Proyecto Integrado · GameCollection

Agustín Peppe Arias - IES Portada Alta

Objetivo

Gestionar una Colección de videojuegos y conseguir mejores ofertas.

GameCollection es un proyecto basado en experiencia personal, que resuelve una problemática a la hora de comprar videojuegos de segunda mano.

Ofrece:

- Consulta de precios
- Control de gasto
- Gestión de la Colección
- Comodidad y Rapidez

Tecnologías

RAWG

- Spring Boot
- Maven
- Mysql
- Angular
- NginX
- AWS
- Uso de API chatGPT
- Uso de API Cloudinary
- Uso de API externa
- Kotlin - Jetpack Compose

Backend

Elegí **Spring Boot** para el Backend, por el hecho de aprender una nueva tecnología que sume a mi conocimiento.

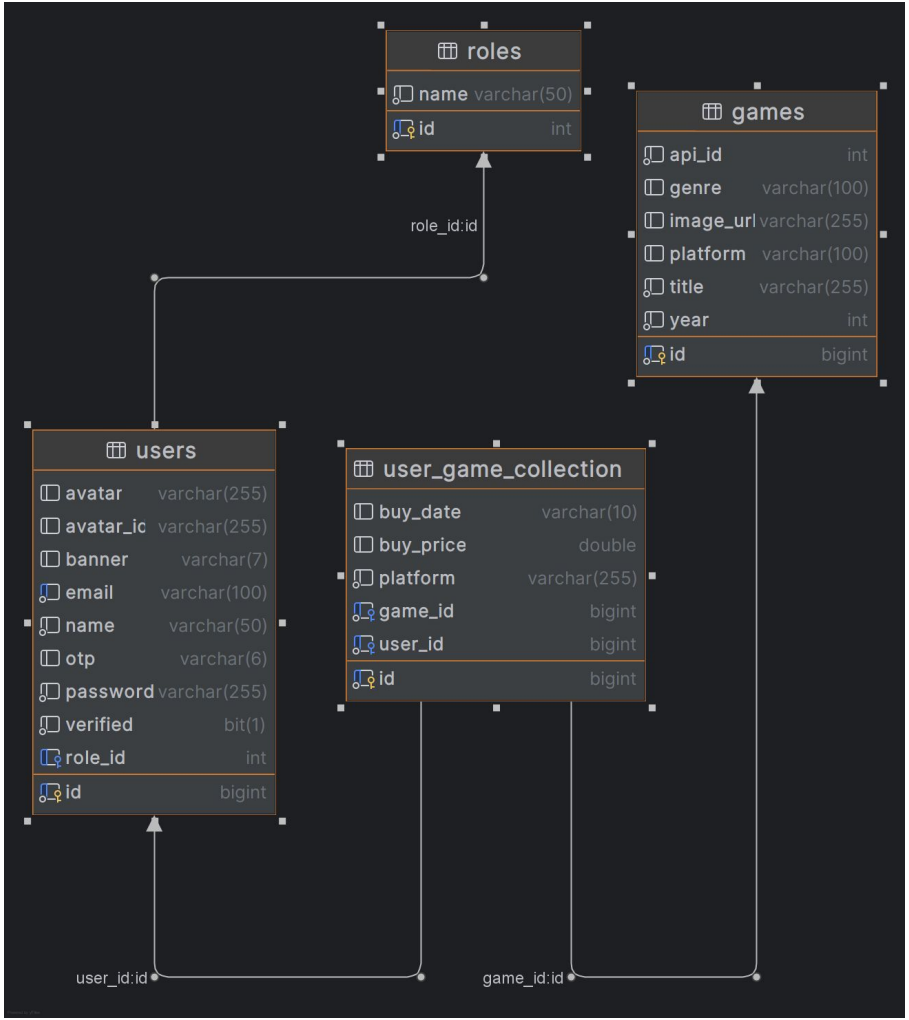
La idea inicial fue crear **Microservicios**.

Se encuentra alojado en una **instancia** de **AWS** la cual escucha en los puertos 22, 80 y 443 los cuales nos sirven tanto para realizar consultas como para conectarnos mediante **ssh** con una **pair-key** y cuenta con Certificado SSL**.**

MicroServicios

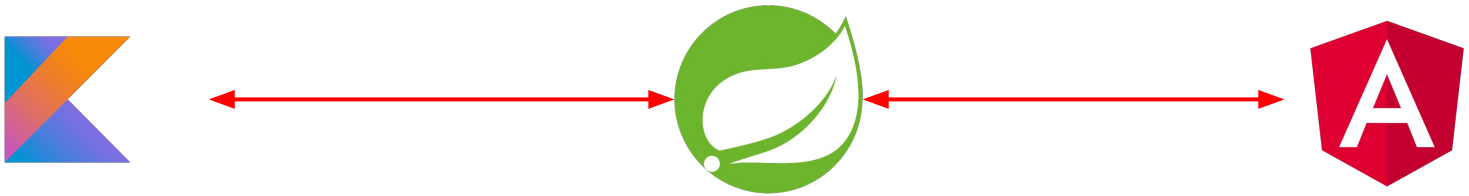
- Control de usuarios
- Servicio de Mailing
- JWT (Json Web Token - Security)
- CRUD Colección

Modelo de datos



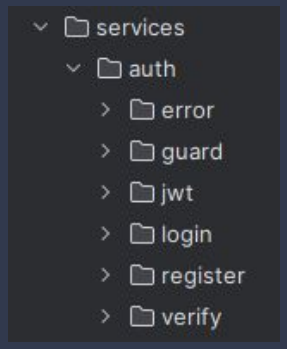
Funcionamiento

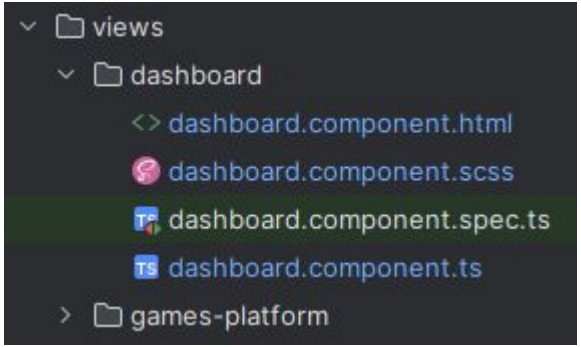
El backend se encuentra alojado en una instancia **EC2**, el cual apunta a la variante de mi dominio “api.t1tus.dev” mediante el servicio **Route 53**. Se comunica con el motor de **Aurora RDS (Mysql)** que, previamente, se configuró para con las credenciales necesarias para su correcto funcionamiento.



En la **instancia EC2** se instaló una distribución de **Linux 22.04 LTS** con **Nginx** para facilitar su configuración, se realizó la subida de la versión final del .jar del backend, y se encuentra corriendo en un service.
La dist del Frontend está subida a un **Bucket S3** que a su vez se conecta con un servicio **Cloudfront** para obtener un enlace certificado. (https)
La App se mueve en local mediante una **.APK (10.0.0.2)**
Tanto el front como la app se comunican mediante “**https://api.t1tus.dev**”

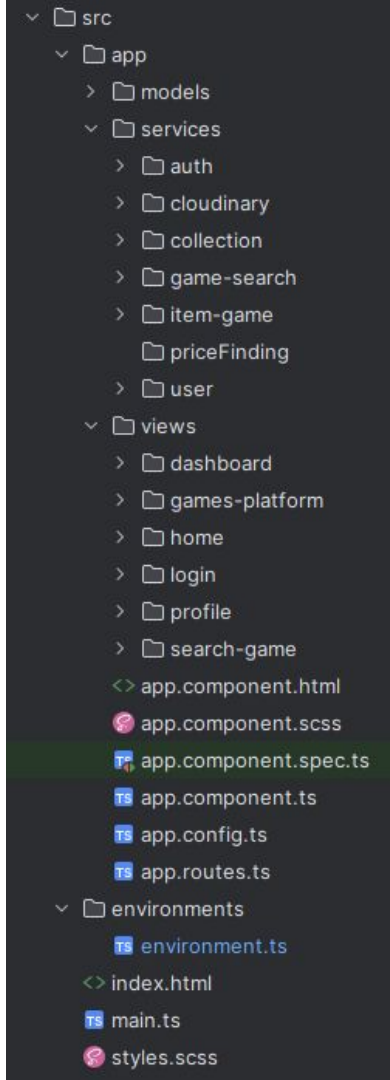
Frontend





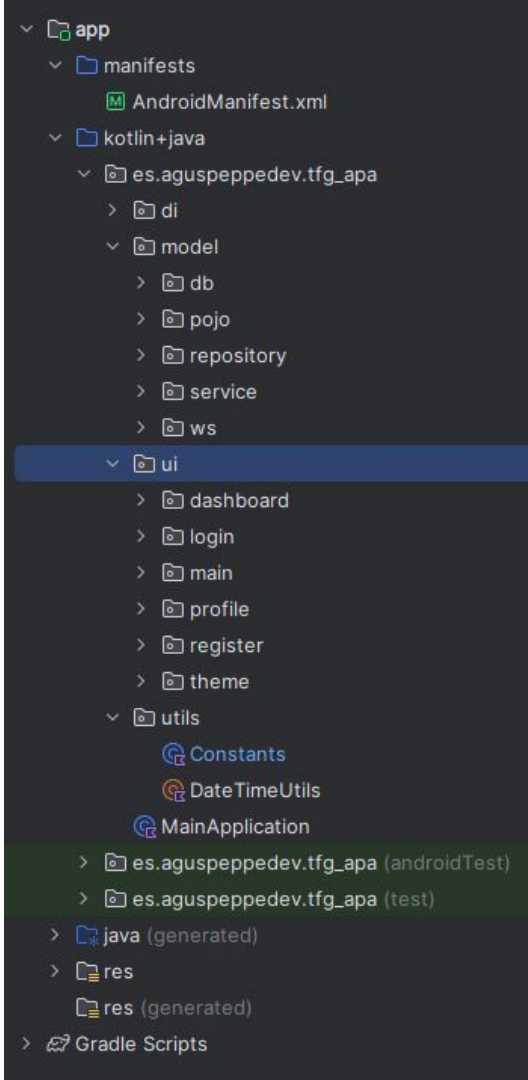

```
export const routes: Routes = [  Show usag
  {
    path: '',
    component: HomeComponent,
    title: 'Tu Colección'
  },
  {
    path: 'login',
    component: LoginComponent,
    title: 'Página de Inicio de sesión'
  },
  {
    path: 'dashboard',
    component: DashboardComponent,
    title: 'Página de usuario',
    canActivate: [AuthGuard]
  },
  {
    path: 'platform/:platformName',
    component: GamesPlatformComponent,
    title: 'Juegos',
    canActivate: [AuthGuard]
  },
  {
    path: 'profile',
    component: ProfileComponent,
    title: 'Perfil del usuario',
    canActivate: [AuthGuard]
  },
  {
    path: '**',
    redirectTo: '',
    title: 'Página de inicio'
  }
];
```

```
@Injectable({  Show usages  ▲ agustinpeppe90 *
  providedIn: 'root',
})
export class GameSearchService {
  private apiUrl : string  = 'https://api.rawg.io/api/games'; // API de RAWG

  constructor(private http: HttpClient) {}  no usages  ▲ agustinpeppe90

  searchGames(query: string): Observable<any> {  Show usages  ▲ agustinpeppe90 *
    const url = `${this.apiUrl}?key=${Environment.RAWG_API_KEY}&search=${query}&ordering=-rating`;
    return this.http.get<any>(url).pipe(
      map(response : any  =>
        response.results.flatMap((game: any) : any  =>
          game.platforms?.map((p: any) : {apiId: any; title: any; platform: any; ...  => ({
            apiId: game.id,
            title: game.name,
            platform: p.platform.name,
            year: game.released ? parseInt(game.released.substring(0, 4)) : null,
            genre: game.genres?.map((g: any) : any  => g.name).join(', ') || 'Unknown',
            imageUrl: game.background_image || '',
          })) || []
        )
      ),
      catchError((error : any ) : never  => {
        console.error('Error al buscar juegos:', error);
        throw error;
      })
    );
  }
```

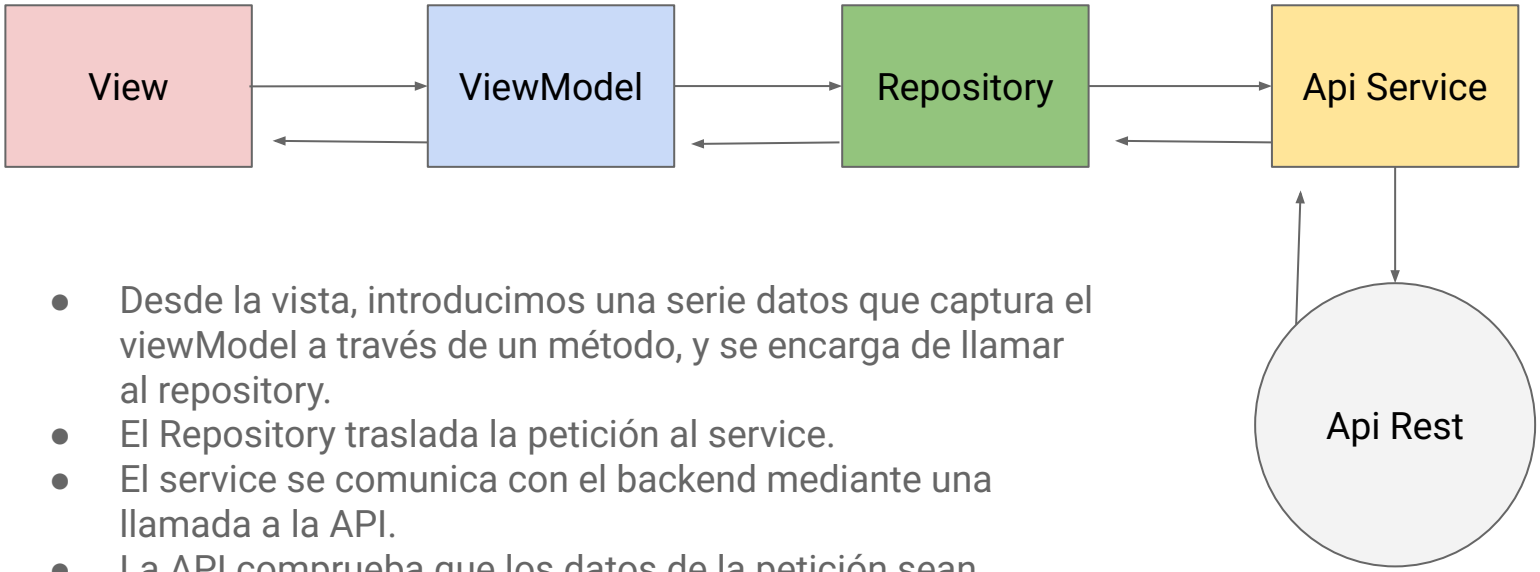


Android Kotlin

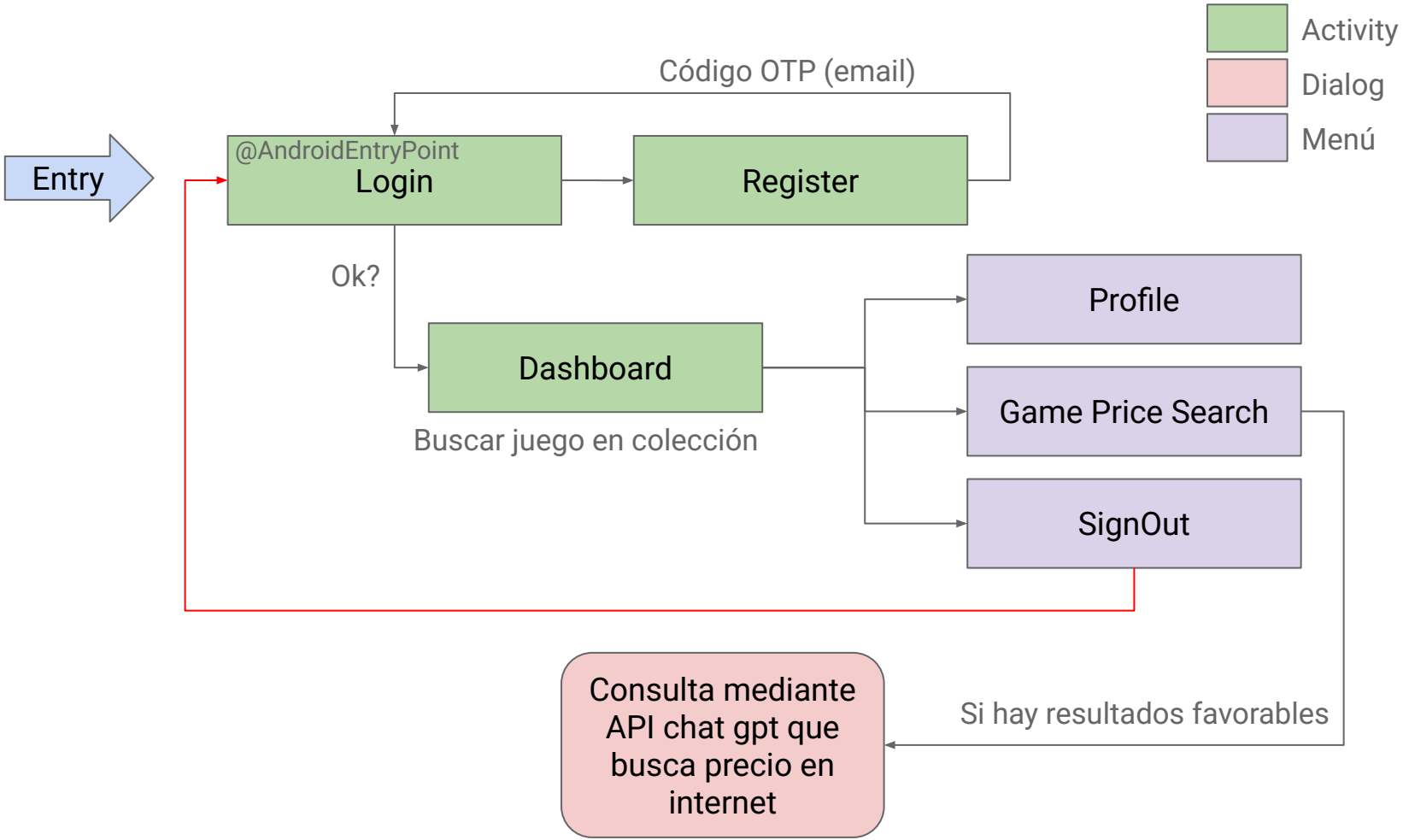
- MVVM
- DI
- DaggerHilt
- Retrofit
- Gson
- Okhttp3
- SQLite



Práctica - MVVM



- Desde la vista, introducimos una serie datos que captura el viewModel a través de un método, y se encarga de llamar al repository.
- El Repository traslada la petición al service.
- El service se comunica con el backend mediante una llamada a la API.
- La API comprueba que los datos de la petición sean correctos y devuelve un response que, posteriormente se muestra en la vista.


Activity
Dialog
Menú
Código OTP (email)
Entry
@AndroidEntryPoint
Login
Register
Ok?
Dashboard
Profile
Game Price Search
SignOut
Buscar juego en colección
Consulta mediante API chat gpt que busca precio en internet
Si hay resultados favorables

16:27

Cambiar theme

Email

Contraseña

Iniciar sesión

¿Todavía no tienes una cuenta?

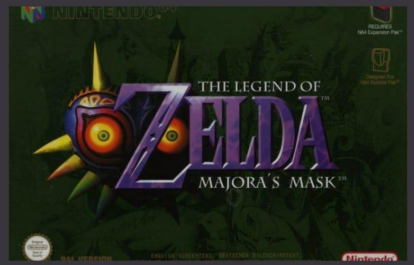
Registrarse

16:28

Mi Colección

Bienvenido, Titus

Buscar juego...



The Legend of Zelda: Majora's Mask
Año: 2000
Género: Adventure, Action
Plataforma: Nintendo 64

Resident Evil
Año: 1996

16:28

Mi Colección

Bienvenido, Titus

Buscar juego

Introduce el nombre del juego

rule of rose

Minimo 3 caracteres

El precio medio de *Rule of Rose* es:

- Wallapop: 300€
- Segundamano: 280€

Salir Buscar

The Legend of Zelda: Majora's Mask
Año: 2000
Género: Adventure, Action
Plataforma: Nintendo 64

rose Rose rosé

q w e r t y u i o p
a s d f g h j k l ñ
z x c v b n m
?123 , .

Enlaces

- **Web:** https://do7gh2ohskl8p.cloudfront.net
- **Backend:** https://github.com/t1tu-portada/tfg-apa
- **Frontend:** https://github.com/t1tu-portada/tfg-frontend-peppe
- **App:** https://github.com/t1tu-portada/tfg-apa-app
- **Games:** https://api.t1tus.dev/games

Fuentes:

https://micoleccciondejuegos.com/
https://spring.io/projects/spring-boot
https://angular.dev/
https://aws.amazon.com/es/

Mejoras

- Interfaz (imagenes plataformas)
- Generar juego mediante código de barras
- App funcione sin internet e incluya las gestiones de alta como el front
- Social
- Sistema de logros
- Más comentarios (código)
- Mejor manejo de errores