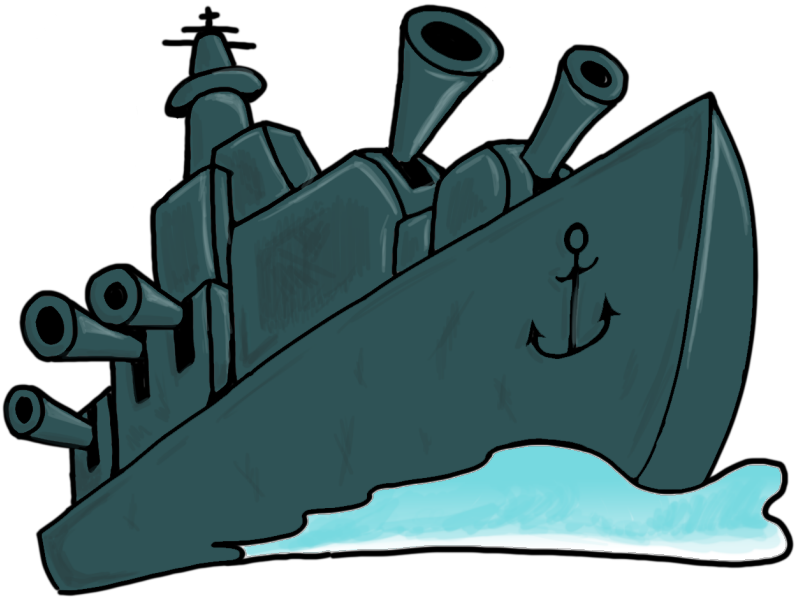


Juego de Batalla Naval

Arquitectura del Sistema

Grupo: La Osa que Baila

Integrantes: Martinez, Facundo Jesus

Mugni, Juan Mauricio

Reynoso Choque, Kevin Walter

2022

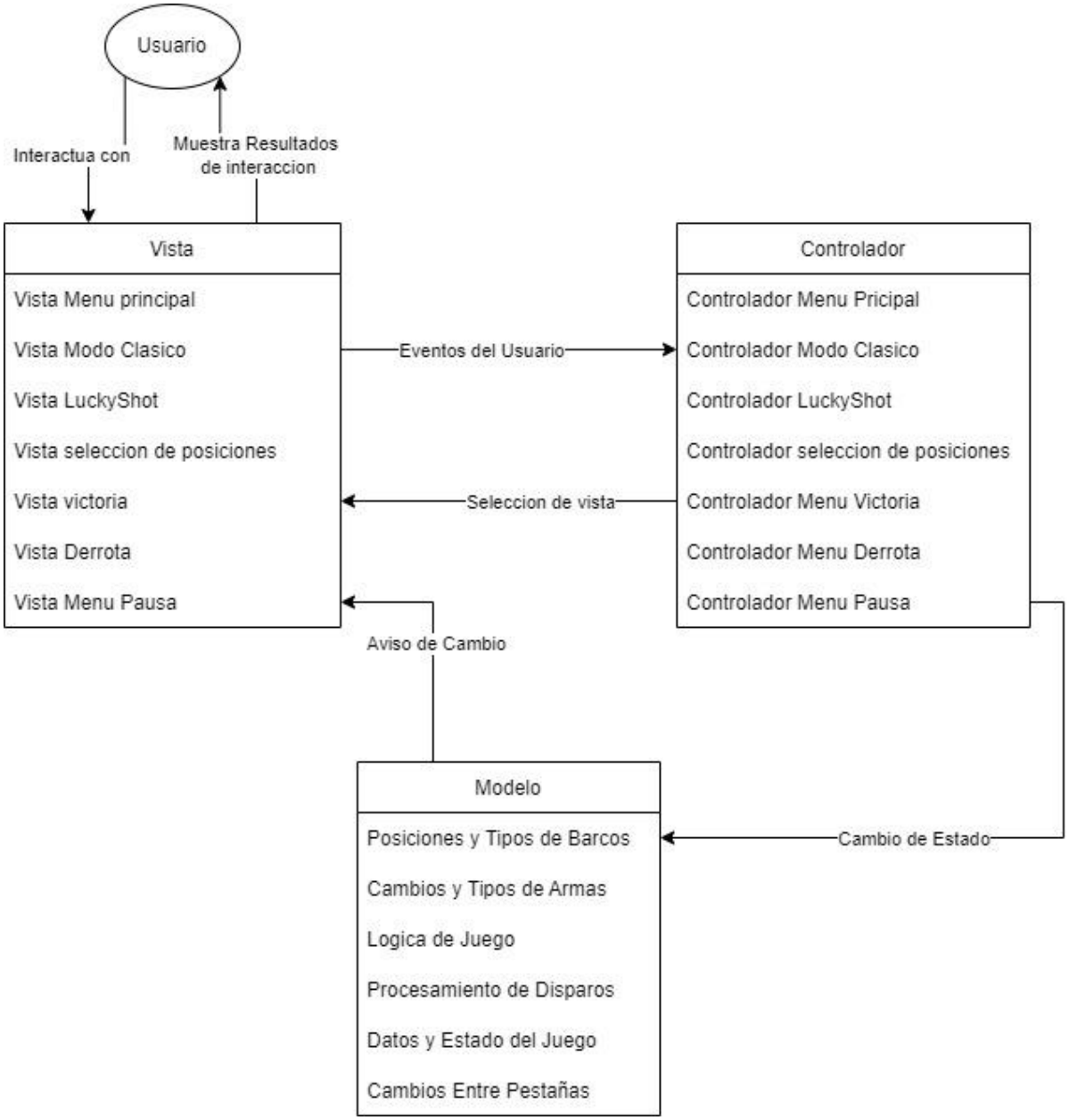
Historial de Cambios

Fecha	Resumen	Autor/es
12/6/2022	Primera presentación del informe realizado	Martinez, Facundo Jesus Mugni, Juan Mauricio Reynoso Choque, Kevin Walter
19/6/2022	Corrección de los puntos comentados	Martinez, Facundo Jesus Mugni, Juan Mauricio Reynoso Choque, Kevin Walter

Índice

Diagrama de Arquitectura General

El proyecto sigue un patrón de diseño MVC (Modelo Vista Controlador), en este se separan los datos de la aplicación, interfaz de usuario y lógica de control en distintos componentes como se puede ver en el siguiente diagrama:



Patrón de Arquitectura

El modelo MVC elegido sigue la siguiente estructura:

- El usuario se comunica con el controlador mediante solicitudes, este luego se comunica con el modelo y las vistas. Esto se realiza al presionar los botones para cada operación que quiera realizar el usuario.
- Los modelos responden a las solicitudes actualizando datos o retornando los mismos, mientras que las vistas retornan una salida determinada dependiendo del resultado de la operación.
- La información a la que puede acceder el usuario es gestionada y representada por La vista
- El modelo lleva registro de los datos del juego, con los cuales, dependiendo de las peticiones del controlador, enviará la información necesaria al controlador o la vista.

Diagrama de Despliegue

En el siguiente Diagrama podemos notar 3 nodos: Mouse, Computadora y Monitor. La comunicación entre estos se da por medio de USB (para mouse y computadora) y HDMI/VGA/Displayport (para computadora y monitor).

Dentro de la PC podemos ver el artefacto BatallaNaval.jar el cual sería el ejecutable del juego.

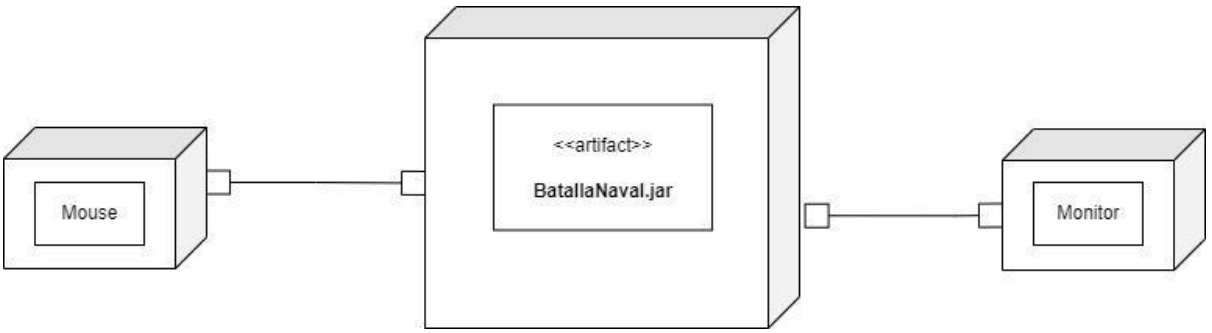
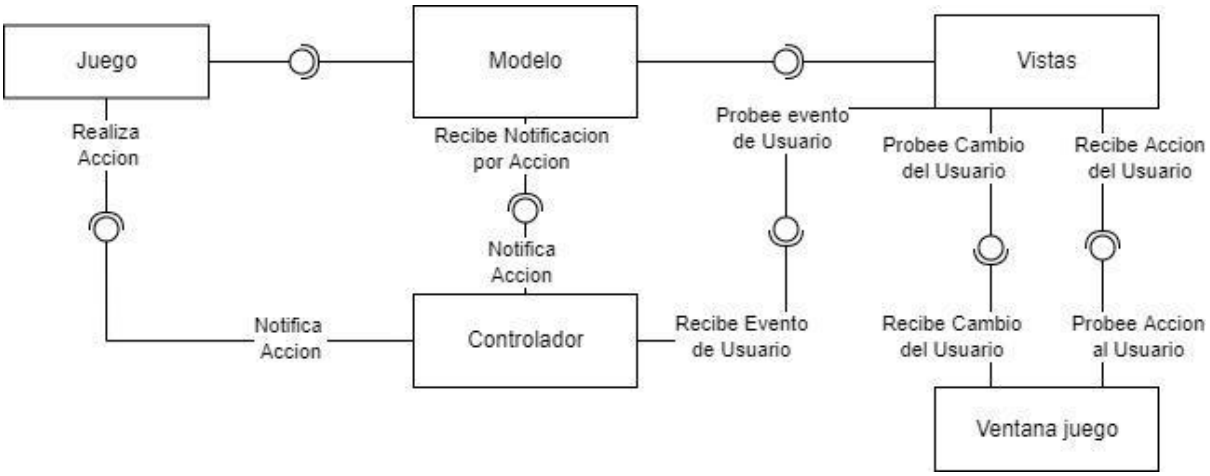


Diagrama de Componentes



Pruebas de Integración

Se realizan estas pruebas para asegurar que los diferentes componentes interactúan correctamente entre sí. Se utiliza para ello el diagrama de componentes anteriormente mostrado.

1. Test de datos: se verifica el funcionamiento de los objetos de clase Barco al ser instanciados durante la selección de posiciones al momento de almacenar sus posiciones correspondientes.

2. Test de creación de la interfaz de usuario: se verifica que se creen todas las vistas necesarias para el desarrollo del juego. Se integran las vistas, controladores y el modelo.

3. Test interacción con ventanas: se verifica el cambio entre ventanas al realizar la acción correspondiente para abrir una nueva ventana. El test incluye las vistas, Controlador y modelo.

4. Test de modo de juego Clásico: se verifica el proceso completo para completar una partida del modo de juego Clásico iniciando desde la selección por el menú principal, completar la ubicación de los barcos e iniciar la partida, hasta cumplir la condición para finalizar la partida. Se incluyen todos los componentes realizados del programa.

5. Test de modo de juego LuckShot: se verifica el proceso completo para completar una partida del modo de juego LuckShot de la misma forma que el modo Clásico pero agregando el funcionamiento de los distintos disparos implementados. Se incluyen todos los componentes realizados del programa.

Cada test está implementado dentro de la Integración Continua al momento de realizar un push o un pull request al main. Para ello, se agregó el correspondiente archivo maven.yml a

fin de ejecutar las correspondientes acciones implementadas mediante GitHub Actions y, también, el archivo pom.xml para trabajar los test con Maven.

Cabe aclarar que los test se ejecutan tanto desde la IDE propia como en el repositorio remoto con Maven. En el caso de ejecutarlos de manera local, se deben descargar e instalar las dependencias correspondientes las cuales la propia IDE se encargaría de ello (a partir del archivo pom.xml). Por el otro caso, se deben limitar los test debido a problemas con generar las vistas desde GitHub Actions.

A continuación, se presenta el archivo maven.yml del workflow.

```yaml
name: Java CI with Maven

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Paso 1 - Checkout
      uses: actions/checkout@v3

    - name: Paso 2 - Set up JDK 11
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'adopt'

    - name: Build with Maven
      run: mvn --batch-mode --update-snapshots test
```