SQUIRREL GAMES

Realizado por: Néstor, Alejandro, David, Antoine, Felipe.

ÍNDICE

INTRODUCCIÓN

En nuestro proyecto, el sistema gestiona los aspectos de los juegos organizados por Squirrel Games, estos incluyen:

Gestión de participantes: Registro y seguimiento de jugadores, deudas y estatus (normal o infiltrado).

Administración de pruebas: Cada juego tiene pruebas que los participantes deben superar para seguir alzarse con la victoria.

Control de roles: Pink Guards tienen distintos roles (Mánager, Soldier y Worker) con jerarquías en la organización del evento.

El sistema organiza, administra y controla las ediciones de los juegos, gestionando los participantes, pruebas y roles del equipo de trabajo, garantizando una experiencia fluida y controlada.

CARACTERÍSTICAS PRINCIPALES

Administrar participantes

Los jugadores son registrados con datos personales; nombre, nacionalidad, estado (normal o infiltrado) y deudas. También gestionan los infiltrados, con restricciones especiales a la hora de su eliminación.

Gestión de usuarios

Gestiona cada prueba realizada del juego con información clave como el id y nombre de los usuarios participantes, facilitando así el control de datos y pudiendo realizar una precisa tabla con los usuarios eliminados y vencedores de cada prueba.

Jerarquía del personal (PinkGuards)

El personal puede ser: Mánager, que supervisan los demás rangos; Soldiers, responsables de la seguridad y tareas especiales; y Workers, encargados de labores operativas y mantenimiento. Cada rango tiene acceso y permisos específicos.

IMPLEMENTACIÓN DEL SISTEMA

Excepciones personalizadas

JugadorDuplicadoException: Evita registros duplicados.

SimulacionNoPermitidaException: Restringe pruebas inválidas.

EliminacionInvalidaException: Controla expulsiones incorrectas.

Tecnologías usadas

En nuestro proyecto hemos utilizado la versión 23 de *Java*, con los siguientes entornos de desarrollo: *Visual Studio Code*, *Eclipse*, *Intellij Idea Community Version*, y algunas herramientas para subir el proyecto a *Github* como *Github Desktop*.

Se ha utilizado *JUnit 5* para realizar pruebas unitarias y *JaCoCo* para la cobertura de código. Se emplea *Git* para un control de las versiones y *Javadoc* para la documentación.

PROCESO DE INSTALACIÓN

Pasos de Instalación:

- Para instalar el proyecto deberemos dirigirnos al repositorio remoto de *Github* y clonarlo a nuestro *IDE*.
- Se debe leer el README para poder informarnos sobre las funcionalidades del proyecto y cómo utilizarlo.
- Por último, tendremos que ejecutar el proyecto en nuestro entorno de desarrollo desde la clase *Main.java*.

ESTRUCTURA DEL PROYECTO

```
SquirrelGames
├── JRE System Library [jre]
├── src
│   ├── squirrelGames
│   │   ├── Main.java
│   │   ├── exceptions
│   │   │   ├── InfiltradoNoEliminableException.java
│   │   │   ├── JugadorDuplicadoException.java
│   │   │   ├── PorcentajeInvalidoException.java
│   │   │   ├── SimulacionNoPermitidaException.java
│   │   │   ├── SupervisorInvalidoException.java
│   │   ├── integrantesJuego
│   │   │   ├── EstadoParticipante.java
│   │   │   ├── Participantes.java
│   │   │   ├── PinkGuard.java
│   │   │   ├── pinkGuardsRanks
│   │   │   │   ├── ArmasManager.java
│   │   │   │   ├── ArmasSoldier.java
│   │   │   │   ├── Manager.java
│   │   │   │   ├── Soldier.java
│   │   │   │   ├── Worker.java
│   │   ├── juegos
│   │   │   ├── Juegos.java
│   │   ├── pruebas
│   │   │   ├── Pruebas.java
├── Cobertura de código (Porcentaje de cobertura)
│   ├── jacoco-resources
│   ├── SquirrelGames
│   ├── index.html
│   ├── jacoco-sessions.html
├── docs (Documentación y Javadoc)
│   ├── squirrelGames
├── test (Pruebas Unitarias)
│   ├── squirrelGames
│   │   ├── integrantesJuego
│   │   ├── juegos
│   │   ├── pruebas
├── pom.xml
├── README.md
```

Se ha realizado una estructura organizada que se divide en los siguientes campos;

El **código (src**), los **test (test)**, la **documentación (docs)**, la **cobertura del código**, el archivo **.gitignore** y el archivo **README**.

https://github.com/AntoineGiz77/SquirrelGames

TESTING Y COBERTURA DE CÓDIGO

- **Cobertura de Código**

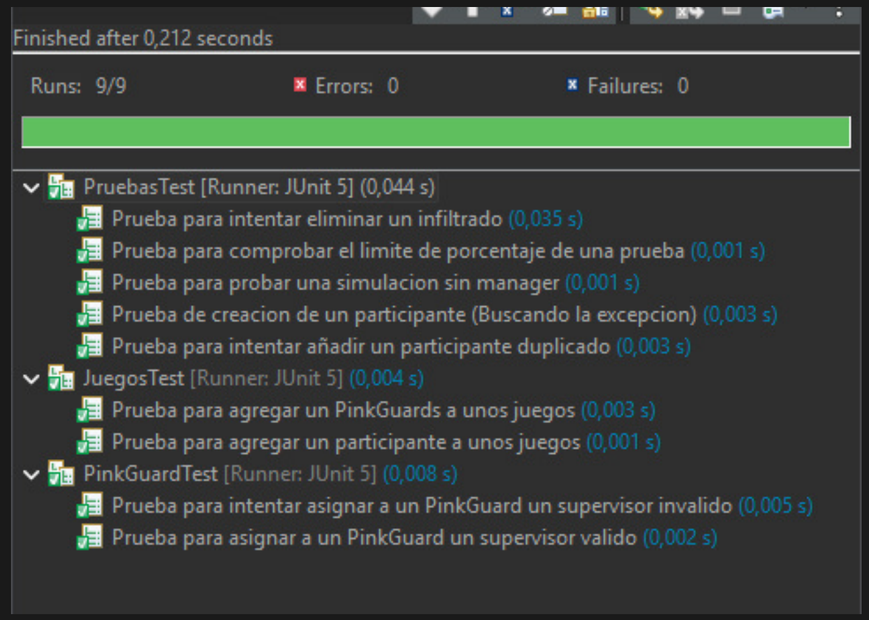
  Se utilizó *JaCoCo* para analizar la cobertura de pruebas, asegurando que las funcionalidades críticas estén correctamente verificadas.

- **Pruebas Unitarias**

  Las pruebas unitarias fueron implementadas con JUnit, validando el correcto funcionamiento de clases clave como Juegos, Participantes y Pruebas.

- **Capturas de Pruebas**

  Se incluyen evidencias visuales de los test ejecutados para demostrar su efectividad y cobertura.



test (1) (6 mar 2025 12:34:12) > SquirrelGames

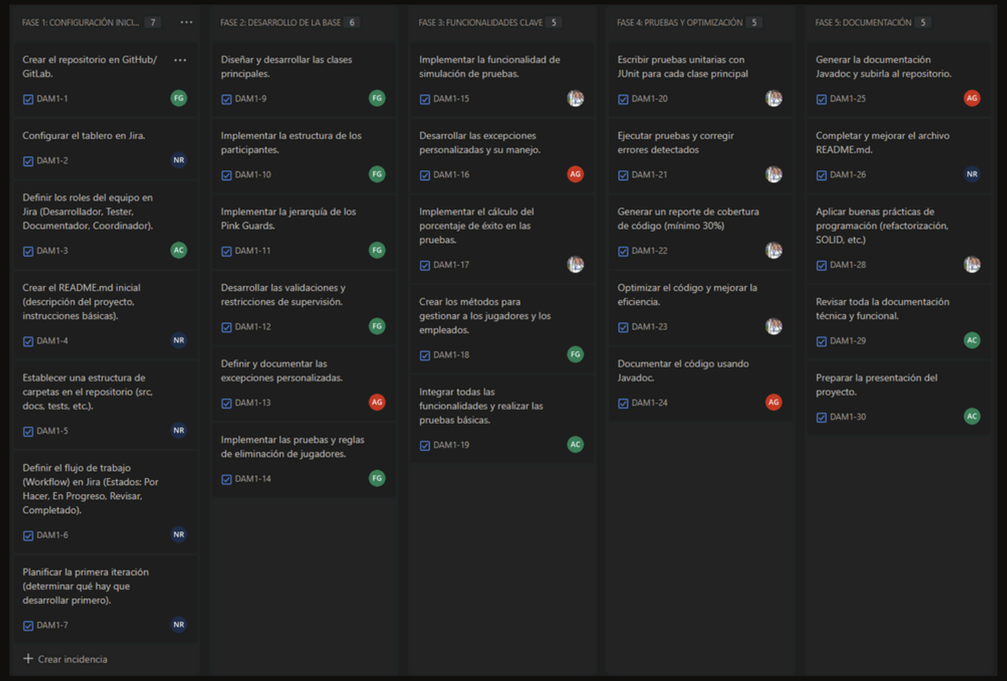
SquirrelGames

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
src		31 %		50 %	50	83	178	278	38	65	4	16
test		91 %		n/a	5	19	0	45	5	19	0	3
Total	807 of 1.424	43 %	17 of 34	50 %	55	102	178	323	43	84	4	19

RETOS Y SOLUCIONES

Organizar la función de cada integrante del grupo

Se solucionó gracias al uso de una herramienta de gestión de proyectos (Jira), en la cual designábamos las tareas a realizar y cada integrante escogía las tareas que podía hacer.



Ayudarnos entre los compañeros a completar las tareas pendientes

Lo solucionamos comunicándonos a través de un servicio de mensajería instantánea y un chat de voz para coordinarnos mejor en diversas tareas.

La fase de pruebas resultó ser un componente crítico del proyecto presentando desafíos significativos.

Las principales dificultades encontradas se centraron en el establecimiento de conexiones estables entre las dependencias y la resolución de errores de código.

MUCHAS GRACIAS