Bargain Games

Autor: Víctor Béjar Martín

Tutor: Javier Martín Rivero

Fecha de entrega:

Convocatoria: 2024 2025

Motivación **1**
Objetivos propuestos **1**
Metodología utilizada **1**
Tecnologías y herramientas **1**
Estimación de recursos y planificación: **2**
Análisis: **2**
Diseño: **3**
Abstract **3**
Diagramas **4**
Documentación **5**
Pruebas **6**
Conclusiones **6**
Vías futuras **7**

Introducción

Un proyecto de Android desarrollado con la Api de steam para buscar tus juegos favoritos y deseados, ver sus detalles y recibir notificaciones de sus ofertas y noticias.
La aplicación cuenta con un buscador y unos juegos recomendados cuando se seleccionen podrás ir a una vista más detallada del juego. Ahí puedes añadirlo tanto a favoritos como a deseados, luego si cambias de página a la de guardados puedes revisar qué juegos tienes en cada uno y te llegarán notificaciones de novedades y ofertas de estos juegos.

Motivación

Steam muchas veces no te notifica correctamente los juegos en oferta de tu lista de deseados. Además la visualización de noticias no es siempre la más correcta y no te avisa tampoco de ellas. Por eso esta app servirá para que te avise correctamente de estás novedades.

Abstract

Steam is the leading platform for video games among me and my friends. However, despite its popularity, it has a major weakness: the poor communication of news and discounts related to the games users care about. Because of this, I decided to develop an Android app that improves this experience by keeping players informed about their favorite titles.

The app will allow users to add games to a favorites list and receive notifications when there are news updates. Additionally, if a game is added to the wishlist, users will be alerted when the game goes on sale.

Beyond notifications, the app will offer features such as searching for games, viewing detailed descriptions, trailers, images, and the latest news articles directly inside the app.

My main motivation is to create a simple and efficient tool that makes it easier for gamers to stay updated and take advantage of the best offers without having to manually search for them. I believe that by improving the way news and discounts are delivered, users can have a better experience and enjoy their favorite games even more.

Objetivos propuestos

Generales:

- Una búsqueda de juegos de steam.
- Poder ver la lista de juegos deseados / favoritos.
- Notificación de noticias y ofertas.

Específicos:

- Poder marcar juegos como favoritos.
- Poder marcar juegos como deseados.
- Visualización de la información de juegos correcta.
- Ver ofertas de juegos.
- Ver información de noticias.

Metodología utilizada

KANBAN

Backlog:
Obtener datos del juego por la api de steam.
Convertir datos en card.
Poner filtros sencillos de usar en el buscador del usuario.
Notificaciones al usuario.
Saber cuando un juego está en oferta para mandar notificación.
Saber cuando un juego tiene una noticia para mandar notificación.
Base de datos con id del juego, favorito y deseado.
Sacar información de la noticia por la api de steam.
Poder poner un juego en favoritos o deseados.
Poder quitar un juego de favoritos o deseados.

Tecnologías y herramientas

Se ha usado Android studio como ide de desarrollo ya que android es el sistema operativo, que mayor presencia tiene y además movil es donde tiene más sentido el proyecto ya que es donde más accesibilidad para las notificaciones tiene el Usuario.

En el apartado librerías se ha usado Retrofit, Gson, Room, WorkManager y GLide.

Retrofit es la librería usada para obtener información de la API esta tiene un uso muy intuitivo y bien documentado, se ha elegido ya que al ser la más popular es muy fácil encontrar información.

Gson se usa para poder convertir en objetos los Json que son recuperados a través de la Api la elección de esta y no otra es por su gran compatibilidad con Retrofit y android en general al ser de Google.

Room permite convertir objetos a una base de datos de SQLite, esto permite un sencillo paso de los juegos de la API a la base de datos, además de que al ser una librería oficial de Android para interactuar con SQLite tiene mucha información al ser muy usada.

WorkManager nos permite dejar una tarea en segundo plano y ejecutarla en el momento que le digamos además de que si no es posible se pospone a otro momento.

Android Studio: *ide de desarollo para android,*

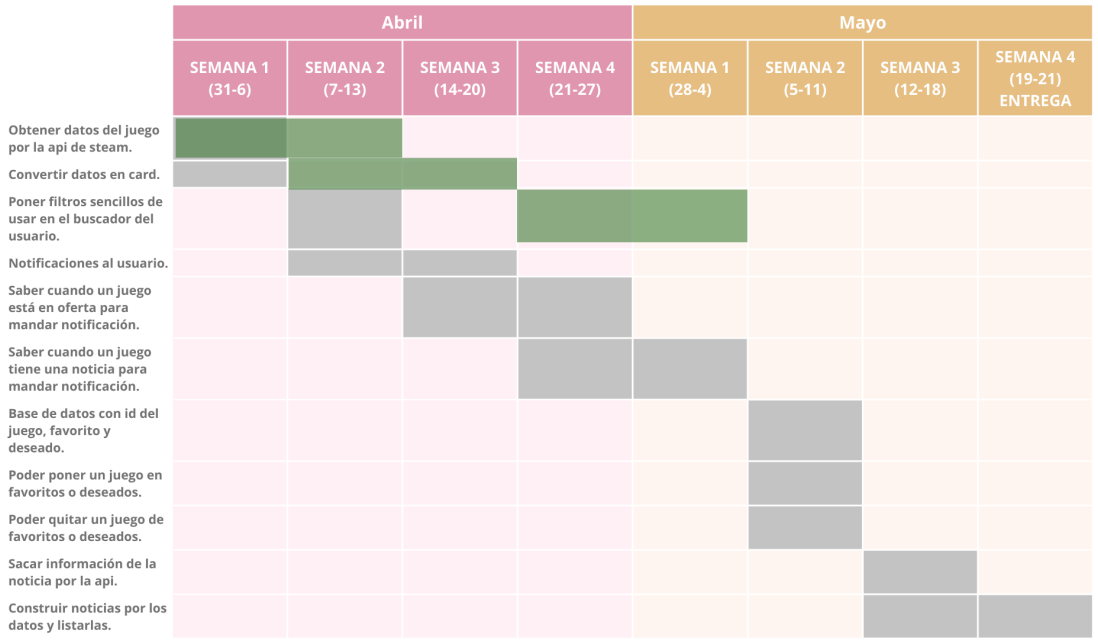
Api de steam: *(wiki, infodeDBsteam , infogit) es la principal tienda de juegos,*

Gson *librería de google para leer json,*

Sqlite*: perfecta para hacer una base de datos pequeña y local,*

Se usará ***kotlin*** *como lenguaje principalmente ya que es el lenguaje principal de Android Studio, aunque algunas clases con mayor complejidad se harán en* ***java****.*

Estimación de recursos y planificación:



Análisis:

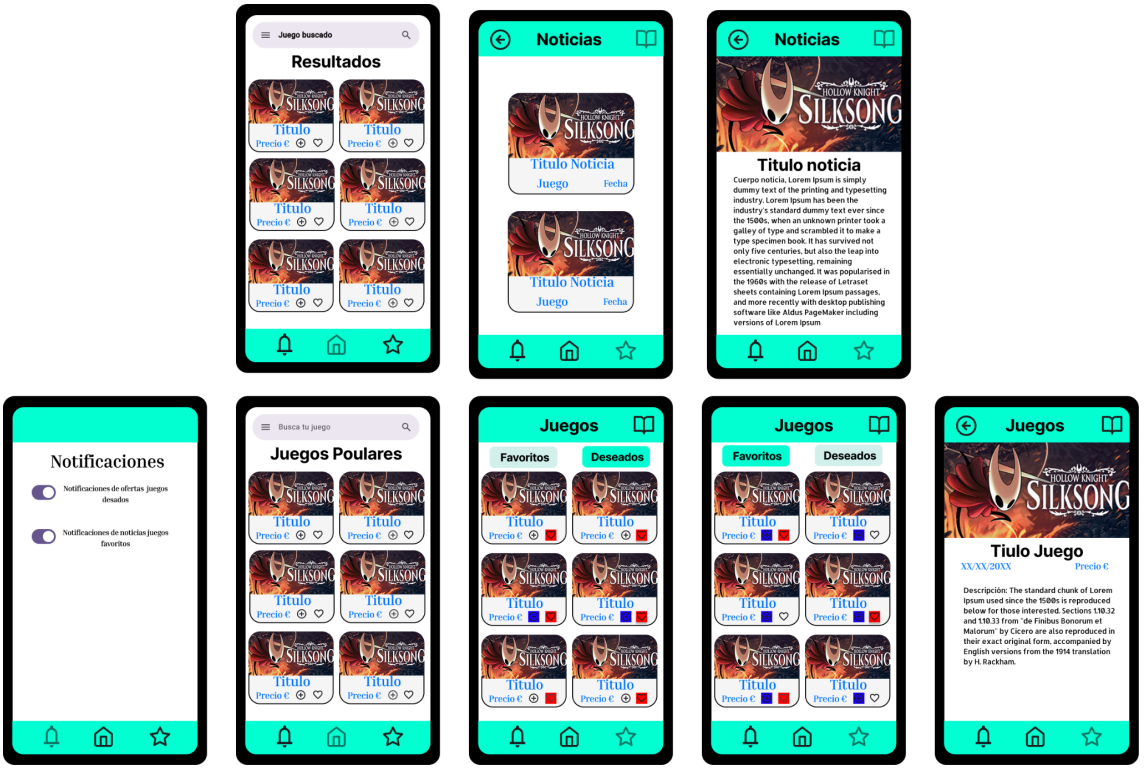
Requisitos funcionales:

Buscar juegos de steam.
Ver información de estos juegos.
Lista de favoritos y lista de deseados.
Notificaciones de noticias de juegos favoritos.
Notificaciones de ofertas de juegos deseados.
Ver información de noticias.
Visualizar juegos populares.
Visualizar tus listas de juegos.

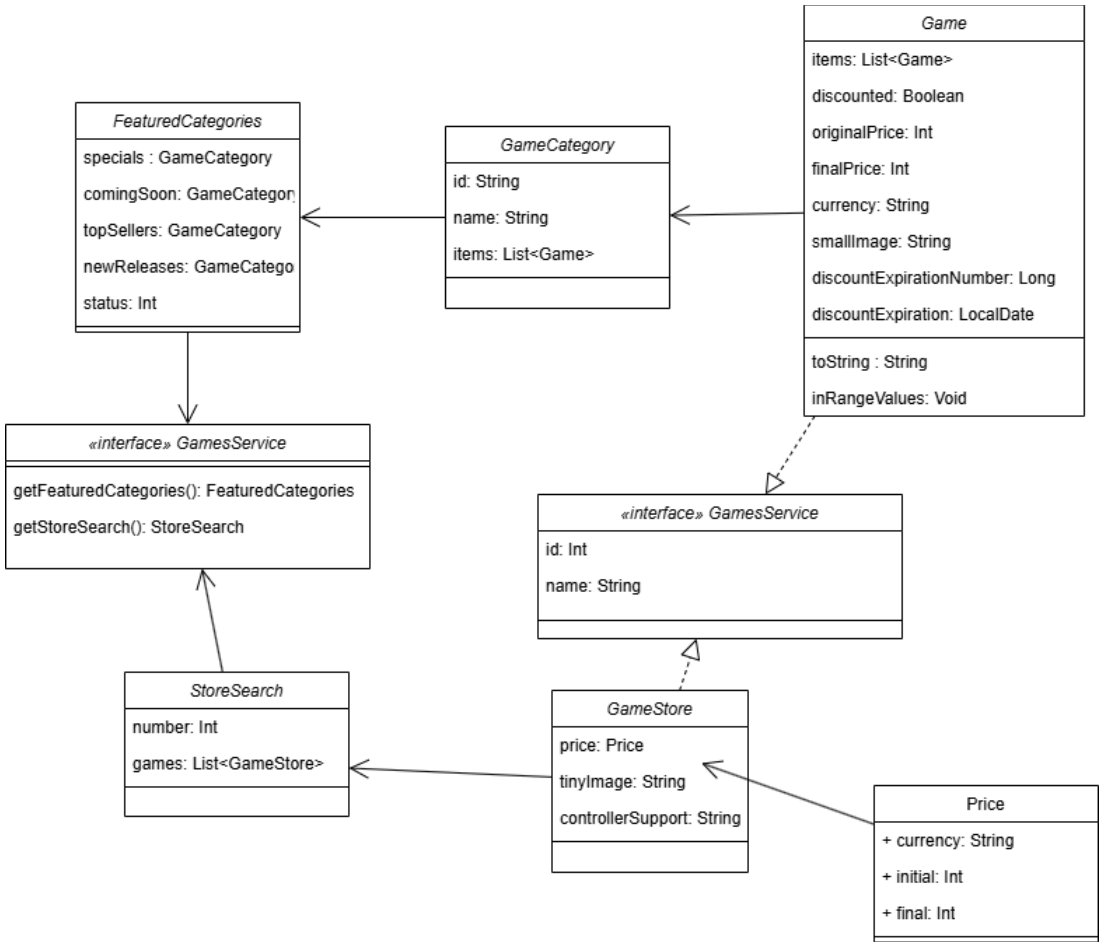
Requisitos no funcionales:

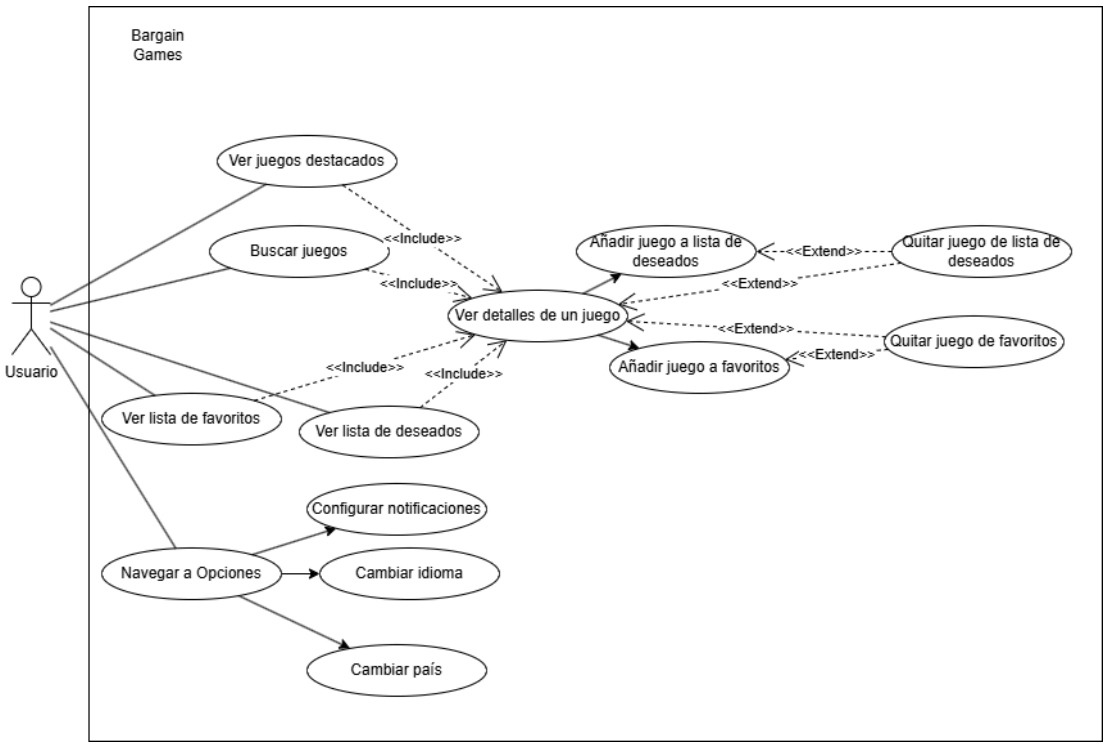
Las notificaciones deben agruparse para evitar el envío masivo y no saturar al usuario.
Las ofertas solo se revisarán de tus juegos en lista de deseados a las 10 am PDT (hora donde se actualizan las ofertas de steam).
Las noticias se actualizan cada 6 horas y cuando el usuario ingresa a la app.
Debe permitir la navegación fluida entre secciones sin necesidad de demasiados clics o acciones.
La app debe ajustarse a distintos tamaños de dispositivos.
La app debe de ser intuitiva de usar.
La app debe ajustarse al modo oscuro de los dispositivos.

Diseño:



Diagramas


Game
items: List<Game>
discounted: Boolean
originalPrice: Int
finalPrice: Int
currency: String
smallImage: String
discountExpirationNumber: Long
discountExpiration: LocalDate
toString : String
inRangeValues: Void
FeaturedCategories
specials : GameCategory
comingSoon: GameCategor
topSellers: GameCategory
newReleases: GameCatego
status: Int
GameCategory
id: String
name: String
items: List<Game>
«interface» GamesService
getFeaturedCategories(): FeaturedCategories
getStoreSearch(): StoreSearch
«interface» GamesService
id: Int
name: String
StoreSearch
number: Int
games: List<GameStore>
GameStore
price: Price
tinyImage: String
controllerSupport: String
Price
+ currency: String
+ initial: Int
+ final: Int


Bargain Games
Ver juegos destacados
Buscar juegos
<<Include>>
<<Include>>
Añadir juego a lista de deseados
<<Extend>>
Quitar juego de lista de deseados
<<Extend>>
Ver detalles de un juego
<<Extend>>
Quitar juego de favoritos
Añadir juego a favoritos
<<Extend>>
Usuario
<<Include>>
<<Include>>
Ver lista de favoritos
Ver lista de deseados
Configurar notificaciones
Navegar a Opciones
Cambiar idioma
Cambiar país

Caso de uso 1	Ver juegos destacados
Alias	-
Actores	- Usuario
Requisito funcional	- Llamar a la API de steam y conseguir los juegos destacados del momento
Descripción	El usuario cuando ingrese a la APP o se mueva podrá ver los juegos destacados en steam en ese momento.
Referencias	Ver detalles de un juego (5)
Comentarios	Ningún comentario

Caso de uso 2	Buscar juegos
Alias	-
Actores	- Usuario
Requisito funcional	- Recuperar la palabra puesta por el usuario y llamar a la API de steam y conseguir los juegos que devuelvan según la palabra puesta
Descripción	El usuario podrá buscar cualquier juego de Steam a través de la búsqueda de este
Referencias	Ver detalles de un juego (5)
Comentarios	Ningún comentario

Caso de uso 3	Ver lista de deseados
Alias	-
Actores	- Usuario
Requisito funcional	- Recuperar los juegos que en la base de datos estén guardados como deseados
Descripción	El usuario podrá filtrar sus juegos guardados por deseados
Referencias	Ver detalles de un juego (5)
Comentarios	Ningún comentario

Caso de uso 4	Ver lista de favoritos
Alias	-
Actores	- Usuario
Requisito funcional	- Recuperar los juegos que en la base de datos estén guardados como favoritos
Descripción	El usuario podrá filtrar sus juegos guardados por favoritos
Referencias	Ver detalles de un juego (5)
Comentarios	Ningún comentario

Caso de uso 5	Ver detalles de un juego
Alias	-
Actores	- Usuario
Requisito funcional	- Obtener los datos del juego que el usuario quiera ver sus detalles a través de la API de steam
Descripción	El usuario podrá ver los detalles de cualquier juego que quiera
Referencias	Añadir juego a la lista de deseados(6)
Comentarios	Ningún comentario

Caso de uso 6	Añadir juego a la lista de deseados
Alias	-
Actores	- Usuario
Requisito funcional	- Se mete los datos del juego en la base de datos y se marca el campo deseado, si ya estaba en la base de datos solo se marca
Descripción	El usuario podrá marcar sus juegos como deseados y guardarlos
Referencias	Quitar juego a la lista de deseados (8)
Comentarios	Ningún comentario

Caso de uso 7	Añadir juego a la lista de favoritos
Alias	-
Actores	- Usuario
Requisito funcional	- Se mete los datos del juego en la base de datos y se marca el campo favoritos, si ya estaba en la base de datos solo se marca
Descripción	El usuario podrá marcar sus juegos como favoritos y guardarlos
Referencias	Quitar juego a la lista de favoritos(9)
Comentarios	Ningún comentario

Caso de uso 8	Quitar juego a la lista de deseados
Alias	-
Actores	- Usuario
Requisito funcional	- Se quita la marca de la base de datos de deseados, si no hay otra se quita la base de datos
Descripción	El usuario podrá desmarcar sus juegos como deseados y quitarlos de guardados
Referencias	Ver detalles de un juego (5)
Comentarios	Ningún comentario

Caso de uso 9	Quitar juego a la lista de favoritos
Alias	-
Actores	- Usuario
Requisito funcional	- Se quita la marca de la base de datos de deseados, si no hay otra se quita la base de datos
Descripción	El usuario podrá desmarcar sus juegos como favoritos y quitarlos de guardados

Referencias	Ver detalles de un juego (5)
Comentarios	Ningún comentario

Caso de uso 10	Navegar menú de opciones
Alias	-
Actores	- Usuario
Requisito funcional	- Se mete a un menú
Descripción	El usuario podrá cambiar cosas en este menú
Referencias	Configurar notificaciones (11) Cambiar país(12) Cambiar idioma(13)
Comentarios	Ningún comentario

Caso de uso 11	Configurar notificaciones
Alias	-
Actores	- Usuario
Requisito funcional	- Se recibe las notificación según instrucciones del usuario
Descripción	El usuario podrá cambiar las notificaciones
Referencias	
Comentarios	Ningún comentario

Caso de uso 12	Cambiar país
Alias	-
Actores	- Usuario

Requisito funcional	- Se cambia el país con el que se hacen las llamadas a steam
Descripción	El usuario podrá cambiar su país
Referencias	
Comentarios	Ningún comentario

Caso de uso 13	Cambiar idioma
Alias	-
Actores	- Usuario
Requisito funcional	- Se cambia el idioma con el que se hacen las llamadas a steam
Descripción	El usuario podrá cambiar su idioma
Referencias	
Comentarios	Ningún comentario

Documentación

```
interface GamesService {
    @GET(FETURED)
    suspend fun getFeaturedCategories(): FeaturedCategories

    @GET(SEARCH)
    suspend fun getStoreSearch(
        @Query("term") term: String,
        @Query("l") language: String = "spanish",
        @Query("cc") countryCode: String = "ES"
    ): StoreSearch

}
```

Las dos funciones que llaman a la API a través del uso de retrofit, se tiene que usar en una interfaz, usamos constantes para que si cambia la API solo tener que tocar el archivo constantes. Con retrofit hay que crear una interfaz con las funciones que se

quieran hacer con la API para luego declararlo así:

```
private fun buscarJuego(termino: String) {
    lifecycleScope.launch(Dispatchers.IO) {
        try {
            val response = service.getStoreSearch(termino, language: "spanish", countryCode: "ES")
            val games = response.games ?: emptyList()
            withContext(Dispatchers.Main) {
                if (games.isNotEmpty()) {
                    searchAdapter.submitList(games)
                    mostrarResultados()
                } else {
                    mostrarNoResultados()
                }
            }
        } catch (e: Exception) {
            Log.e( tag: "HomeFragment",  msg: "Error buscando juego", e)
            withContext(Dispatchers.Main) {
                mostrarNoResultados()
            }
        }
    }
}
```

Esto es como se busca un juego, metemos el término el término se recoge de la barra de tareas que es propia de android y cuando se a buscar y esta algo escrito se manda aquí. Ahora mismo está el idioma por defecto en español y el país en España, esto va a ser editable en opciones en algún momento próximo. Dentro de la respuesta hay una lista “games”, ahí se encuentran todos los juegos que corresponden a la búsqueda a través de la API. Ahora hablamos con el layout y si la lista no está vacía metemos los juegos en el Layout a través de un adaptador y mostramos la lista de juegos, sino mostramos que no se han encontrado juegos.

```
override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
    val gameStore = getItem(position)
    (holder as ViewHolder).run {
        with(binding){
            Nombre.text = gameStore.name

            Precio.text = gameStore.price?.let {
                (it.final.toFloat() / 100).toString() + " " + it.currency
            } ?: "Sin precio"

            Glide.with(itemView.context)
                .load(gameStore.tinyImage)
                .diskCacheStrategy(DiskCacheStrategy.AUTOMATIC)
                .into(imageGame)
        }
    }
}
```

Este es el adaptador a partir de cada elemento que tenga una lista convierte el ítem en una interfaz. Glide es una librería para obtener imágenes de internet a través de una URL, usamos la imagen pequeña ya que carga más rápido y la resolución es más baja pero no se nota por el tamaño, está puesto para que la imagen sea guardada según el criterio de la librería. Además tenemos que comprobar si hay precio si no se pone por ahora sin precio, aunque el objetivo es mirar si el juego es gratis o no ha salido.

```
with(supportFragmentManager) {
    beginTransaction()
        .add(R.id.nav_host, homeFragment)
        .hide(favouriteFragment).commit()

    beginTransaction()
        .add(R.id.nav_host,favouriteFragment)
        .commit()

    binding.navView.setOnItemSelectedListener { menuItem ->
        when(menuItem.itemId) {
            R.id.navigation_home -> {
                beginTransaction().hide(currentFragment).show(homeFragment).commit()
                currentFragment = homeFragment
            }
            R.id.navigation_favorite -> {
                beginTransaction().hide(currentFragment).show(favouriteFragment).commit()
                currentFragment = favouriteFragment
            }
        }
        true
    }
```

Con este pequeño código metemos saltos entre menús sencillos.

```
withContext(Dispatchers.Main) {
    binding.apply {
        name.text = game.name
        description.text = HtmlCompat.fromHtml(game.description, HtmlCompat.FROM_HTML_MODE_LEGACY)

        context?.let {
            Glide.with(it)
                .load(game.image)
                .diskCacheStrategy(DiskCacheStrategy.AUTOMATIC)
                .into(imageGame)
        }
```

Así ponemos las descripciones de html y las imágenes de internet

Pruebas

Se ha hecho algún test muy sencillo en la app, y actualmente como el usuario solo puede meter por el buscador hay pocos valores máximos. Y se han comprobado principalmente en la entrada de API aunque no debería haber valores incorrectos.

Datos a meter	Dato esperado
Porcentaje descuento	Menor a 100 mayor a 0
Precio final	Menor a precio inicial o igual

Conclusiones

Creo que se han conseguido los objetivos, se han completado la parte de ofertas y notificaciones. También se ha hecho la búsqueda de juego y los juego más importantes del día según Steam. Me

Vías futuras

Una de las próximas cosas a añadir sería una pestaña donde puedas revisar bien todas las notificaciones que hayan salido. Eso se suma a un apartado de noticias donde puedas ver todas las noticias, de tus juegos favoritos con sus notificaciones y una página para leer correctamente estas. También un menú de configuración donde puedas cambiar tu moneda, país e idioma.

Bibliografía/Webgrafía

1. RJackson. (n.d.). *User:RJackson/StorefrontAPI*. Team Fortress Wiki. Recuperado el 21 de mayo de 2025, de https://wiki.teamfortress.com/wiki/User:RJackson/StorefrontAPI

2. SteamDB. (2023, 28 de julio). *Store Prices API*. Steam Database Blog. https://steamdb.info/blog/store-prices-api/

3. Revadike. (n.d.). *Get App Details*. Internal Steam Web API Wiki. GitHub. Recuperado el 21 de mayo de 2025, de https://github.com/Revadike/InternalSteamWebAPI/wiki/Get-App-Details