MOVIEROLE

Escuela técnica Superior de Ingeniería Informática

Introducción a Ingeniería de Software

Indice

Miembros y sus Roles

1. Pablo Astudillo Fraga – Modelador + Documentador
2. Jorge Camacho García – Analista + Modelador
3. Marina Sayago Gutiérrez – Product Owner + Dis. Gráfico
4. José Fco. Artacho Martín – Analista + Programador
5. Ignacio Alba Avilés – Analista + Programador
6. Antonio Fernández Rodríguez – Programador + Dis. Gráfico
7. Diego López Reduello – Product Owner + Programador
8. Iván Delgado Alba – Tester + Scrum Master
9. Manuel Jesús Jerez Sánchez – Scrum Master + Dis. Gráfico
10. Mario Merino Zapata – Tester + Documentador

Problema

- Debido a la abundancia de entretenimiento audiovisual, nuestro cliente quería que cualquier persona pudiese estar conectado a una red en la que las personas compartiesen opiniones, gustos y recomendaciones. Para poder así tener una mejor experiencia audiovisual.

Solución

- MoviErole consiste en una aplicación web enfocada al entretenimiento.
- En está se podrán consultar tanto información de la película como la opción de llevar un seguimiento personal de los contenidos.
- Además te brindamos la oportunidad de pertenecer a una comunidad donde interactuar y compartir gustos con otros usuarios.

Requisitos

RF1

Como interesado en usar Erole, quiero ser capaz de crearme un perfil con mi información personal para poder usar la aplicación.

RF3

Como usuario de la aplicación, quiero poder iniciar sesión con mi propia clave y que me brinde seguridad de que otra persona no acceda a mi cuenta.

RF7

Como usuario, quiero tener la posibilidad de realizar búsquedas de los elementos almacenados en la BD para poder encontrar los contenidos deseados de forma sencilla.

RNF2 y RNF4

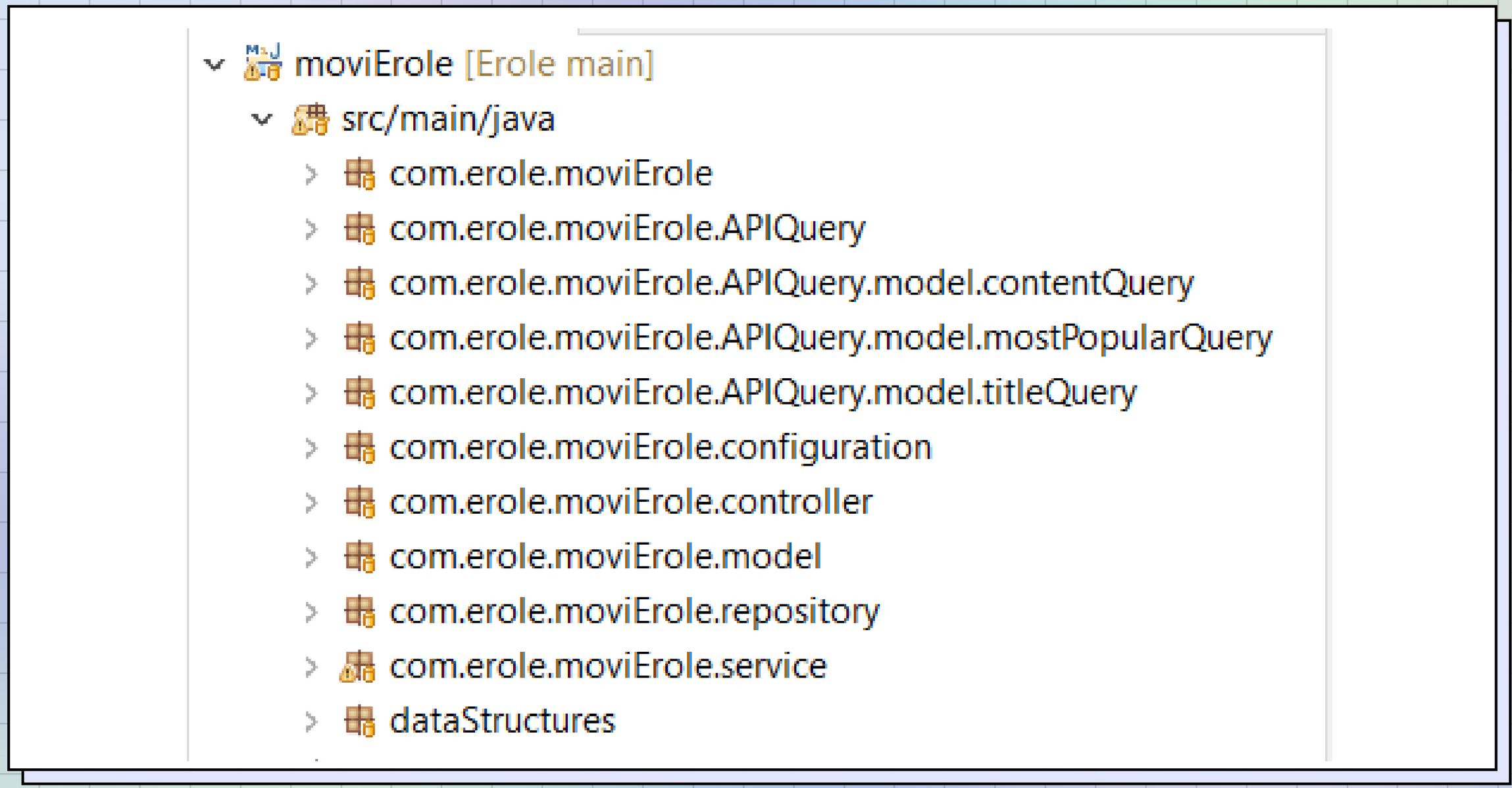
- Como gestor de la base de datos quiero que el contenido pueda estar etiquetado según género, duración, plataforma, reparto.
- Como equipo de desarrollo queremos que los datos sean importados de una base de datos externa a través de una API, para no tener que introducir los datos manualmente y tener disponibles una gran cantidad de datos actualizados

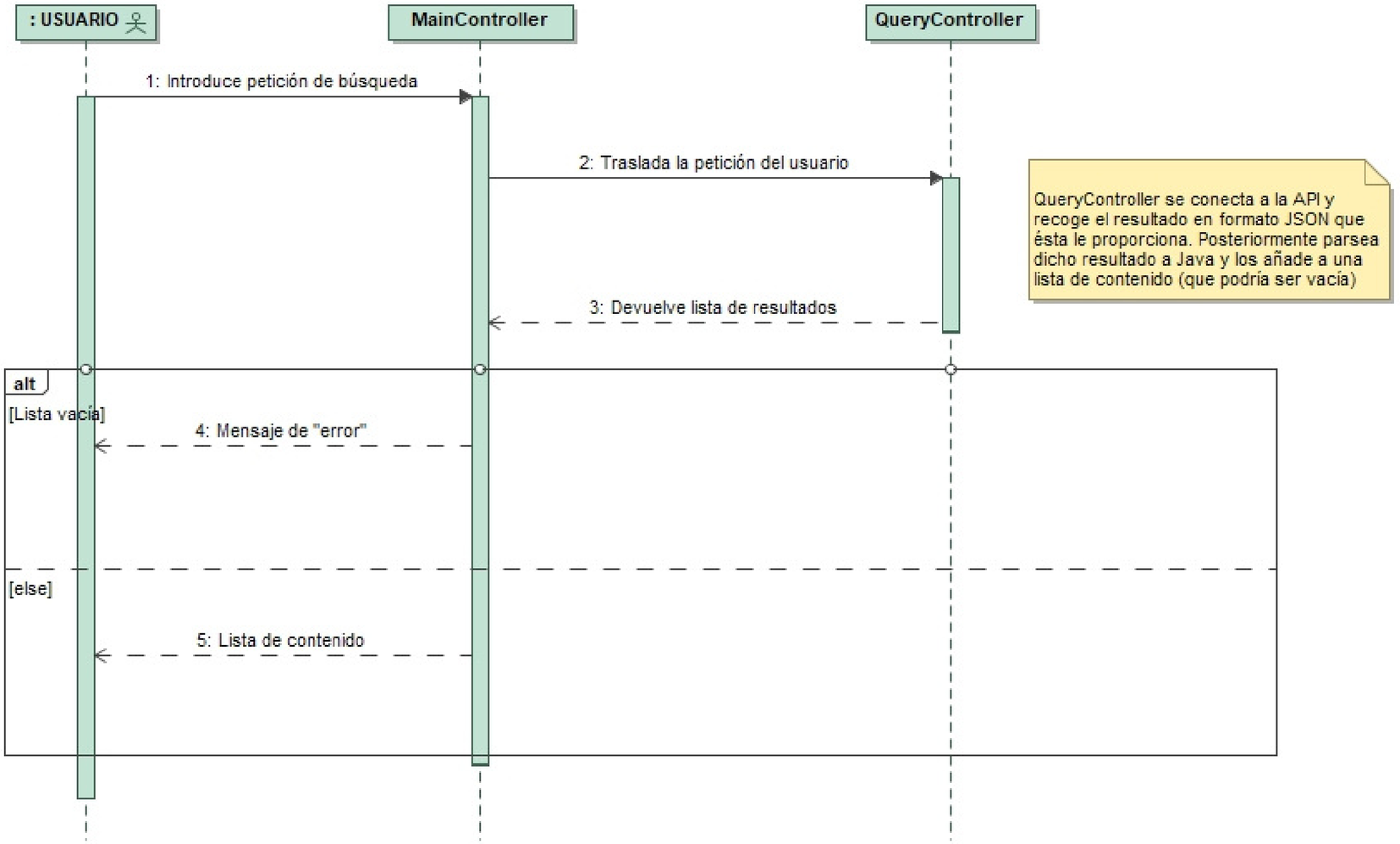
Planificación

- Finalmente, tras consultar información sobre diferentes métodos, decidimos utilizar la metodología Scrum.
- Esta metodología se basa en sprints, de los cuales hemos hecho 8 de una duración de entre una y dos semanas.
- En cada sprint hemos repartido las diferentes tareas en equipos de 2 a 3 personas favoreciendo así la productividad y la cooperación.

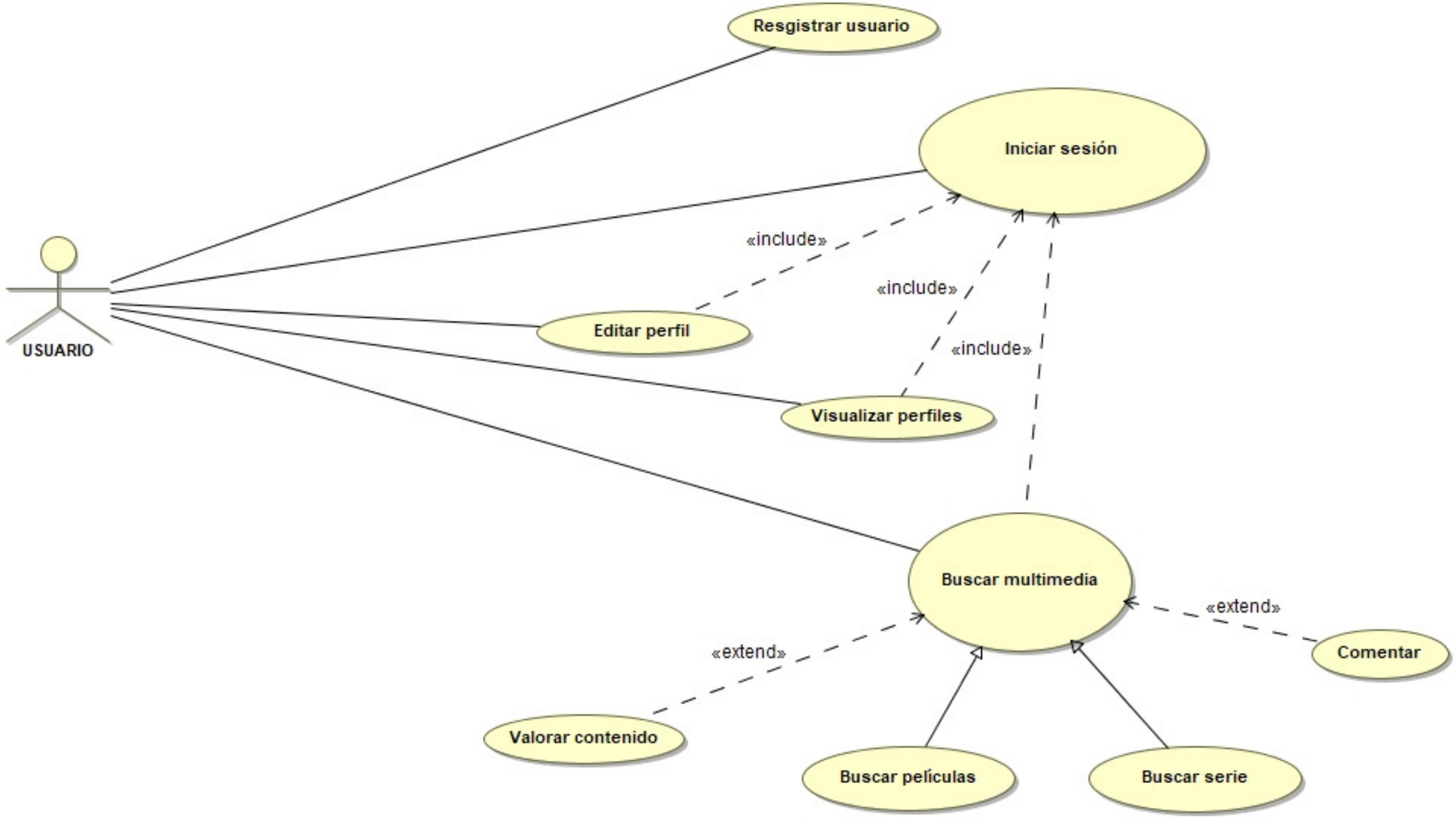
Arquitectura

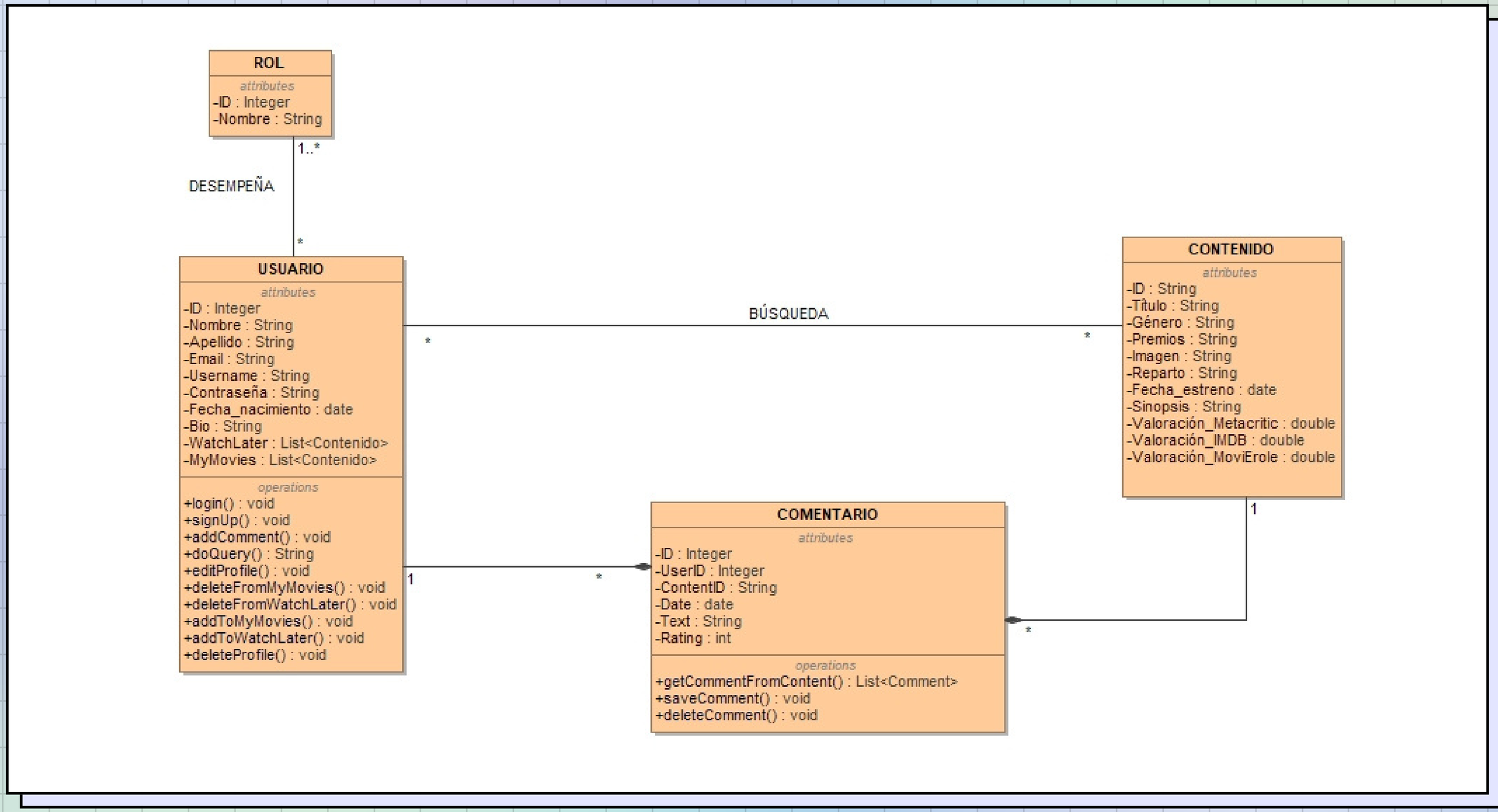
- Hemos decidido utilizar una arquitectura en la que mezclamos un sistema modelo-vista-controlador y cliente servidor.


moviErole [Erole main]
src/main/java
com.erole.moviErole
com.erole.moviErole.APIQuery
com.erole.moviErole.APIQuery.model.contentQuery
com.erole.moviErole.APIQuery.model.mostPopularQuery
com.erole.moviErole.APIQuery.model.titleQuery
com.erole.moviErole.configuration
com.erole.moviErole.controller
com.erole.moviErole.model
com.erole.moviErole.repository
com.erole.moviErole.service
dataStructures


: USUARIO
MainController
QueryController
1: Introduce petición de búsqueda
2: Traslada la petición del usuario
QueryController se conecta a la API y recoge el resultado en formato JSON que ésta le proporciona. Posteriormente parsea dicho resultado a Java y los añade a una lista de contenido (que podría ser vacía)
3: Devuelve lista de resultados
alt
[Lista vacía]
4: Mensaje de "error"
[else]
5: Lista de contenido

Modelos

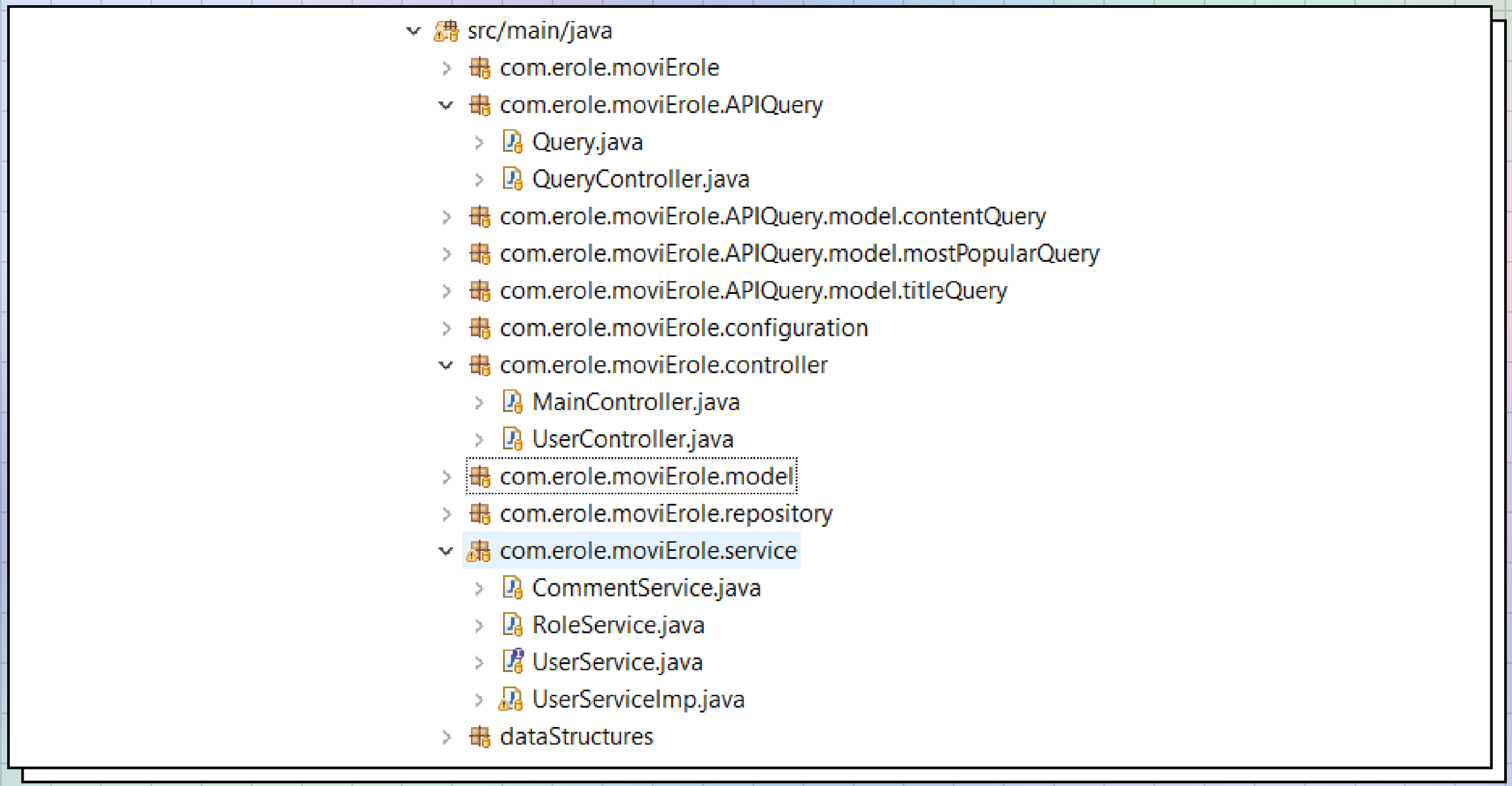

Resgistrar usuario
Iniciar sesión
«include»
«include»
«include»
USUARIO
Editar perfil
Visualizar perfiles
Buscar multimedia
«extend»
Comentar
«extend»
Valorar contenido
Buscar películas
Buscar serie


ROL
attributes
-ID : Integer
-Nombre : String
1..*
DESEMPEÑA
*
USUARIO
attributes
-ID : Integer
-Nombre : String
-Apellido : String
-Email : String
-Username : String
-Contraseña : String
-Fecha_nacimiento : date
-Bio : String
-WatchLater : List<Contenido>
-MyMovies : List<Contenido>
operations
+login() : void
+signUp() : void
+addComment() : void
+doQuery() : String
+editProfile() : void
+deleteFromMyMovies() : void
+deleteFromWatchLater() : void
+addToMyMovies() : void
+addToWatchLater() : void
+deleteProfile() : void
BÚSQUEDA
*
*
CONTENIDO
attributes
-ID : String
-Título : String
-Género : String
-Premios : String
-Imagen : String
-Reparto : String
-Fecha_estreno : date
-Sinopsis : String
-Valoración_Metacritic : double
-Valoración_IMDB : double
-Valoración_MoviErole : double
1
1
*
*
COMENTARIO
attributes
-ID : Integer
-UserID : Integer
-ContentID : String
-Date : date
-Text : String
-Rating : int
operations
+getCommentFromContent() : List<Comment>
+saveComment() : void
+deleteComment() : void

Principios

```java
/**
     * Recoge la peticion de movernos al login
     * @param model → este atributo conecta el html con el proyecto java
     * @return → nos mostrara en el navegador el archivo html alojado en esa direccion
     */
    @RequestMapping("/user/login")
    public String moveToLogin(Model model) {
        model.addAttribute("user", new User());
        return "/user/login";
    }

    /**
     * Recoge la peticion de movernos a la pagina ppal del proyecto.
     * @return → nos devuelve el html correspondiente a la pagina ppal
     */
    @RequestMapping("/app")
    public String mainPage(Model model) {
        model.addAttribute("query", new Query());
        model.addAttribute("loggedUser",
userServ.searchByUsername(MoviEroleApplication.getLoggedUser()));
        return "app/index";
    }
```

```java
    @RequestMapping("/app/user/edit")
    public String editProfile(Model model) {
        model.addAttribute("user", userServ.searchByUsername(MoviEroleApplication.getLoggedUser()));
        return "user/edit";
    }

    @RequestMapping("/user/edit")
    public String saveChanges(User user) {
        if(userServ.save(user) ≠ null) return "redirect:/logout?edit";
        else return "redirect:/app/myUser?error";
    }

    @RequestMapping("/app/user/delete")
    public String deleteProfile() {
        userServ.deleteUser(userServ.searchByUsername(MoviEroleApplication.getLoggedUser()));
        return "redirect:/logout?delete";
    } return go(f, seed, [])
}
```


src/main/java
com.erole.moviErole
com.erole.moviErole.APIQuery
Query.java
QueryController.java
com.erole.moviErole.APIQuery.model.contentQuery
com.erole.moviErole.APIQuery.model.mostPopularQuery
com.erole.moviErole.APIQuery.model.titleQuery
com.erole.moviErole.configuration
com.erole.moviErole.controller
MainController.java
UserController.java
com.erole.moviErole.model
com.erole.moviErole.repository
com.erole.moviErole.service
CommentService.java
RoleService.java
UserService.java
UserServiceImp.java
dataStructures

Pruebas

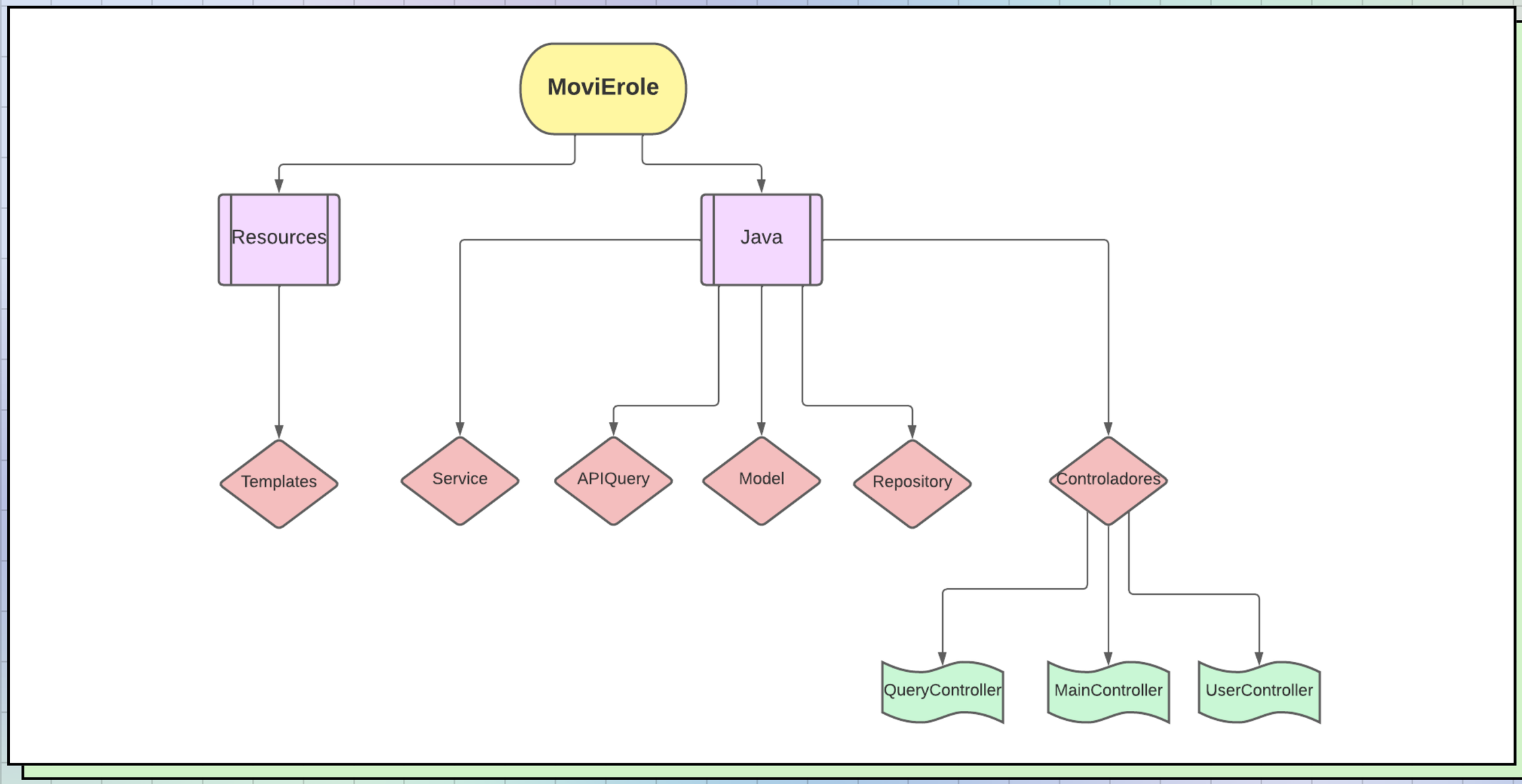
Actor Tests

```java
@Test //Un actor sin argumentos no contiene información
void inicialmenteEstaVacio() {
	assertEquals(actor.getName(), null);
}

@Test //Tras insertar datos, el usuario contiene información
void contieneInformacion() {
	actor = new Actor("id", "image", "name", "asCharacter");
	assertEquals(false, actor.getId().isEmpty());
	assertEquals(false, actor.getImage().isEmpty());
	assertEquals(false, actor.getName().isEmpty());
	assertEquals(false, actor.getName().isEmpty());
}
```

Comment Tests

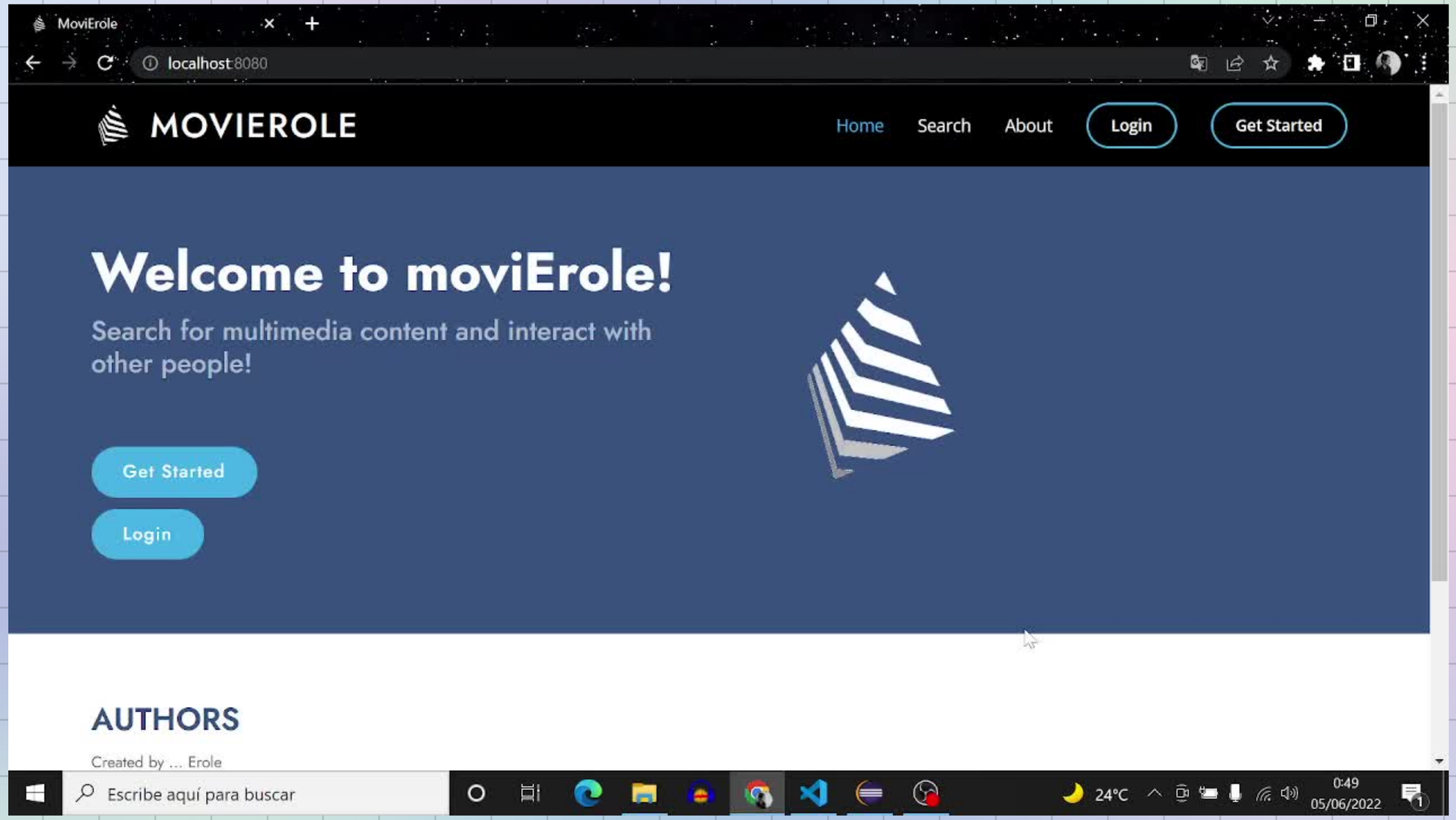
```
        @Test //Un actor sin argumentos no contiene información
        void inicialmenteEstaVacio() {
                assertEquals(comment.getId(), null);
        }

        @Test //Tras insertar datos, el usuario contiene información
        void contieneInformacion() {
                User user = new User();
                comment = new Comment(user, "1", "comment", 0); //Creamos un objeto
de tipo comentario
                assertEquals(false, comment.getUser() == null);
                assertEquals(false, comment.getContentId() == null);
                assertEquals(false, comment.getText() == null);
        }

}
```

Modelo de implementación


MoviErole
Resources
Java
Templates
Service
APIQuery
Model
Repository
Controladores
QueryController
MainController
UserController

Despliegue

Resultados


MoviErole
localhost:8080
MOVIEROLE
Home
Search
About
Login
Get Started
Welcome to moviErole!
Search for multimedia content and interact with other people!
Get Started
Login
AUTHORS
Created by ... Erole
Escribe aquí para buscar
24°C
0:49
05/06/2022

Conclusiones

- El trabajo en equipo y el uso de la metodología Scrum, que favorece la comunicación y la cooperación, nos ha permitido llevar a cabo un desarrollo organizado y a tiempo de nuestro producto.
- La distribución del trabajo ha sido clave, debido a esto, hemos mantenido coherencia en el proyecto, además de mejorar el desarrollo de las distintas tareas.

¡Gracias!