

UTNMDP

Regional Mar del Plata

TRABAJO FINAL: PROGRAMACION II

MATERIA: PROGRAMACION II.

PROFESORES: Leonardo Gabriel Chiessa y Marcos Ezequiel Ramos.

ALUMNOS: Julian Giampietro y Agustina Delgado.

AÑO: 2024

UTN Universidad Tecnológica Nacional, Facultad Regional
Mar del Plata.

STORM



Introduccion:

Este proyecto es una aplicación en Java para gestionar usuarios y sus juegos favoritos. Incluye clases para usuarios, administradores y juegos, con funcionalidades como agregar juegos favoritos, banear usuarios, poder comentarlo y darle una puntuacion. (Dependiendo de su rango).

Estructura:

MenuLogin.java: gestiona el inicio de sesión de usuarios, incluye un menú de inicio, métodos para registro y login de usuarios, y la eliminación de datos de usuario. Utiliza listas de usuarios y juegos, guía al usuario en el proceso de inicio de sesión y registro.

Menu.java: gestiona listas de usuarios y juegos, proporcionando menús para administrar perfiles, juegos y opciones específicas para administradores y desarrolladores. Incluye métodos para

cargar listas, mostrar perfiles, modificar datos, gestionar juegos y eliminar cuentas.

Game.java: La clase Games representa un juego en el sistema. Implementa la interfaz ShowInfo y gestiona atributos como ID de juego, nombre, género, ID de desarrollador, editor, comentarios, precio y calificación general. La clase incluye métodos para obtener y establecer estos atributos, así como para mostrar información del juego.

Comments.java: Representa un comentario en el sistema. Gestiona atributos como ID de comentario, texto del comentario, ID del usuario y calificación de usuarios. La clase incluye métodos para obtener y establecer estos atributos, y un constructor que inicializa los mismos, asegurando que su puntaje esté entre 0 y 5.

Person.java: Clase abstracta para atributos comunes de usuarios y administradores.

Developer.java: La clase Developers en el archivo extiende la clase Users y añade atributos específicos como company y followers. Además de heredar los atributos y métodos de Users, esta clase incluye métodos para obtener y establecer la compañía, y un método para obtener el número de seguidores, el cual se inicializa aleatoriamente.

Users.java: La clase Users es parte del modelo y representa a un usuario en el sistema. Hereda de Person e implementa ShowInfo. Gestiona el ID de usuario, juegos favoritos y

comentarios. Proporciona métodos para autenticación, agregar juegos favoritos, mostrar información del usuario y sus comentarios.

Administrator.java: Extiende Users y añade atributos como salario y nivel de moderación.

Genre.java: diferentes géneros de juegos, como **ACTION, ADVENTURE, ARCADE, RPG, STRATEGY, SHOOTER** y **SIMULATION**. Cada uno de estos valores representa un tipo específico de juego y se utiliza para categorizar los juegos en el sistema.

Funcionalidad:

Al ingresar al programa, tendremos la opción de iniciar sesión o registrarnos. Únicamente podemos registrarnos como usuarios comunes, ya que los administradores o desarrolladores serán registrados a través del código.

```
System.out.println("-----Bienvenido a Storm-----");
System.out.println("1) Login");
System.out.println("2) Registrarse");
System.out.println("3) Salir");
option = scanner.nextInt();
```

Ejemplos de usuario administrador o developers:

```
users.add(new Users("user1", "password1", "user1@example.com"));
users.add(new Users("user2", "password2", "user2@example.com"));
users.add(new Users("user3", "password3", "user3@example.com"));
users.add(new Developers("developer", "devpass", "dev@example.com", "developer", 0));
users.add(new Administrator("admin", "adminpass", "admin@example.com"));
```

Una vez iniciada la sesión, se identificará el tipo de usuario. De esta forma, podremos acceder al menú correspondiente: administrador, usuario o desarrollador.

```
public void usertype (List<Users> listaUsuarios, List<Games> listaJuegos, Users usuarioLogeado) {

    if (usuarioLogeado instanceof Administrator) {
        adminMenu(listaUsuarios, listaJuegos, usuarioLogeado);

    } else if (usuarioLogeado instanceof Developers) {
        devMenu(listaJuegos, usuarioLogeado, usuarioLogeado);

    } else {
        mainMenu (listaUsuarios, listaJuegos, usuarioLogeado);
    }
}
```

Una vez que el programa identifica el tipo de usuario, se mostrarán los distintos menús correspondientes:

Administrador:

```
System.out.println("-----Admin Menu-----");
System.out.println("1) Ver todos los usuarios");
System.out.println("2) Ver todos los juegos");
System.out.println("3) Agregar juego");
System.out.println("4) Eliminar juego");
System.out.println("5) Eliminar usuario");
System.out.println("6) Salir");
```

Developer:

```
System.out.println("-----Developer Menu-----");
System.out.println("1) Ver todos los juegos");
System.out.println("2) Agregar juego");
System.out.println("3) Modificar juego");
System.out.println("4) Salir");
option = scanner.nextInt();
```

Usuarios:

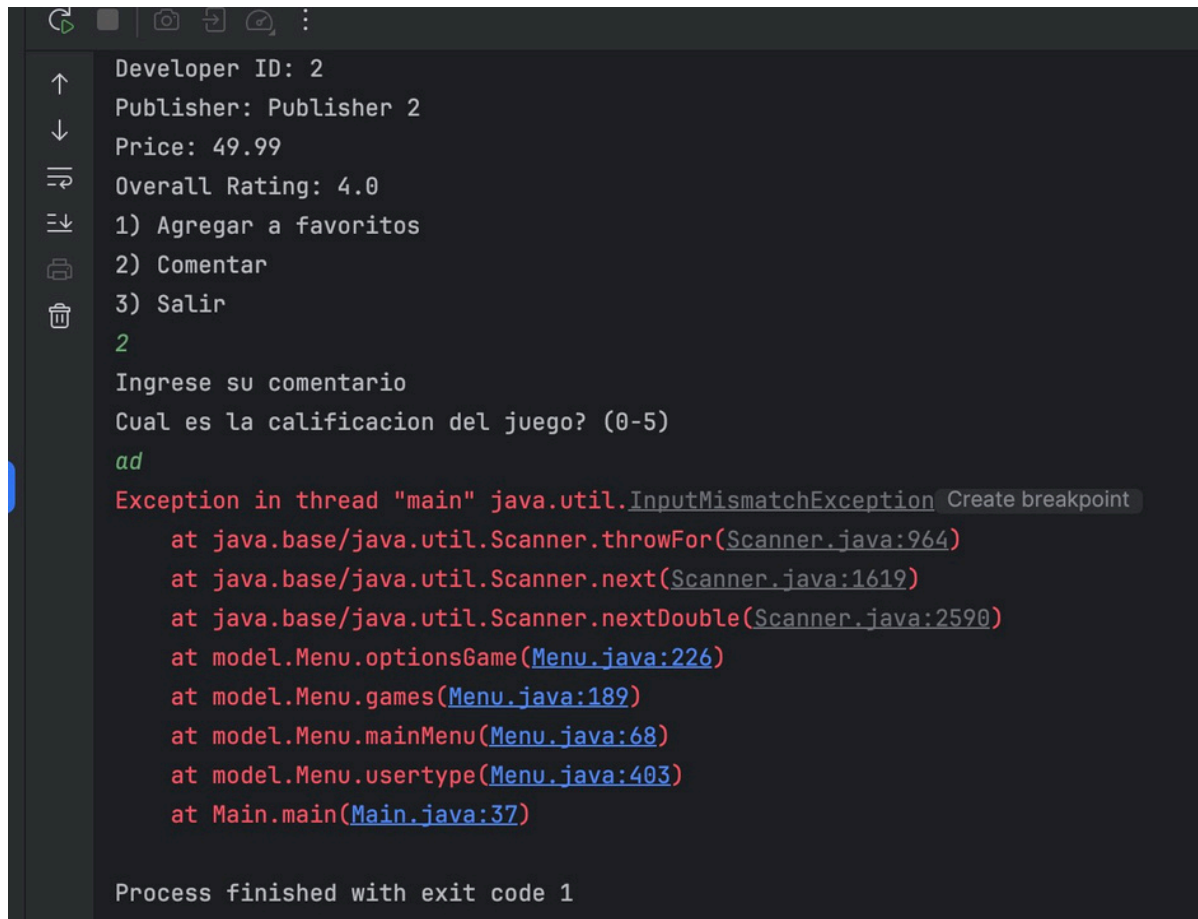
```
System.out.println("-----STORM-----");
System.out.println("1) Lista de Juegos");
System.out.println("2) Juegos Favoritos");
System.out.println("3) Buscar juego");
System.out.println("4) Salir");
option = scanner.nextInt();
```

Problemas y soluciones:

En el desarrollo de la construcción del código, nos surgían diferentes problemas, desde excepciones no trabajadas hasta funcionamientos incorrectos.

Se identificó un problema en el uso de Scanner donde los saltos de línea no se manejaban correctamente, lo que provocaba que solo se guardara la primera palabra de los

comentarios ingresados. Para resolver este inconveniente, se implementó un uso más adecuado del método `scanner.nextLine()`, asegurando así que se capture la línea completa de texto ingresado por el usuario.



```
Developer ID: 2
Publisher: Publisher 2
Price: 49.99
Overall Rating: 4.0
1) Agregar a favoritos
2) Comentar
3) Salir
2
Ingrese su comentario
Cual es la calificacion del juego? (0-5)
ad
Exception in thread "main" java.util.InputMismatchException Create breakpoint
    at java.base/java.util.Scanner.throwFor(Scanner.java:964)
    at java.base/java.util.Scanner.next(Scanner.java:1619)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2590)
    at model.Menu.optionsGame(Menu.java:226)
    at model.Menu.games(Menu.java:189)
    at model.Menu.mainMenu(Menu.java:68)
    at model.Menu.usertype(Menu.java:403)
    at Main.main(Main.java:37)

Process finished with exit code 1
```

Al eliminar una cuenta propia, el código finalizaba y lanzaba una excepción. Para solucionar este problema, se optimizó la lógica de finalización del ciclo, permitiendo que el flujo del programa regrese al menú de inicio de sesión de manera controlada.

```
Saliendo...
-----Inicio-----
1) Mi Perfil
2) Juegos
3) Salir
1
-----Perfil-----
1) Ver perfil
2) Modificar datos
3) Eliminar cuenta
4) Ver comentarios
5) Salir
3
Eliminar cuenta
Exception in thread "main" java.util.ConcurrentModificationException Create breakpoint
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1597)
    at model.MenuLogin.lambda$deleteAllUserData$2(MenuLogin.java:94)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1597)
    at model.MenuLogin.lambda$deleteAllUserData$3(MenuLogin.java:93)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1597)
    at model.MenuLogin.deleteAllUserData(MenuLogin.java:92)
    at Main.main(Main.java:40)

Process finished with exit code 1
```

Se detectó un problema donde los usuarios comunes, incluso después de ser baneados por un administrador, podían seguir utilizando las funciones del sistema. Para solucionarlo, se implementó un manejo adecuado de excepciones mediante un bloque try-catch en el nivel más alto del programa (main), asegurando que los usuarios baneados no puedan acceder al menú de funciones
