.

Con el objetivo de poder diferenciar claramente el contexto en el que se está ejecutando la app se debe diferenciar el nombre de la aplicación, así como su nombre de paquete, incluyendo el sufijo “DEV”, permitiendo en este sentido incluso poder instalar ambas versiones (“debug” y “releáse”) de la app en el mismo dispositivo.

Menú lateral

Las aplicaciones Android incluirán el elemento Drawer Layout para proporcionar un menú lateral con accesos directos a distintas secciones de la aplicación.

Además de accesos directos a secciones, el menú lateral deberá de informar de la versión de la aplicación que se está ejecutando, además del entorno de desarrollo. Para ello, en caso de que la aplicación se esté ejecutando en el entorno de desarrollo de PRE se incluirá un texto indicativo a la versión de la aplicación.

Login

Desde las aplicaciones Android podremos autenticar al usuario contra los sistemas de información del SAS de forma que tras la autenticación obtengamos un token de acceso para sucesivas llamadas a dichos sistemas.

Para este propósito las apps se integrará con el nuevo sistema de login basado en WebView a través del producto de mSSPA denominado "**Web de Opciones de Autenticación**", que actualmente contiene como opciones de login Certificado Digital, Cl@ve, WebMovil SSPA (login con QR) y DMSAS (login profesional).

Esta integración se llevará a cabo haciendo uso de la API y servicios expuestos a través de CA para la posterior obtención del token de acceso para sucesivas peticiones para la obtención de datos.

Deeplinking

Las aplicaciones pertenecientes al proyecto mSSPA deben tener la capacidad de abrirse desde enlaces externos desde otras aplicaciones tales como el gestor de correo o otra aplicación del proyecto mSSPA.

Se debe establecer, de forma clara, las urls que las aplicaciones Android sabrán interpretar como links externos para poder abrir la aplicación o la sección correspondiente en cada momento.

Logger – Timber

Con el objetivo de poder conocer los eventos y las posibles anomalías que pudiesen suceder, se propondrá un sistema de Logger para identificar claramente qué y dónde se ha producido el evento o anomalía.

Así mismo se deberá evitar escribir mensajes en los logs en las versiones de las aplicaciones que se publiquen en el store ya que estos datos sólo son útiles durante el desarrollo de la aplicación. Además, también se evitará la escritura de datos confidenciales del usuario.

Ocultar información en segundo plano

Debido a la confidencialidad de la información con las que las aplicaciones del proyecto mSSPA trabajarán, es de debido cumplimiento que las aplicaciones oculten la información mostrada cuando dichas aplicaciones pasen a segundo plano.

Evitar capturas de pantalla

Al igual que el apartado anterior, las aplicaciones del proyecto mSSPA no permitirán que se puedan realizar capturas de pantalla para evitar así filtrar información confidencial del usuario.

Autenticación biométrica

Cada vez son más los dispositivos con capacidades biométricas integradas, como lector de huella o reconocimiento facial.

Las aplicaciones Android dispondrán de autenticación biométrica siempre y cuando el dispositivo lo permita. Este método de autenticación sentará las bases para que futuras funcionalidades puedan hacer uso de la autenticación biométrica en caso de ser requerida.

Seguridad

La seguridad es un aspecto clave debido a la sensibilidad de los datos a tratar en las aplicaciones del proyecto mSSPA. En este sentido se establecen una serie de medidas mínimas de seguridad tales como:

Comunicación segura

Cualquier comunicación con el servidor vía API deberá ser una comunicación segura bajo protocolo HTTPS.

Dispositivos rooteados

Las apps Android harán las comprobaciones pertinentes para detectar si el dispositivo ha sido “rooteado” o no. Un dispositivo “rooteado” es un dispositivo en el que el usuario tiene control total sobre los privilegios de todas las aplicaciones y procesos que se ejecutan en el mismo, por lo que podría tener acceso a los datos sensibles del usuario.

Ofuscación de código

Se establecerán medidas para la correcta ofuscación del código de la app Android siguiendo las directrices que marca la propia plataforma Android.

Accesibilidad

Cualquier app Android sigue unas pautas de implementación de capacidades de accesibilidad para hacerla usable a personas con algún tipo de limitación, ya sea auditiva o visual. Para ello, deberá seguir la normativa que provee la Oficina de Calidad.

Internacionalización

Aunque el ámbito de las aplicaciones del proyecto mSSPA está claramente definido a la Comunidad Autónoma de Andalucía, además del castellano, las apps Android también soportarán el inglés como idioma principal de la aplicación, facilitando con esto su integración con proyectos internacionales.

Tests

La estabilidad y calidad del código es una pieza fundamental dentro de cualquier proyecto software que se desarrolle.

Para garantizar la robustez y calidad del código, se establecerán test (unitarios, funcionales o de integración) en las distintas capas de la arquitectura de la aplicación que sean capaces de garantizar un buen porcentaje de cobertura de código.

Inicio

Repositorio documental

Fichero	Modificado
Archivo PNG image2020-5-21_14-55-31.png	19/01/2024 by JUAN LUIS LARA RUIZ GRANADOS
Archivo PNG image2020-5-20_11-22-3.png	19/01/2024 by JUAN LUIS LARA RUIZ GRANADOS

Descargar todo