ULPGC Universidad de Las Palmas de Gran Canaria | Escuela de Ingeniería Informática

Diseño de la Base de Datos

Programación de Aplicaciones Móviles Nativas

Autor: Fernando Sanfiel Reyes

Daniel Betancor Zamora

Pablo Santana Susilla

Fecha: 30/10/2023

Índice

1.- Identificación de los modelos y componentes de la capa de persistencia

De cara a nuestro proyecto final, hemos decidido continuar con nuestra propuesta de arquitectura: ***Clean Architecture***. El motivo principal, es más que obvio, pues esta arquitectura ofrece muchas más ventajas que desventajas con respecto a la capa de persistencia de los datos. Concretamente, *Clean Architecture* nos permite desarrollar el proyecto con mucha más seguridad y confianza de cara al futuro, pues ofrece una gran mantenibilidad y escalabilidad, además de facilitar la implementación de cualquier tipo de base de datos a lo largo del proceso de desarrollo o durante las diferentes versiones del producto.

Partiendo de nuestra elección de arquitectura, deberemos definir en que consiste la persistencia de los datos, y de que manera se relaciona con los modelos y componentes de nuestro proyecto. **La capa de persistencia** es un componente fundamental en la arquitectura de software, pues desempeña un papel fundamental en la gestión de datos de una aplicación. En el contexto de *Clean Architecture*, la capa de persistencia se encarga de manejar la interacción con las fuentes de datos, como pueden ser las bases de datos, servicios web, APIs y otro tipo de sistemas. Esta capa es esencial para asegurar que los datos se almacenen, recuperen y modifiquen de manera eficiente y que la lógica de negocio se mantenga independiente de los detalles de implementación. Además, esta capa se relaciona estrechamente con los **modelos** y **componentes** de la aplicación. Los "modelos" representan las estructuras de datos que la aplicación utiliza para representar información, mientras que los "componentes" son las piezas de código que gestionan la interacción con esas estructuras de datos y las fuentes de datos subyacentes.

Por eso mismo, a continuación trataremos de identificar aquellos modelos y componentes que tienen más presencia en nuestra aplicación, explicando detalladamente cuál es la función de cada uno, en que capa de la *Clean Architecture* interactúan, y de qué manera respetan las dependencias establecidas por la arquitectura.

Se ha de destacar que la capa de persistencia se ubica principalmente en la capa de infraestructura de nuestro proyecto, pues es donde se encuentran aquellos componentes que nos permiten trabajar con los datos y asegurar la persistencia de los mismos.

1.1.- Modelos de la aplicación

En primer lugar, trataremos de identificar los modelos de nuestra aplicación, los cuales conforman la base sobre la que se sustenta la capa de persistencia, ya que definen la estructura de los objetos necesarios para la representación del modelo de negocio. Concretamente, son los objetos que representan la información que se almacena en la base de datos o se utiliza en la aplicación, por ejemplo: los productos, los usuarios, las listas de compra, las listas de favoritos, los gastos, etc.

Este tipo de objetos o modelos se ubican en la capa más interna, la capa de dominio, pues son objetos que se han definido detalladamente en el comienzo del proyecto y que representan la lógica de negocio, por lo que no deben cambiar. Cualquier cambio a estos modelos suponen un replanteamiento general de la lógica del negocio y del dominio de la aplicación, por lo que deben ser establecidos correctamente desde un principio y ser totalmente independientes del resto de capas superiores.

A continuación se muestran algunos ejemplos de la estructura establecida para varios modelos de nuestra aplicación.

PRODUCTO

- **Descripción:** Modelo que representa la información detallada de un producto que los usuarios pueden escanear con la cámara de su dispositivo móvil.
- **Atributos:** Nombre, código de barras, imagen, ingredientes, información nutricional, precio.

USUARIO

- **Descripción:** Modelo que representa a los usuarios de la aplicación y se utiliza para la autenticación principalmente.
- **Atributos:** Nombre de usuario, correo electrónico, contraseña, listas de la compra, lista de productos favoritos, contactos, gastos.

LISTA DE LA COMPRA

- **Descripción:** Modelo que representa una lista de productos que los usuarios pueden crear, modificar, eliminar y compartir con contactos.
- **Atributos:** Nombre de la lista, fecha de creación, usuarios incluidos, lista de productos.

1.2.- Componentes de la capa de persistencia

Además de los diferentes modelos que acaban de ser presentados, la capa de persistencia alberga varios componentes que son esenciales para gestionar la interacción con la base de datos, servicios externos y otros sistemas de almacenamiento. A continuación, se incluirán los principales componentes que han sido identificados en la capa de persistencia de nuestro proyecto.

DATA SOURCES

Los *Data Sources* son componentes que se encargan de interactuar con fuentes de datos externas, es decir, la función principal de estos es establecer la conexión con los recursos tecnológicos de almacenamiento que se vayan a usar. Concretamente, en nuestro proyecto, diferenciamos dos tipos de *Data Sources*: uno para **Firebase** y otro para la **API externa**. El Data Source de Firebase maneja la interacción con *Firebase Authentication* y *Firebase Firestore* para autenticar usuarios y almacenar información relacionada con listas de compras, favoritos y gastos. Y, por otro lado, el *Data Source* de la API externa se encarga de obtener información de productos escaneados desde la API y convertirla en objetos de modelo de datos.

Estos componentes tienen bastante presencia en la capa de infraestructura, pues son los que establecen la conexión con todas aquellas tecnologías y servicios que provienen del exterior.

REPOSITORIOS

Los repositorios actúan como interfaces que definen las operaciones necesarias para acceder y manipular datos. En otras palabras, establecen un contrato entre la capa de casos de uso y la capa de infraestructura. Además, este tipo de componentes son responsables de proporcionar métodos como la obtención de información de productos escaneados, la gestión de listas de compras y la autenticación de usuarios.

Resumiendo lo visto hasta ahora, podría decirse que los *Data Sources* se encargan de establecer las conexiones con los servicios de la capa de persistencia, y en los repositorios se implementan todos los métodos responsables de acceder y tratar los datos.

Al fin y al cabo, este tipo de componente es una especie de adaptador (tal y como se contemplaría en la arquitectura hexagonal).

CLASES DE MAPEO (MAPPERS)

Finalmente, en nuestro proyecto también contemplamos las clases de mapeo o *mappers*. Este tipo de componentes se utilizan para transformar datos entre formatos utilizados por la capa de infraestructura y la capa de dominio. Por ejemplo, un componente de este tipo podría convertir los datos de productos provenientes de la API externa en objetos de modelo de datos de producto.

2.- Selección de tecnología para la base de datos

La base de datos que se utilizará en nuestra aplicación *GoShop* será *Firebase*, pero ¿por qué *Firebase* y no otra base de datos? Esta elección se ve respaldada por cuestiones de compatibilidad, dado que tanto *Firebase* como Android son productos de Google. Esta sinergia entre las plataformas asegura una integración fluida y optimizada, lo que facilita el desarrollo y la implementación de características específicas de Android en la aplicación. Esto se traduce en una experiencia de usuario más fluida y en un menor esfuerzo de desarrollo, lo que convierte a *Firebase* en nuestra opción preferida para aprovechar al máximo el ecosistema Android.

Otro punto a favor de *Firebase* es que proporciona un sistema de autenticación sólido que se integra fácilmente en nuestra aplicación, ya que ofrece métodos de inicio de sesión con correo electrónico, *Google* o *Facebook*. En *SQLite*, *Realm* o *DataStore*, tendríamos que implementar nuestro propio sistema de autenticación desde cero o utilizar bibliotecas de terceros.

3.- Integración de la lógica de la capa de persistencia

En cuanto a integración, vamos a hablar del registro y la autenticación de usuarios. Tal y como se comenta en el apartado anterior, *Firebase* permite implementar estas funciones de una forma muy sencilla, además de facilitar al usuario el registro e inicio de sesión haciendo uso de Google. A continuación, se comentará la integración de la lógica del registro y autenticación de usuarios, además de la funcionalidad que estos tienen (una vez se han registrado) de crear listas con diferentes productos de alimentación.

Comenzando por la funcionalidad del registro y autenticación de usuarios, se explicará la implementación desde la capa más interna de la aplicación a la más externa. La capa de Dominio contendrá los DTOs que definen los atributos que tendrá un usuario, en nuestro caso, serán los datos personales del usuario. Pasando a la capa de aplicación, haciendo uso de los DTOs, estos serán mostrados en la aplicación en la página de perfil personal del usuario. La capa de Interfaz contendrá todas las consultas necesarias para el registro y autenticación de usuarios, consultas que se facilitan haciendo uso de los DTOs tanto para la publicación como para la consulta de datos. Por último, la capa de infraestructura hace uso del servicio *Authentication* que ofrece *Firebase* de forma que facilita altamente el desarrollo de la capa de Interfaces.

Con respecto a lo anterior, se quiere comentar brevemente la implementación de la funcionalidad de crear listas de alimentos para un usuario registrado. La implementación en cuanto a capas, es muy similar a la anteriormente comentada. Lo único relevante a destacar, por lo cual existe la restricción de uso de esta funcionalidad únicamente a usuarios registrados, es que el uso de *Firebase Authentication* para la gestión de usuarios nos devuelve un identificador de usuario (UID) único, lo que nos permite la creación de una colección en la base de datos que almacene las listas de dicho usuario y sean solo de ese usuario. Esto facilita el desarrollo de las capas de Interfaces y Aplicación, ya que la capa de Interfaces tan solo tiene que hacer uso de *Firebase Authentication* para la consulta del UID del usuario autenticado y con dicho UID consultar por las listas de dicho usuario. En cuanto a la capa de Aplicación, facilita en gran cantidad el desarrollo, ya que se puede generalizar bastante la implementación haciendo uso de los DTOs necesarios para las listas de alimentos, sabiendo que es la capa de Interfaces la que se encarga de proporcionarle la información del usuario correcto.

4.- Análisis de posibles casos futuros

4.1- ¿Qué pasa si en un futuro se quisiera cambiar el motor de base de datos?

Teniendo en cuenta que utilizaremos *Clean Arquitecture* esto no sería un gran problema, ya que Solo sería necesario modificar las capas de Infraestructura (sustituyendo *Firebase* por el nuevo motor de base de datos deseado). Y también sería necesario modificar la capa de interfaces debido a que habría que cambiar el formato de consultas a la base de datos. El resto de nuestra aplicación quedaría intacto facilitando mucho el cambio de motor de base de datos y el volcado de datos en la nueva base de datos, ya que *Firebase* facilita esto permitiendo la extracción de datos en formato JSON.

4.2.- ¿Qué partes de tu aplicación tendrías que modificar?

Como se comentó en el apartado anterior tan solo se verían modificadas las capas de Infraestructura e Interfaces. La de Infraestructura cambiando *Firebase* por el nuevo motor de base de datos deseado, y en la de Interfaces sería necesario sustituir las consultas a la base de datos de *Firebase* por unas consultas nuevas que se adapten al tipo de consultas del nuevo motor de base de datos.

4.3.- ¿Qué dificultades anticipas?

La mayor dificultad que podría generar el uso de esta arquitectura sería un cambio en el modelo de negocio. Esto se debe a que es un cambio que se vería propagado a todas las capas de la arquitectura. Por este motivo, es importante tener muy clara la perspectiva y modelo de negocio previamente al desarrollo de la aplicación.

4.4.- ¿Cómo podrías diseñar tu aplicación para minimizar el impacto de tal cambio?

Haciendo uso de la *Clean Arquitecture* minimizamos notablemente el impacto de cualquier cambio futuro que pueda ser necesario. Esto se debe a que una de las principales características de esta arquitectura es su alto grado de escalabilidad, facilitando así cualquier tipo de funcionalidad que se quiera agregar en el futuro o cualquier cambio que se quiera hacer en la capa de Infraestructura.

6.- Enlace a GitHub

Repositorio en GitHub: https://github.com/PAMN2023/DataBaseDesign

7.- Bibliografía

[1] KeepCoding. "¿Por qué usar Firebase?" [Recurso en línea]. Disponible en: https://keepcoding.io/blog/por-que-usar-firebase/

[2] Firebase. "Documentación de Firebase para Android". [Recurso en línea]. Disponible en: https://firebase.google.com/docs/android/learn-more?hl=es-419