Documentación Sprint 2

Link del GitHub

https://github.com/rexxar433/GRUPO-52--Ciclo-3.git

Historias de usuario Sprint 2

Historia de Usuario			
ID:	U02	**Nombre:**	Crear Usuario
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como usuario creo las credenciales de acceso		
Funcionalidad:	Crear cuenta de usuario		
Criterio de Aceptación:	El usuario proporciona - Nombre - Apellido - Edad - Correo electrónico - Contraseña		

Historia de Usuario			
ID:	C01	**Nombre:**	Listar Películas
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin o como usuario		
Funcionalidad:	Listar películas		
Criterio de aceptación:	El usuario ingresa a la plataforma o el usuario ingresa a la página películas		

Historia de Usuario			
ID:	C02	**Nombre:**	Buscar Película por titulo
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin o como usuario		
Funcionalidad:	Listar películas por titulo		
Criterio de aceptación:	El usuario ingresa una palabra para que sea filtrada por el campo titulo		

Historia de Usuario			
ID:	C03	**Nombre:**	Buscar Película por genero
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin o como usuario		
Funcionalidad:	Listar películas por genero		
Criterio de aceptación:	El usuario ingresa un género para que sea filtrada por este		

Historia de Usuario			
ID:	C04	**Nombre:**	Buscar Película por argumento
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin o como usuario		
Funcionalidad:	Listar películas por argumento		
Criterio de aceptación:	El usuario ingresa una palabra para que sea filtrada por el campo argumento		

Historia de Usuario			
ID:	C05	**Nombre:**	Buscar Película por clasificación
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin o como usuario		
Funcionalidad:	Listar películas por clasificación		
Criterio de aceptación:	El usuario ingresa la clasificación para que sea filtrada por este campo		

Historia de Usuario			
ID:	C06	**Nombre:**	Buscar Película
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin o como usuario		
Funcionalidad:	Buscar película		
Criterio de aceptación:	El usuario busca una película por su respectivo ID		

Historia de Usuario			
ID:	A01	**Nombre:**	Agregar Película

Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin agrego películas a la pagina		
Funcionalidad:	Agregar películas		
Criterio de aceptación:	El admin proporciona - titulo - fecha lanzamiento - duración - sinopsis - compañía - clasificación - genero		

Historia de Usuario			
ID:	A02	**Nombre:**	Eliminar Película
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin		
Funcionalidad:	Eliminar Película		
Criterio de aceptación:	El administrador puede eliminar una película usando su correspondiente ID		

Historia de Usuario			
ID:	A04	**Nombre:**	Listar Películas Admin
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin		
Funcionalidad:	Listar películas		
Criterio de aceptación:	El administrador se le despliega una lista de películas donde las pueda eliminar o editar		

Historia de Usuario			
ID:	A05	**Nombre:**	Eliminar Usuario
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin		
Funcionalidad:	Eliminar Usuario		
Criterio de aceptación:	El administrador puede eliminar a un usuario usando su correspondiente ID		

Historia de Usuario			
ID:	A06	**Nombre:**	Listar Usuarios

Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin		
Funcionalidad:	Listar Usuarios		
Criterio de aceptación:	El administrador se le despliega una lista de películas donde puede eliminar usuarios		

Historia de Usuario			
ID:	A07	**Nombre:**	Listar Géneros
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin		
Funcionalidad:	Listar Géneros		
Criterio de aceptación:	El administrador se le despliega una lista de Géneros donde los puede eliminar o editar		

Historia de Usuario			
ID:	A11	**Nombre:**	Listar Categorías
Prioridad de negocio:	Alta	**Iteración Asignada:**	
Rol:	Yo como admin		
Funcionalidad:	Listar Categorías		
Criterio de aceptación:	El administrador se le despliega una lista de categorías donde las puede editar		

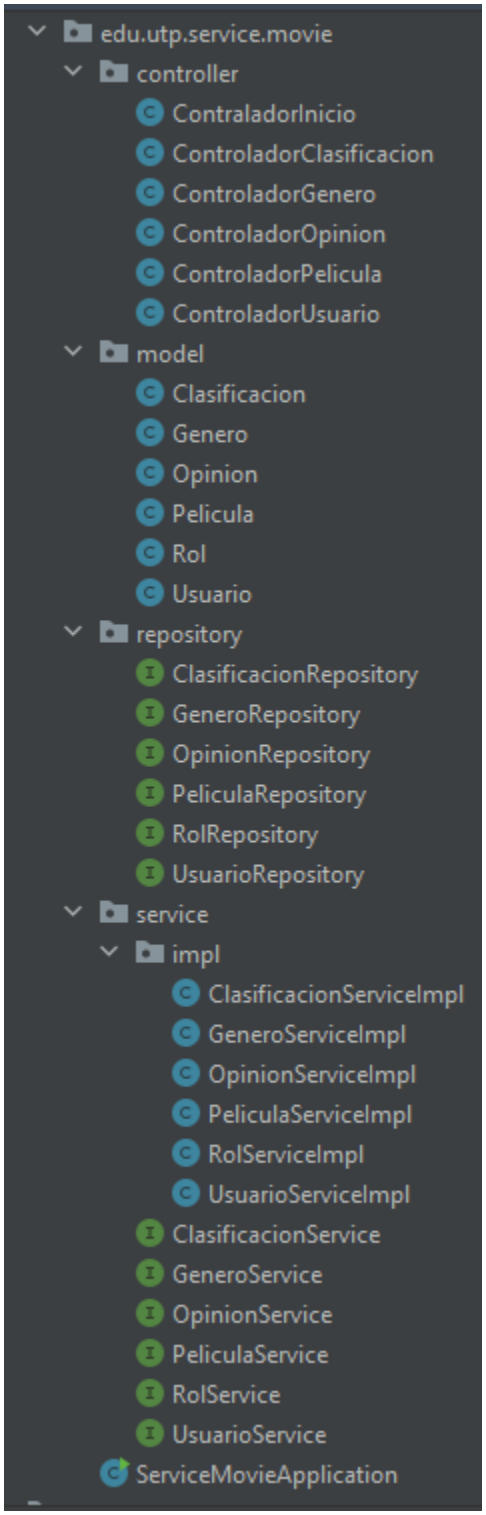
Pantallazos de código

Maven POM (Librerias que se usaron)

- JPA
- Validation
- thymeleaf
- starter-web
- boostrap
- Font-awesome
- webjars-location
- devtools
- mysql-connector
- Lombok

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>bootstrap</artifactId>
        <version>5.2.0</version>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>font-awesome</artifactId>
        <version>6.1.2</version>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>webjars-locator</artifactId>
        <version>0.45</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
```

Estructura del Proyecto



Controlador Pelicula

```java
private PeliculaService peliculaService;

@Autowired
private ClasificacionService clasificacionService;

@Autowired
private GeneroService generoService;

dymmdaniel
@GetMapping("/")
public String listarPeliculas(Model model){
    var peliculas :List<Pelicula> =peliculaService.listarPeliculas();
    model.addAttribute( attributeName: "peliculas",peliculas);
    return "admin/Peliculas";
}

dymmdaniel +1
@GetMapping("/{id}")
public String buscarPelicula(Model model,@PathVariable Long id){
    var pelicula :Pelicula =peliculaService.buscar(id);
    if(pelicula==null){
        log.info("No existe la pelicula");
        model.addAttribute( attributeName: "titulo", attributeValue: "No existe");
    }else {
        model.addAttribute( attributeName: "titulo",pelicula.getTitulo());
        model.addAttribute( attributeName: "pelicula", pelicula);
    }
    return "movie/pelicula";
}

dymmdaniel
@GetMapping("/titulo/{titulo}")
public String buscarPeliculaTitulo(@PathVariable String titulo,Model model){
    var peliculas :List<Pelicula> =peliculaService.findByTitulo(titulo);
    if(peliculas.isEmpty()){
        // Colocar error
    }else {
      model.addAttribute( attributeName: "peliculas", peliculas);
    }
```

Controlador Usuario

```java
//@GetMapping("/Login")
dymmdaniel +1
public String login(Model model){

    return "auth/login";
}

dymmdaniel +1
@GetMapping("/registrar")
public String register(Usuario usuario){

    return "auth/createUser";
}

dymmdaniel +1
@PostMapping("/crear")
public String crearUsuario(@Valid Usuario usuario, Errors errores){
    if(errores.hasErrors()){
        return "auth/createUser";
    }else{
        Rol rol=rolService.buscar( id: 2);
        usuario.setRol(rol);
        usuarioService.guardar(usuario);
        return "redirect:/";
    }
}
dymmdaniel +1
@GetMapping("/agregar")
public String agregar(Usuario usuario) { return "auth/createUser"; }

dymmdaniel
@GetMapping("/eliminar")
public String eliminar(Usuario usuario){
    usuarioService.eliminar(usuario);
    return "redirect:/api/usuario/";
}
```

Modelo de Ejemplo, Película con su relación Muchos a muchos con genero

```java
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@NotEmpty(message="Por favor agregue un titulo")
@Size(max=45, message="No se puede colocar un nombre tan largo.")
private String titulo;

@NotEmpty(message="Por favor agregue un argumento o sinopsis de la pelicula.")
private String argumento;

@Min(value=1, message="La duracion no tiene un valor permitido.")
private int duracion;

@Min(value=1, message="el año no tiene un valor permitido.")
private int ano;

private double puntuacion;

//private image imagen;

1 usage
@ManyToOne(cascade = {CascadeType.PERSIST,CascadeType.MERGE,CascadeType.DETACH,CascadeType.REFRESH})
@JoinColumn(name="clasificacion_id")
private Clasificacion clasificacion;

@ManyToMany
@JoinTable(name = "pelicula_genero",
        joinColumns = @JoinColumn(name = "pelicula_id"),
        inverseJoinColumns = @JoinColumn(name = "genero_id"))
private List<Genero> generos;

3 usages
@OneToMany(mappedBy = "pelicula",cascade = {CascadeType.PERSIST,CascadeType.MERGE,CascadeType.DETACH,CascadeType.REFRESH})
private List<Opinion> opiniones;
```

Ejemplo de Repository, PeliculaRepository

```java
import java.util.Optional;
3 usages  dymmdaniel +1
@Repository
public interface PeliculaRepository extends JpaRepository<Pelicula,Long> {

    1 usage  dymmdaniel
    @Query(value = "select * from pelicula a where a.titulo like %:titulo% ", nativeQuery = true)
    public List<Pelicula> findByTitulo(String titulo);

    1 usage  dymmdaniel
    @Query(value = "select * from pelicula a where a.argumento like %:argumento% ", nativeQuery = true)
    public List<Pelicula> findByArgumento(String argumento);

    1 usage  dymmdaniel
    @Query(value = "select pelicula.*,clasificacion.nombre as clasificacion\n" +
            "from pelicula\n" +
            "inner join clasificacion\n" +
            "on pelicula.clasificacion_id = clasificacion.id \n" +
            "where clasificacion.nombre=:clasificacion", nativeQuery = true)
    public List<Pelicula> findByClasificacion(String clasificacion);
    1 usage  dymmdaniel
    @Query(value = "select pelicula.*\n" +
            "from pelicula \n" +
            "inner join pelicula_genero \n" +
            "on pelicula_genero.pelicula_id = pelicula.id \n" +
            "inner join genero\n" +
            "on pelicula_genero.genero_id = genero.id \n" +
            "where genero.nombre =:genero",nativeQuery = true)
    public List<Pelicula> findByGenero(String genero);


}
```

Ejemplo de Service, Pelicula service

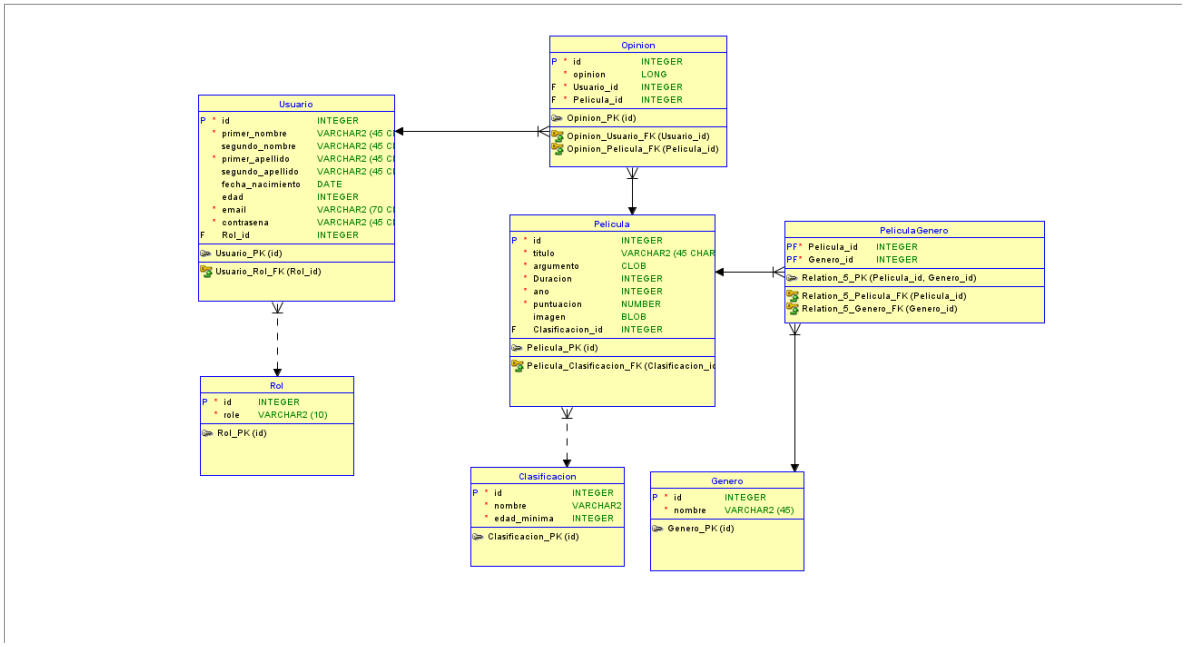
```
8 usages  1 implementation  dymmdaniel +1
public interface PeliculaService {
    2 usages  1 implementation  dymmdaniel
    public List<Pelicula> listarPeliculas();

    1 implementation  dymmdaniel
    public Pelicula guardar(Pelicula pelicula);

    1 implementation  dymmdaniel
    public void eliminar(Pelicula pelicula);

    1 implementation  dymmdaniel
    public Pelicula buscar(Long id);

    1 usage  1 implementation  dymmdaniel
    public List<Pelicula> findByTitulo(String titulo);

    1 usage  1 implementation  dymmdaniel
    public List<Pelicula> findByArgumento(String argumento);

    1 usage  1 implementation  dymmdaniel
    public List<Pelicula> findByClasificacion(String clasificacion);

    1 usage  1 implementation  dymmdaniel
    public List<Pelicula> findByGenero(String genero);
}
```

Ejemplo implementación service, PeliculaServiceImpl

```
dymmdaniel +1
@Service
public class PeliculaServiceImpl implements PeliculaService {

    8 usages
    @Autowired
    private PeliculaRepository peliculaRepository;
    2 usages  dymmdaniel
    @Override
    @Transactional(readOnly = true)
    public List<Pelicula> listarPeliculas() {
        return (List<Pelicula>) peliculaRepository.findAll();
    }

    dymmdaniel +1
    @Override
    @Transactional
    public Pelicula guardar(Pelicula pelicula) { return peliculaRepository.save(pelicula); }

    dymmdaniel
    @Override
    @Transactional
    public void eliminar(Pelicula pelicula) { peliculaRepository.delete(pelicula); }

    dymmdaniel
    @Override
    @Transactional(readOnly = true)
    public Pelicula buscar(Long id) {
        return peliculaRepository.findById(id).orElse( other: null);
    }

    1 usage  dymmdaniel
    @Override
    @Transactional(readOnly = true)
    public List<Pelicula> findByTitulo(String titulo) {
        return (List<Pelicula>) peliculaRepository.findByTitulo(titulo);
    }

    1 usage  dymmdaniel
    @Override
    @Transactional(readOnly = true)
    public List<Pelicula> findByArgumento(String argumento) {
```

Base de datos

Se hace uso de SQL Developer para poder observar la estructura de la base de datos donde tenemos lo siguiente



Create tables

```
16      -- predefined type, no DDL - XMLTYPE
17
18  •   CREATE TABLE clasificacion (
19          id          INTEGER NOT NULL auto_increment,
20          nombre      VARCHAR(45) NOT NULL,
21          edad_minima INTEGER NOT NULL,
22          primary key(id)
23      );
24
25  •   CREATE TABLE genero (
26          id     INTEGER NOT NULL auto_increment,
27          nombre VARCHAR(45) NOT NULL,
28          primary key(id)
29      );
30
31  •   CREATE TABLE opinion (
32          id          INTEGER NOT NULL auto_increment,
33          opinion     TEXT NOT NULL,
34          usuario_id  INTEGER NOT NULL,
35          pelicula_id INTEGER NOT NULL,
36          primary key(id)
37      );
38
39  •   CREATE TABLE pelicula (
40          id              INTEGER NOT NULL auto_increment,
41          titulo          VARCHAR(45) NOT NULL,
42          argumento       TEXT NOT NULL,
43          duracion        INTEGER NOT NULL,
44          ano             INTEGER NOT NULL,
45          puntuacion      DOUBLE NOT NULL,
```

Creación de las relaciones

```
ALTER TABLE pelicula_genero ADD CONSTRAINT pelicula_genero_pk PRIMARY KEY ( pelicula_id,
                                                                            genero_id );
ALTER TABLE opinion
    ADD CONSTRAINT opinion_pelicula_fk FOREIGN KEY ( pelicula_id )
        REFERENCES pelicula ( id );

ALTER TABLE opinion
    ADD CONSTRAINT opinion_usuario_fk FOREIGN KEY ( usuario_id )
        REFERENCES usuario ( id );

ALTER TABLE pelicula
    ADD CONSTRAINT pelicula_clasificacion_fk FOREIGN KEY ( clasificacion_id )
        REFERENCES clasificacion ( id );

ALTER TABLE pelicula_genero
    ADD CONSTRAINT pelicula_genero_genero_fk FOREIGN KEY ( genero_id )
        REFERENCES genero ( id );

ALTER TABLE pelicula_genero
    ADD CONSTRAINT pelicula_genero_pelicula_fk FOREIGN KEY ( pelicula_id )
        REFERENCES pelicula ( id );

ALTER TABLE usuario
    ADD CONSTRAINT usuario_rol_fk FOREIGN KEY ( rol_id )
        REFERENCES rol ( id );
```

Informe De Retrospectiva

Para este segundo sprint tanto las creaciones de las páginas en HTML junto con sus estilos y contenidos, está casi completado de igual manera por la parte del back y la conexión con la base de datos como se tenía previsto, tal vez se vio un poco complicado la implementación de thymeleaf para conectar el front con lo realizado en el back pero estamos trabajando para mejorar en eso, para el próximo sprint nos propondremos a mejorar nuestra sincronización a la hora de debatir nuestras dudas e inquietudes respecto al proyecto, ahora teniendo en cuenta que un compañero se encuentra en el extranjero.