Ingeniería de Software I
Mariana Juarez Goldemberg - 108441

DISEÑO DE API REST Y ARQUITECTURA
PARA APP DE PELÍCULAS

1. Definir una **API REST**.

REQUEST: POST

User estándar (no Administrador)

- **Login User**: permite que un usuario inicie sesión.

`/api/v1/users/token`

RESPONSE
200: User logueado correctamente.
401: credenciales incorrectas.

- **Registrar**: permite que un nuevo usuario se registre.

`/api/v1/users/register`

RESPONSE
201: User creado correctamente.
400: datos inválidos.

- **Calificar una película**: permite calificar una película a los usuarios logueados.

`/api/v1/movies/{movieID}/rate`

RESPONSE
200: Calificación exitosa.
401: Falta de autenticación.

- **Seguir a un usuario**: permite seguir a otro usuario a los usuarios logueados.

`/api/v1/users/{userID}/follow`

RESPONSE
200: Follow a user exitoso.
401: Falta de autenticación.

- **Aceptar o rechazar solicitudes de seguimiento**: permite que los usuarios logueados acepten o rechacen solicitudes de seguimiento.

`/api/v1/users/{userID}/requests/follow`

RESPONSE
200: Solicitud aceptada o rechazada.
401: Falta de autenticación.

User Administrador

- **Login User Admin**: permite que se loguee un usuario administrador.

`/api/v1/admin/token`

RESPONSE
200: User administrador logueado correctamente.
401: Falta de autenticación.

- **Crear nuevo administrador**: permite a un usuario administrador crear otro usuario admin.

`/api/v1/admin/users`

RESPONSE
200: User administrador creado correctamente.
401: Falta de autenticación.

- **Crear nueva categoría de película**: permite a un usuario administrador crear una nueva categoría de peículas.

`/api/v1/admin/movieTag`

RESPONSE
200: Categoría de película creada correctamente.
401: Falta de autenticación.

- **Crear actor**: permite a un usuario administrador crear un nuevo actor.

`/api/v1/admin/actors`

RESPONSE
200: Actor creado correctamente.
401: Falta de autenticación.

- **Crear película**: permite a un usuario administrador crear una nueva película.

`/api/v1/admin/movies`

RESPONSE
200: Película creada correctamente.
401: Falta de autenticación.

REQUEST: GET

- **Obtener películas recomendadas**: permite a cualquier usuario buscar recomendaciones de películas.

`/api/v1/movies/recommendations`

RESPONSE
200: Películas encontradas.
401: Falta de autenticación.

- **Buscar usuarios**: permite a los usuarios buscar a otros a partir de su username.

`/api/v1/users/search`

RESPONSE
200: Usuarios encontrados.
401: Falta de autenticación.

- **Ver calificaciones**: permite que un usuario pueda ver las calificaciones de otro.

`/api/v1/users/{userID}/ratings`

RESPONSE
200: Calificaciones encontradas.
401: Falta de autenticación.

- **Ver seguidores**: permite a un usuario ver su propia lista de seguidores.

`/api/v1/users/{userID}/followers`

RESPONSE
200: Seguidores encontrados.
401: Falta de autenticación.

- **Ver seguidos**: permite a un usuario ver a los otros que sigue.

`/api/v1/users/{userID}/followedBy`

RESPONSE
200: Seguidos encontrados.
401: Falta de autenticación.

- **Ver perfil de usuario**: permite a un usuario ver su propio perfil.

`/api/v1/users/{userID}`

RESPONSE
200: Perfil de usuario encontrado.
401: Falta de autenticación.

Ingeniería de Software I
Mariana Juarez Goldemberg - 108441

REQUEST: PUT

User estándar (no Administrador)

- **Editar usuario**: permite a un usuario editar su perfil.

`/api/v1/users/{userID}/edit`

RESPONSE
200: Usuario editado correctamente.
401: Falta de autenticación.

User Administrador

- **Editar categoría**: permite a un usuario administrador editar una categoría de película.

`/api/v1/admin/categories/edit`

RESPONSE
200: Categoría editada correctamente.
401: Falta de autenticación.

- **Editar actor**: permite a un usuario administrador editar un actor.

`/api/v1/admin/actors/edit`

RESPONSE
200: Actor editado correctamente.
401: Falta de autenticación.

- **Editar película**: permite a un usuario administrador editar una película.

`/api/v1/admin/movies/edit`

RESPONSE
200: Película editada correctamente.
401: Falta de autenticación.

REQUEST: DELETE

User estándar (no Administrador)

- **Eliminar usuario**: permite a un usuario eliminar su propio perfil.

`/api/v1/user/{userID}/delete`

RESPONSE
200: Eliminación exitosa.
401: Falta de autenticación.
404: Usuario no encontrado.

User Administrador

- **Eliminar administrador**: permite a un usuario administrador eliminar su usuario.

`/api/v1/admin/{userID}/delete`

RESPONSE
200: Eliminación exitosa.
401: Falta de autenticación.
404: Usuario administrador no encontrado

- **Eliminar categoría**: permite a un usuario administrador eliminar una categoría de película.

`/api/v1/admin/categories/delete`

RESPONSE
200: Eliminación exitosa.
401: Falta de autenticación.

404: Categoría de película no encontrada.

- **Eliminar actor**: permite a un usuario administrador eliminar un actor.

`/api/v1/admin/actors/delete`

RESPONSE
200: Eliminación exitosa.
401: Falta de autenticación.
404: Actor no encontrado.

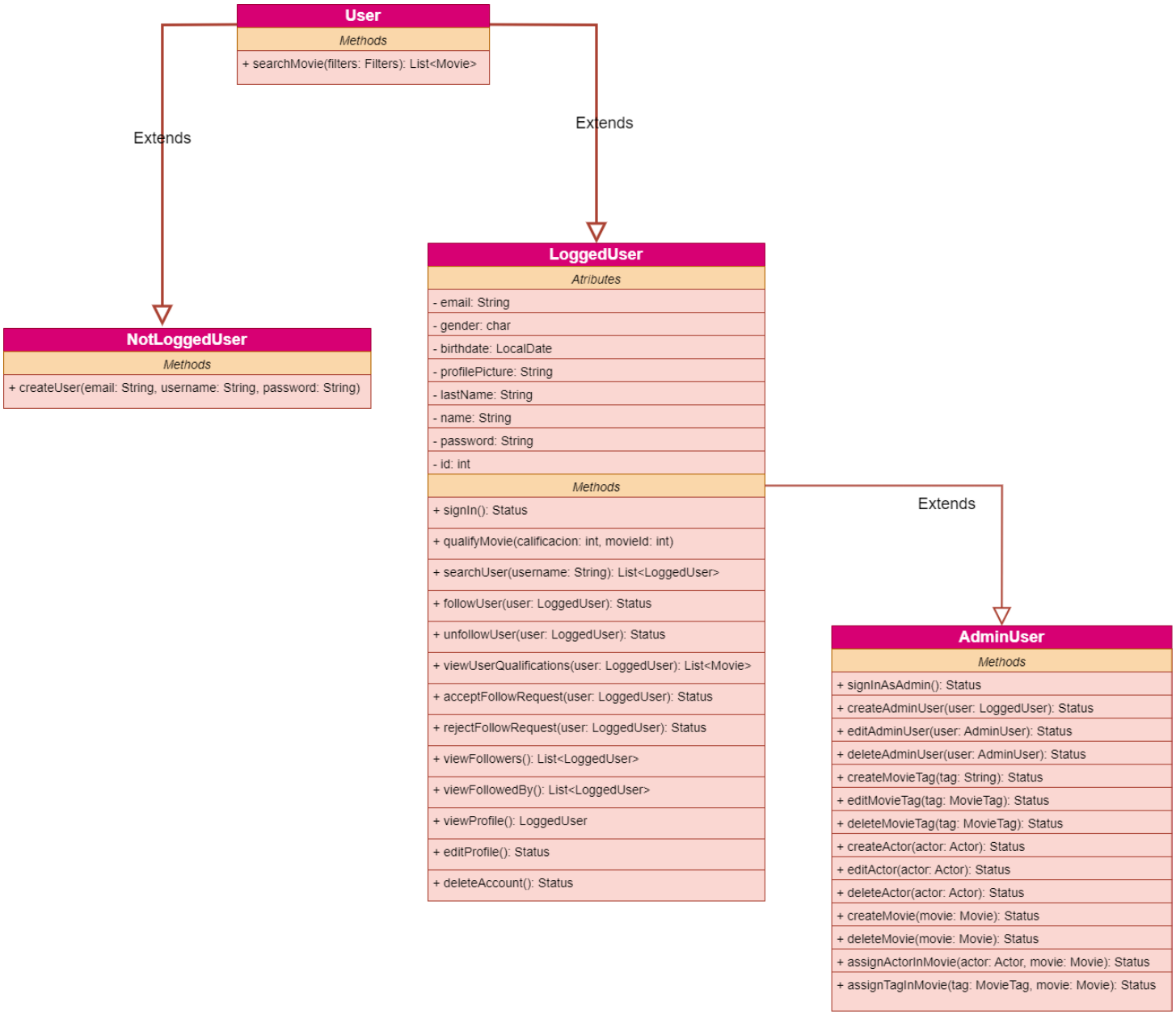
- **Eliminar película**: permite a un usuario administrador eliminar una película.

`/api/v1/admin/movies/delete`

RESPONSE
200: Eliminación exitosa.
401: Falta de autenticación.
404: Película no encontrada.

2. **Modelo de dominio** preliminar.

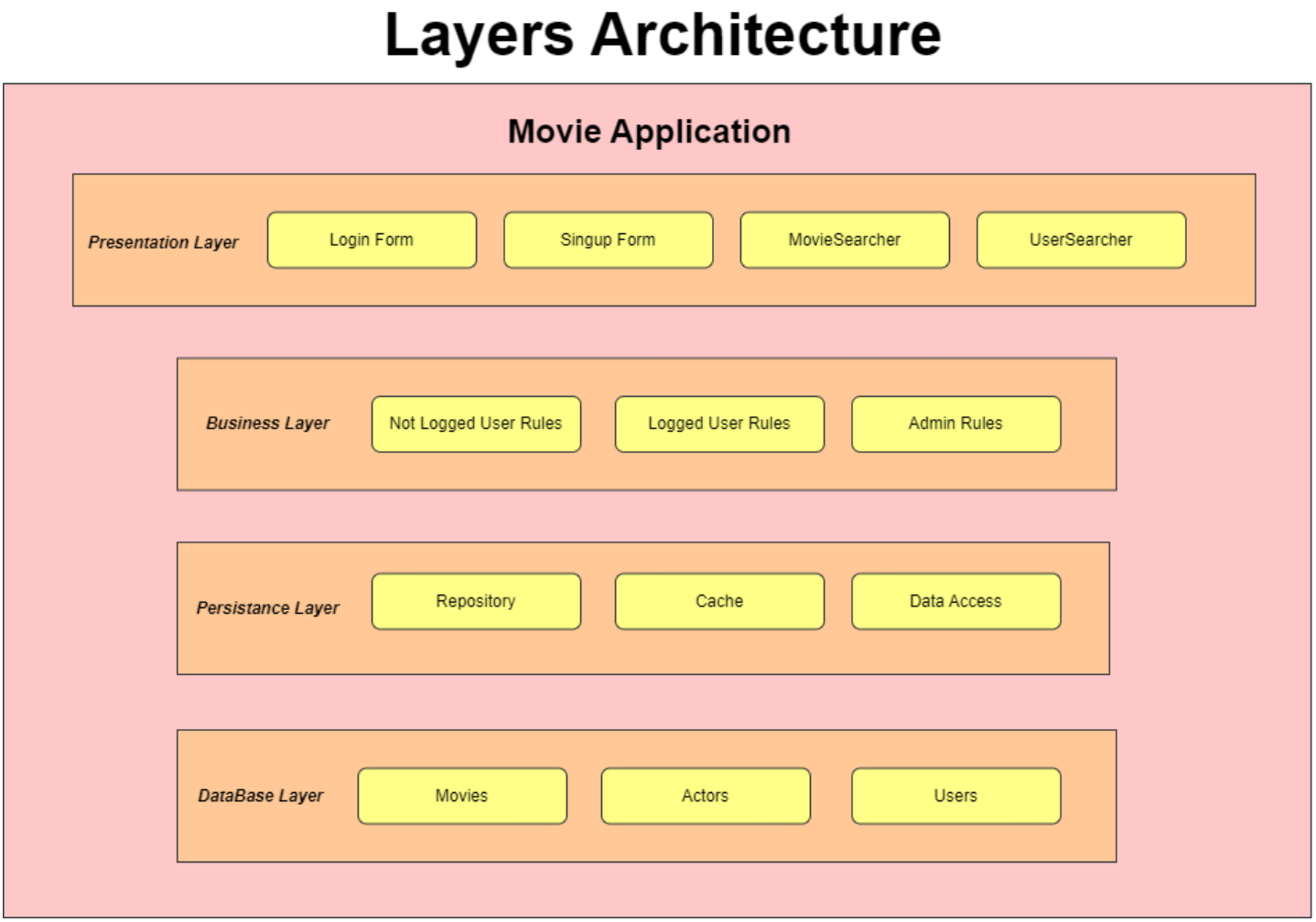
Se definió un modelo de dominio en un diagrama UML para exponer las diferentes clases de Usuario que podrían usar la API.



Además, se definen las clases Movie (que representa una película), MovieTag (que representa una categoría de películas) y Actor (que

representa un actor). Como el modelo de dominio de la API se centraba en los usuarios, no se agregaron al diagrama UML.

3. Propuesta de **arquitectura** preliminar



Se realizó el modelo de la arquitectura a partir de capas, teniendo la capa de presentación, donde se encuentra la interfaz con el usuario, los forms de registro e ingreso y los buscadores.

Más abajo nos encontramos con la capa de negocios donde establecemos las reglas que posee cada usuario para utilizar nuestra API.

Tenemos la capa de persistencia donde nos encontramos con el acceso a los datos, con un patrón Repository para ubicar el acceso a datos en la capa externa de la aplicación y así mantener el dominio

agnóstico a las fuentes. Además también se agregó un componente de caché para las búsquedas recientes.

Por último, tenemos la capa de la base de datos donde se guarda la información de la aplicación.