Intel y Gentes

Documento de Arquitectura

Autores: Agüero, David - Bastida, Lucas - Miranda, Noelia - Palmiotti, Mauro

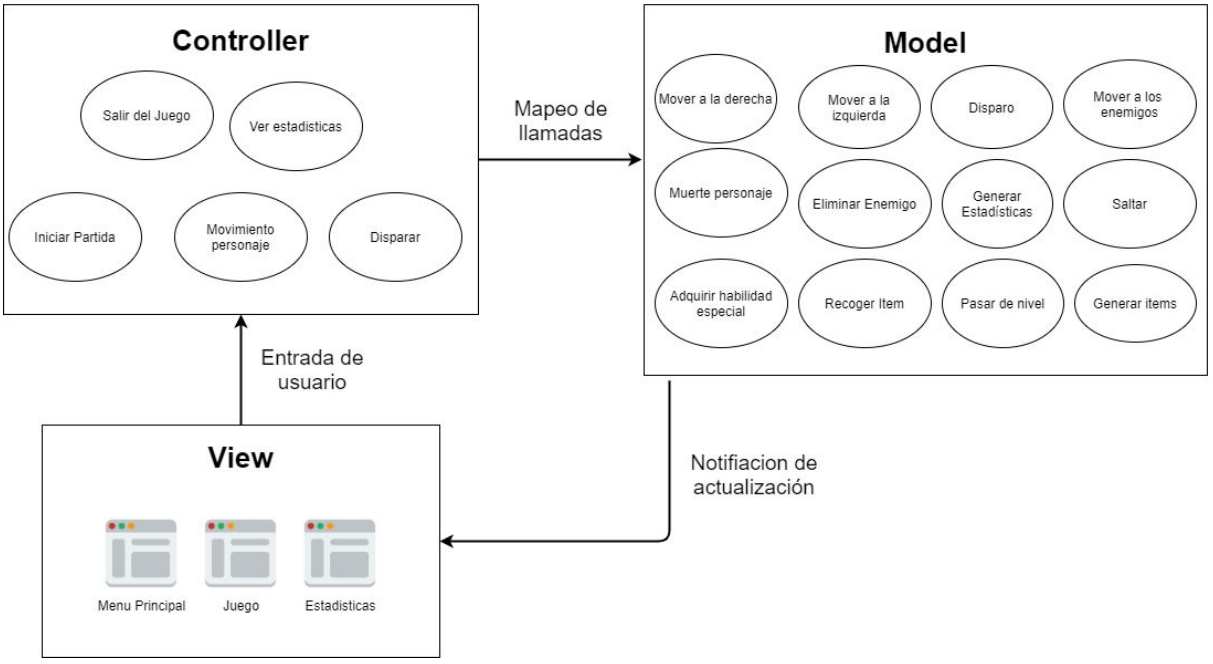
Fecha: 19/06/2020

Versión del Documento: 1.0.1

Historial de cambios

Versión	Fecha	Resumen de Cambios	Autor
0.1.0	14/06/2020	documento inicial	Intel y Gentes
0.2.0	18/06/2020	Diag. Arquitectura, paquetes, componentes	Bastida, Lucas
1.0.0	19/06/2020	Creacion pruebas de integración	Bastida, Lucas
1.0.1	23/06/2020	Traspaso prueba al documento “Pruebas”	Bastida, Lucas

1. Diagrama de arquitectura



1.1 Explicación del patrón de arquitectura utilizado

1.1.1 Breve introduccion del patron Model-View-Controller

Se utilizó el patrón de Arquitectura MVC, el cual es un patrón de arquitectura de software que separa los datos y la lógica de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

1.1.2 El patrón MVC en nuestro proyecto

Tenemos 3 **vistas** diferentes:

- una para mostrar el menú inicial que se despliega al abrir el juego
- otra para mostrar el juego una vez iniciada la partida
- y una para visualizar las estadísticas

Los **controladores** se encargan de tomar los inputs del usuario en estas vistas e interpretar qué significan para el modelo.

La clase juego es la que representa el **modelo** en nuestro programa y es la que contiene la lógica del funcionamiento del juego.

1.1.2 El porqué del MVC - Requerimientos no funcionales

Esta arquitectura MVC nos permite que los datos sufran modificaciones independientemente de su representación y viceversa, es decir, se pueden agregar vistas y no tocar el modelo o tocar el modelo y no tocar las vistas, sin romper el sistema

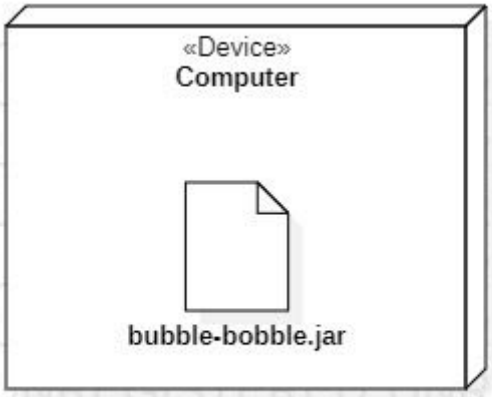
Esto se traduce en una clara ventaja: la mantenibilidad. En el largo plazo, si tenemos que extender o ampliar el sistema, tenemos una ventaja muchísimo más grande que en el caso de que estuviese todo en una misma aplicación.

También se optó por este modelo porque ninguno de los integrantes del grupo tiene experiencia desarrollando videojuegos o creando interfaces gráficas. El MVC también es usado cuando los requerimientos para interacciones y la presentación de datos son desconocidas. (Sommerville Figure 6.4)

Como desventaja, es más complejo hacer un MVC cuando las interacciones son simples. El costo que pagamos por usar este patrón es la complejidad.

2. Diagrama de despliegue

El software del juego solo requiere ser instalado en una computadora con Java SE 11:



3. Diagramas de componentes