

# MOVIEROLE

Escuela técnica Superior de Ingeniería Informática  
Introducción a Ingeniería de Software

# Indice

○ ○ ○ ○

- |          |                  |           |                            |
|----------|------------------|-----------|----------------------------|
| <b>1</b> | Miembros y Roles | <b>8</b>  | Patrones/Principios        |
| <b>2</b> | Problema         | <b>9</b>  | Pruebas                    |
| <b>3</b> | Solución         | <b>10</b> | Estrategias y Herramientas |
| <b>4</b> | Requisitos       | <b>11</b> | Modelo de Implementación   |
| <b>5</b> | Planificación    | <b>12</b> | Resultados                 |
| <b>6</b> | Arquitectura     | <b>13</b> | Conclusiones               |
| <b>7</b> | Modelos          |           |                            |

# 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

- Nuestra aplicación consiste en una aplicación web enfocada al entretenimiento.
- En esta 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 para poder ofrecerlo de manera más eficiente al usuario.
- 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 media de semana y media.
- En cada tarea hemos hecho equipos de 2 a 3 personas favoreciendo así la productividad y realización gracias a los diferentes puntos de vista.























# Arquitectura

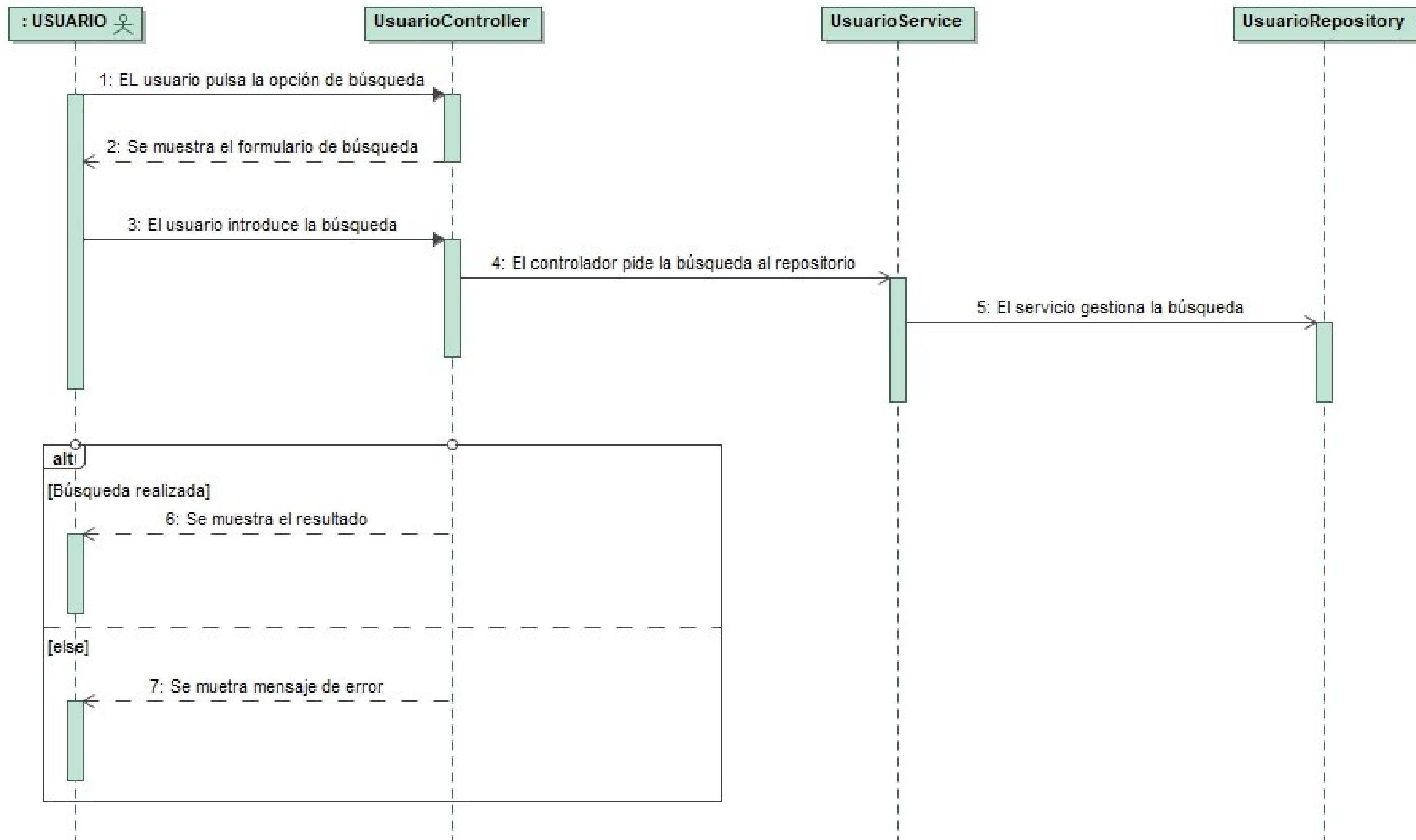
- Hemos decidido utilizar una arquitectura en la que mezclamos un sistema vista controlador y cliente servidor.



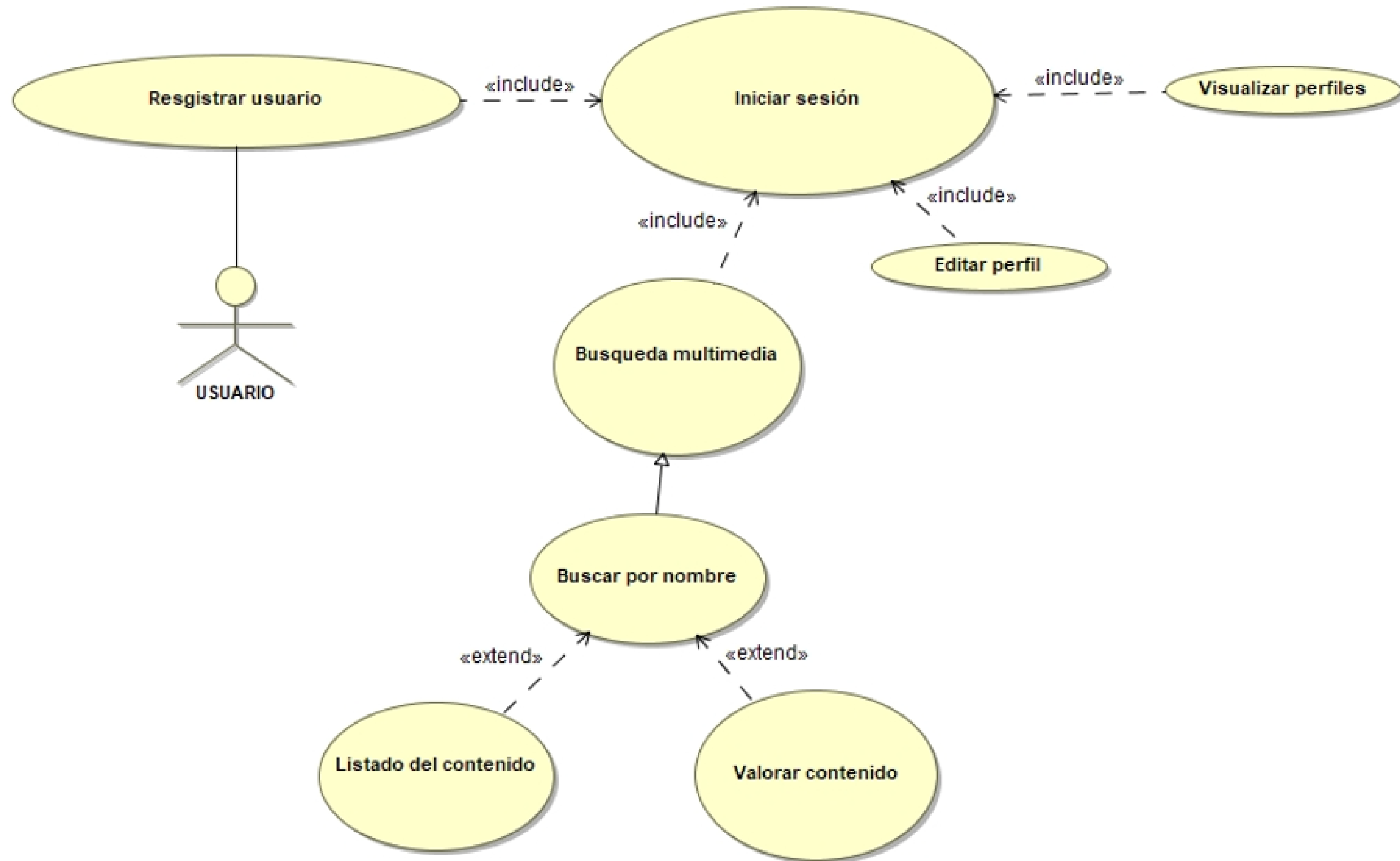
▼  > moviErole [Erole main]

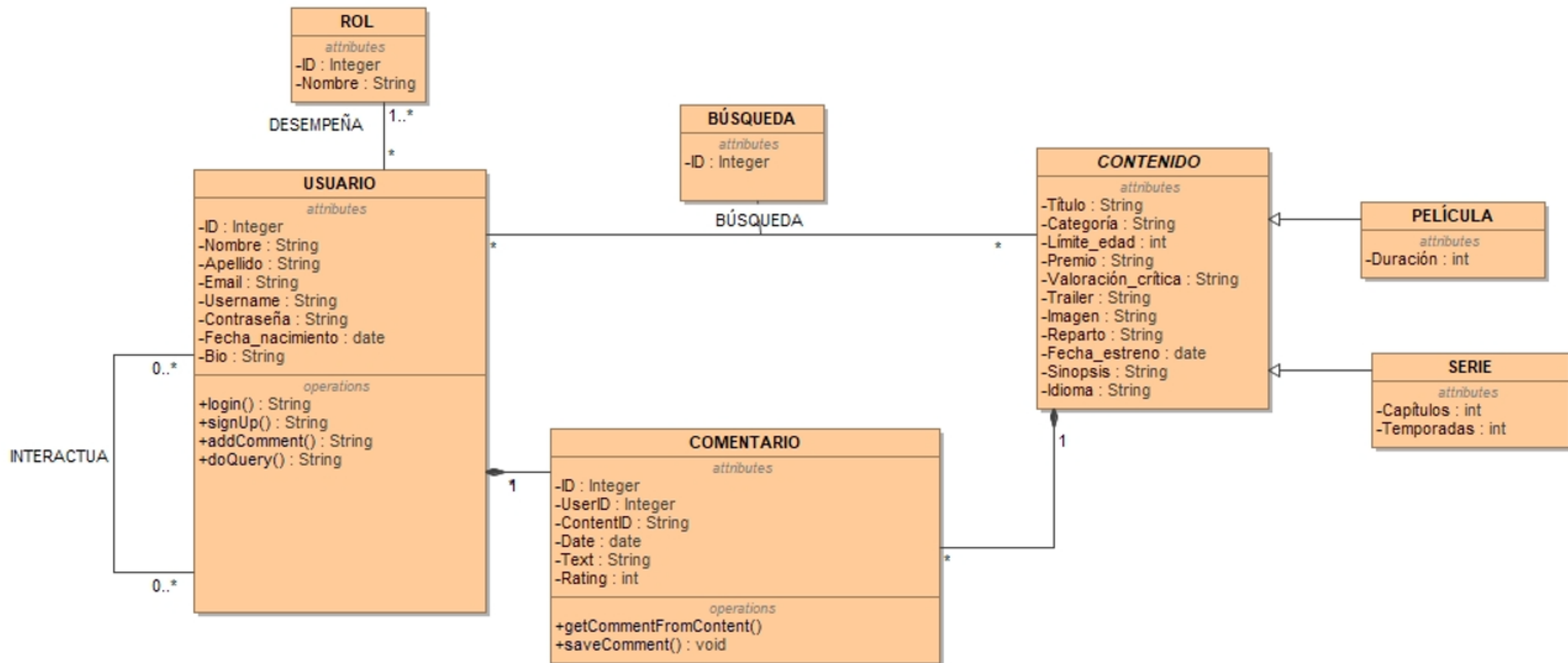
▼   src/main/java

- >   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



# Modelos





# **Patrones / Principios**

```
@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", userService.searchByUsername(MovieRoleApplication.getLoggedUser()));
    return "app/index";
}

@RequestMapping("/app/user/edit")
public String editProfile(Model model) {
    model.addAttribute("user", userService.searchByUsername(MovieRoleApplication.getLoggedUser()));
    return "user/edit";
}

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

# Pruebas



# Actor Tests

```
@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



# Despliegue



# Resultados



# Conclusiones

- Al haber tenido una metodología Scrum bien organizada y la comunicación del equipo hemos tenido un desarrollo correcto y a tiempo de nuestra aplicación web.
- La distribución del trabajo ha sido clave, ya que al formar grupos de mínimo dos personas manteníamos una coherencia y podíamos mejorar el contenido de cualquier tarea gracias a la comunicación y diferentes puntos de vista.

**¡Gracias!**