Diseño API REST

[TA046] Ingeniería de Software I

2do Cuatrimestre de 2024

Alumno:	Slepowron Majowiecki, María Mercedes
Padrón:	109454
Email:	mslepowron@fi.uba.ar

El siguiente documento presenta los puntos generales de definición para el diseño de una API REST para una aplicación móvil de calificación de películas. La aplicación busca proveer a los usuarios con la posibilidad de calificar películas, buscar películas recomendadas e interactuar con otros usuarios.

Definición de los Endpoints de la API REST

Usuario Estándar

1. Login

***Descripción**:* permite que los usuarios inicien sesión.

Request:

- Headers: Authorization: Basic {base64(email:password)}
- **POST** /api/v1/users/token

Response:

Código HTTP: 200 OK

```
{
    “access_token:  “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...”,
    “refresh_token”: “m9obiBEb2UiLCJpYXQiOjE1MTYyMzkwMj...”
}
```

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

2. Refresh de Token

***Descripción**:* permite al usuario autenticado obtener un nuevo Access_token con un refresh_token

Request:

- **POST** /api/v1/users/token/refresh
- **Body:**

```
{
    “refresh_token”: “m9obiBEb2UiLCJpYXQiOjE1MTYyMzkwMj...”
}
```

Response:

Código HTTP: 200 OK

```
{
    “access_token:  “neweyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...”,
    “refresh_token”: “newm9obiBEb2UiLCJpYXQiOjE1MTYyMzkwMj...”
}
```

Código HTTP: 400 Bad Request si el refresh token no esta en el body de la solicitud

Código HTTP: 401 Unauthorized si el refresh token es invalido, ha expirado o ya se usó

3. User Profile Info

Descripcion: consultar información del perfil de usuario autenticado

Request:

- Headers: Authorization: Bearer {access_token}
- **GET** /api/v1/users/current

Response

Codigo HTTP: 200 OK

{

"id": 15654,

“username” “ismarypoppins”,

"email": "user@example.com",

“first_name”: “Mary”,

“last_name”: “Poppins”,

“profile_picture”: "https://userprofile.com/profile.jpg",

“birthdate”: “2003-03-12”,

“gender”: “Female”

}

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

Esta acción requiere autenticación porque el usuario debe estar loggeado para realizarla.

4. Modify User Profile Info

Descripción: actualizar la información del perfil de usuario autenticado

Request:

- Headers: Authorization: Bearer {access_token}
- **PUT** /api/v1/users/current
- Body:

{

“username” “**mynewusername**”,

"email": "user@example.com",

“first_name”: “Mary”,

“last_name”: “Poppins”,

“profile_picture”: "https://userprofile.com/profile.jpg",

“birthdate”: “2003-03-12”,

“gender”: “Female”

}

Response

Codigo HTTP: 200 OK

{

“message”: “Profile updated successfully”

}

Código HTTP: 400 Bad Request si el formato de los datos es incorrecto

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

5. Movie Rating

Descripcion: calificar películas del 1 al 10

Request:

- Headers: Authorization: Bearer {access_token}

POST /api/v1/movies/{movie_name}/ratings

- Query Parameter: director={movie_director}

Se agrega este parametron a la query para refinar la búsqueda, ya que puede ser que haya más de una película con el mismo nombre.

- Body:

{

“rating” 9

}

Response

Codigo HTTP: 201 Created

{

“message”: “Profile has been deleted successfully”,

“movie_id”: 51526,

“movie_name”: “Harry Potter and the Prisoner of Azkaban”

“rating”: 9

}

Código HTTP: 400 Bad Request el rating no está entre 1 y 10 puntos

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

Código HTTP 404 Not Found si la película no se encuentra en la base.

6. User Search

Descripcion: Buscar el perfil de otro usuario

Request:

- Headers: Authorization: Bearer {access_token}
- **GET** /api/v1/users/serch
- Query Parameters: username

Response

Codigo HTTP: 200 OK

```
{
    “username” “searchedusername”,
    "email": "user@example.com",
    “first_name”: “Jon”,
    “last_name”: “Doe”,
    “profile_picture”: "https://userprofile.com/profile.jpg",
}
```

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

Código HTTP 404 Not Found si el usuario buscado no existe.

7. Follow User

Descripcion: Solicitar seguimiento de perfil de otro usuario

Request:

- Headers: Authorization: Bearer {access_token}
- **POST** /api/v1/users/follow

Response

Codigo HTTP: 200 OK

```
{
    “message” “Your follow request has been sent!”,
}
```

Código HTTP: 400 Bad Request si ya sigue al usuario

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

Código HTTP 404 Not Found si la película no se encuentra en la base.

8. Search rated movies by user

Descripcion: Buscar las películas calificadas por un usuario

Request:

- Headers: Authorization: Bearer {access_token}
- **GET** /api/v1/users/{username}/rated_movies

Se asume que no puede haber dos usuarios con el mismo nombre de usuario.

Response

Codigo HTTP: 200 OK

```
{
    “rated_movies”: [
    {
        “movie_name”: “IT”,
        “movie_director”: “Andrés Muschietti”,
        “movie_rating”: 8
    },
    {
        “movie_name”: “Hocus Pocus”,
        “movie_director”: “Kenny Ortega”,
        “movie_rating”: 10
    }
    ]
}
```

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

Código HTTP 404 Not Found si el usuario buscado no se encuentra.

9. Accept Follow request

Descripcion: Aceptar/rechazar una solicitud de seguimiento de otro usuario

Request:

- Headers: Authorization: Bearer {access_token}
- **PUT** /api/v1/users/follow_request/{username_request}
- **Body:**

```
{
    “status”: “accepted” //or “rejected”
}
```

Response

Codigo HTTP: 200 OK

```
{
        “message” “You have a new follower!”,
        “request_username”: “username”
}
```

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

Código HTTP 404 Not Found si la solicitud no existe

10. Check user’s followers

Descripcion: Ver la lista de seguidores de un usuario

Request:

- Headers: Authorization: Bearer {access_token}
- **GET** /api/v1/users/current/followers

Response

Codigo HTTP: 200 OK

```
{
        “followers”: [
        {
                “username”: “itsmemario”,
                “name”: “Mario”,
                “last_name”: “Mario”,
                “profile_picture”: "https://example.com/mario.jpg"
        },
        {
                “username”: “itsluigi”,
                “name”: “Luigi”,
                “last_name”: “Mario”,
                “profile_picture”: "https://example.com/luigi.jpg"
        }
        ]
}
```

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

11. Check user's following

Descripcion: Ver la lista de seguidos de un usuario

Request:

- Headers: Authorization: Bearer {access_token}
- **GET** /api/v1/users/current/followed

Response

Codigo HTTP: 200 OK

```
{
    "followed": [
    {
        "username": "jamesbond1",
        "name": "James",
        "last_name": "Bond",
        "profile_picture": "https://example.com/james007.jpg"
    },
    {
        "username": "teloresumo",
        "name": "Jorge",
        "last_name": "Pinarello",
        "profile_picture": "https://example.com/avatar.jpg"
    }
    ]
}
```

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

12. Delete User Profile

Descripcion: eliminar el perfil de usuario autenticado de la aplicación.

Request:

- Headers: Authorization: Bearer {access_token}
- **DELETE** /api/v1/users/current
- Body:

```
{
    "username" "myusername",
    "email": "user@example.com",
```

“password”: “userpassword”

“profile_deletion_verifier”: “Yes”

}

Se le solicita al usuario que ingrese sus datos antes de eliminar la cuenta como punto de validación extra.

Response

Codigo HTTP: 200 OK

{

“message”: “Profile has been deleted successfully”

}

Código HTTP: 400 Bad Request si el formato o contenido de los datos es incorrecto

Código HTTP: 401 Unauthorized si el token es invalido o esta expirado

13. Search recommended movies

Descripcion: eliminar el perfil de usuario autenticado de la aplicación.

Request:

- **GET** /api/v1/movies/recommendations
- **Query parameters (opcionales)** movie title, actor name, movie category, page, page_size

Donde page es la pagina de resultados que se tiene que obtener (por ejemplo la 1), y page_size la cantidad de resultados por página.

- Body:

{

“username” “**myusername**”,

"email": "user@example.com",

“password”: “userpassword”

“profile_deletion_verifier”: “Yes”

}

Se le solicita al usuario que ingrese sus datos antes de eliminar la cuenta como punto de validación extra.

Response

Codigo HTTP: 200 OK

{

“recommended_movies”: [

{

“movie_name”: “IT”,

“movie_director”: “Andrés Muschietti”,

“general_rating”: 6.4,

“categories”: [Horror, Mystery],

“actors”: [“Finn Wolfhard”, “Bill Skarsgard”, “Jaeden Martell”]

},

{

“movie_name”: “Hocus Pocus”,

“movie_director”: “Kenny Ortega”,

“general_rating”: 6.9,

“categories”: [Fantasy, Adventure, Horror Comedy, Family],

“actors”: [“Sarah Jessica Parker”, “Thora Birch”, “Omri Katz”, “Kathy Najimy”]

}

]

}

Código HTTP: 400 Bad Request si el formato de los parámetros es erróneo

Código HTTP: 404 Not Found si según los parámetros de búsqueda no se obtuvo ningún resultado en la base.

Usuario Administrador

1. Admin Login

Descripción*:* permite que los usuarios con rol de administrador inicien sesión.

Request:

- Headers: Authorization: Basic {base64(email:password)}
- **POST** /api/v1/admin/token

Response:

Código HTTP: 200 OK

{

“access_token: “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...”,

“refresh_token”: “m9obiBEb2UiLCJpYXQiOjE1MTYyMzkwMj...”

}

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

2. Crear, editar y eliminar otros usuarios de rol admin

***Descripción**:* permite que los usuarios con rol de administrador creen, editen o eliminen a otros usuarios de rol admin.

Crear:

Request:

- Headers: Authorization: Bearer {access_token}
- **POST** /api/v1/admin/users
- **Body :**

{

“username” “newadmin”,

"email": "adminuser@example.com",

“first_name”: “Mike”,

“last_name”: “Williams”,

“password”: “adminpassword123”

}

Response:

Código HTTP: 201 Created

{

“message”: “New admin user has been created”,

“admin_id”: “15462”

}

Código HTTP: 400 Bad Request si alguno de los campos de creación de administrador es incorrecto.

Código HTTP: 401 Unauthorized si las credenciales son incorrectas (ejemplo: el Access token no es de un admin)

Editar:

Request:

- Headers: Authorization: Bearer {access_token}
- **PATCH** /api/v1/admin/users
- **Query parameters: admin_user_id**
- **Body :**

{

“username” “mikeadminuser”,

}

Response:

Código HTTP: 200 OK

{

"message": "Admin updated successfully."

}

Código HTTP: 404 Not Found si se buscó un admin user inexistente

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

Eliminar:

Request:

- Headers: Authorization: Bearer {access_token}
- **DELETE** /api/v1/admin/users
- **Query parameters: admin_user_id**

Response:

Código HTTP: 200 OK

{

"message": "Admin user has been deleted."

}

Código HTTP: 404 Not Found si se buscó un admin user inexistente

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

3. Crear, editar y eliminar categorías de peliculas

Descripción*:* permite que los usuarios con rol de administrador creen, editen o eliminen películas.

Crear:

Request:

- Headers: Authorization: Bearer {access_token}
- **POST** /api/v1/admin/movies/categories
- **Body :**

{

"category_name" "Animation",

}

Response:

Código HTTP: 201 Created

{

"message": "New category has been created",

"category_id": "15462"

}

Código HTTP: 400 Bad Request si la categoría ya existe

Código HTTP: 401 Unauthorized si las credenciales son incorrectas (ejemplo: el Access token no es de un admin)

Editar:

Request:

- Headers: Authorization: Bearer {access_token}
- **PATCH** /api/v1/admin/movies/categories
- **Query parameters: category_name**
- **Body :**

{

“category_name” “Animation for kids”,

}

Response:

Código HTTP: 200 OK

{

“message”: “category updated successfully.”

}

Código HTTP: 404 Not Found si se buscó una categoría inexistente

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

Eliminar:

Request:

- Headers: Authorization: Bearer {access_token}
- **DELETE** /api/v1/admin/movies/categories
- **Query parameters: admin_user_id**

Response:

Código HTTP: 200 OK

{

“message”: “category has been deleted.”

}

Código HTTP: 404 Not Found si se buscó una categoría inexistente

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

4. Crear, editar y eliminar actores

Descripción*:* permite que los usuarios con rol de administrador creen, editen o eliminen actores

Crear:

Request:

- Headers: Authorization: Bearer {access_token}
- **POST** /api/v1/admin/actors
- **Body :**

{

“name”: “Cillian”,

“last_name”: “Murphy”,

“movies”: [Batman Begins, Oppenheimer, Inception],

}

Response:

Código HTTP: 201 Created

{

“message”: “New actor has been added”,

“actor_id”: “15462”

}

Código HTTP: 401 Unauthorized si las credenciales son incorrectas (ejemplo: el Access token no es de un admin)

Editar:

Request:

- Headers: Authorization: Bearer {access_token}
- **PATCH** /api/v1/admin/actors/{actor_id}
- **Query parameters: actor_name, actor_last_name**
- **Body :**

{

“movies”: [**Dunkirk,** Batman Begins, Oppenheimer, Inception],

}

Response:

Código HTTP: 200 OK

{

“message”: “actor data has been updated successfully.”

}

Código HTTP: 404 Not Found si se buscó un actor no registrado

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

Eliminar:

Request:

- Headers: Authorization: Bearer {access_token}
- **DELETE** /api/v1/admin/actors/{actor_id}
- **Query parameters: actor_name, actor_last_name**

Response:

Código HTTP: 200 OK

{

“message”: “actor has been deleted.”

}

Código HTTP: 404 Not Found si se buscó un actor inexistente

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

5. Crear, editar y eliminar películas, asignando actores y categorías (sin modificar las calificaciones de los usuarios).

Descripción*:* permite que los usuarios con rol de administrador creen, editen o eliminen películas asignando actores y categorias

Crear:

Request:

- Headers: Authorization: Bearer {access_token}
- **POST** /api/v1/admin/movies
- **Body :**

{

“movie_name”: “IT”,

“movie_director”: “Andrés Muschietti”,

“general_rating”: 6.9,

“categories”: [Horror, Mystery],

“actors”: [“Finn Wolfhard”, “Bill Skarsgard”, “Jaeden Martell”]

}

Response:

Código HTTP: 201 Created

{

“message”: “New movie added to platform”,

“movie_id”: “15462”

}

Código HTTP: 401 Unauthorized si las credenciales son incorrectas (ejemplo: el Access token no es de un admin)

Editar:

Request:

- Headers: Authorization: Bearer {access_token}
- **PATCH** /api/v1/admin/movies
- **Query parameters: movie_name, movie_director**
- **Body :**

{

“movie_name”: “IT”,

“movie_director”: “Andrés Muschietti”,

“general_rating”: 6.9,

“categories”: [Horror, Mystery, **Suspense, Science Fiction**],

“actors”: [“Finn Wolfhard”, “Bill Skarsgard”, “Jaeden Martell”]

}

Response:

Código HTTP: 200 OK

{

“message”: “movie data has been updated successfully.”

}

Código HTTP: 404 Not Found si se buscó una pelicula no registrada

Código HTTP: 401 Unauthorized si las credenciales son incorrectas

Eliminar:

Request:

- Headers: Authorization: Bearer {access_token}
- **DELETE** /api/v1/admin/movies
- **Query parameters: movie_name, movie_director**

Response:

Código HTTP: 200 OK

{

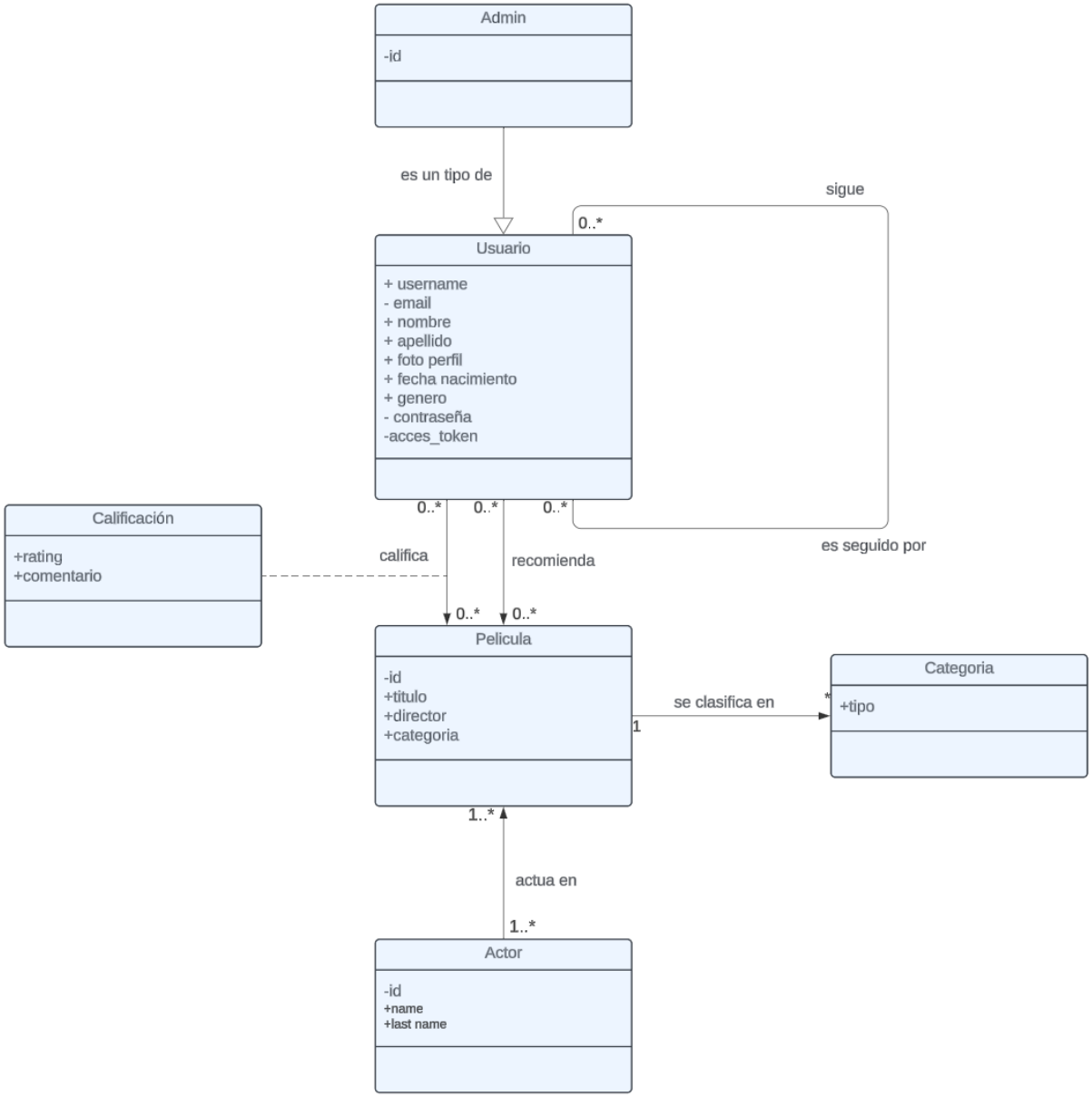
“message”: “movie has been deleted.”

}

Código HTTP: 404 Not Found si se buscó una pelicula inexistente

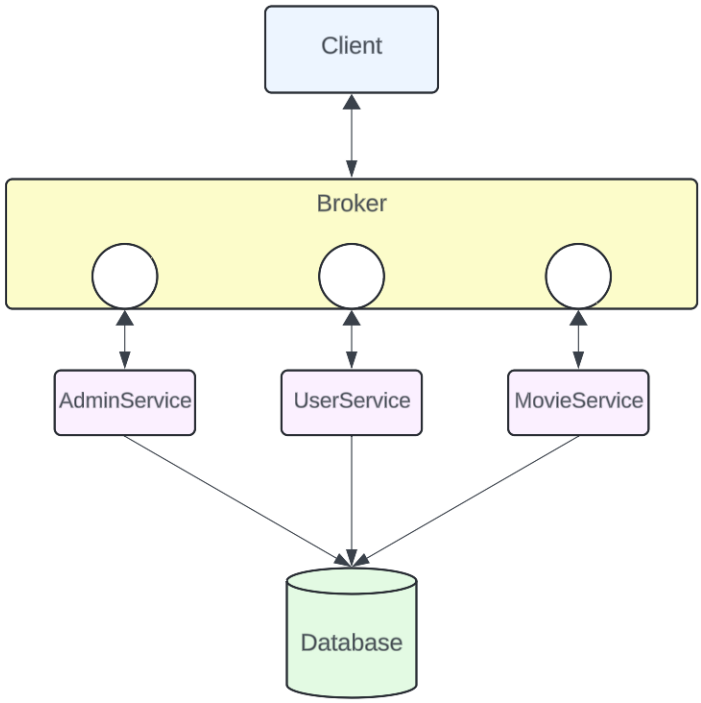
Código HTTP: 401 Unauthorized si las credenciales son incorrectas

Definición de Modelo de Dominio



Definición de Arquitectura

Para la presente aplicación se define el modelado del sistema a través de una arquitectura ***Broker.***



Elementos de la arquitectura

1. **Cliente (capa de presentación)**

Usuarios que van a consumir los servicios a través de la app y van a realizar solicitudes al sistema.

2. **Broker**

Se va a encargar de intermediar la comunicación entre los clientes y los servidores. Este agente también se encarga de las acciones de verificación frente a las actividades de usuario. Por ejemplo, un usuario estándar (no admin) no puede realizar una solicitud al AdminService para, por ejemplo, agregar una nueva película a la app.

3. **Servidores (capa de negocio)**

Por el momento la aplicación, en esta primera versión, se define con 3 servidores.

a. ***AdminService***: servidor que gestiona la lógica y funcionalidades de los usuarios administradores, como crear/editar/eliminar otros usuarios administradores, o agregar/editar películas, etc.
b. ***UserService***: servidor que gestiona la logia relacionada a los usuarios estándares de la aplicación (login de usuario, calificación de película, funcionalidad de seguir usuarios, etc.)
c. ***MovieService***: maneja la lógica e información relacionada con las películas registradas en la aplicación (calculo del rating de una película según las reseñas de usuarios, registro de actores, directores, fecha de lanzamiento, etc.)

4. **Base de Datos (capa de persistencia)**

Los datos de la aplicación a los que acceden los servidores se encuentran centralizados para el planteo de esta arquitectura.