Projectes de Programació

2a Entrega

Víctor Juez
Enric Bover
Mathias Bertorelli
2017/18-Q2

Índice

Funcionalidades

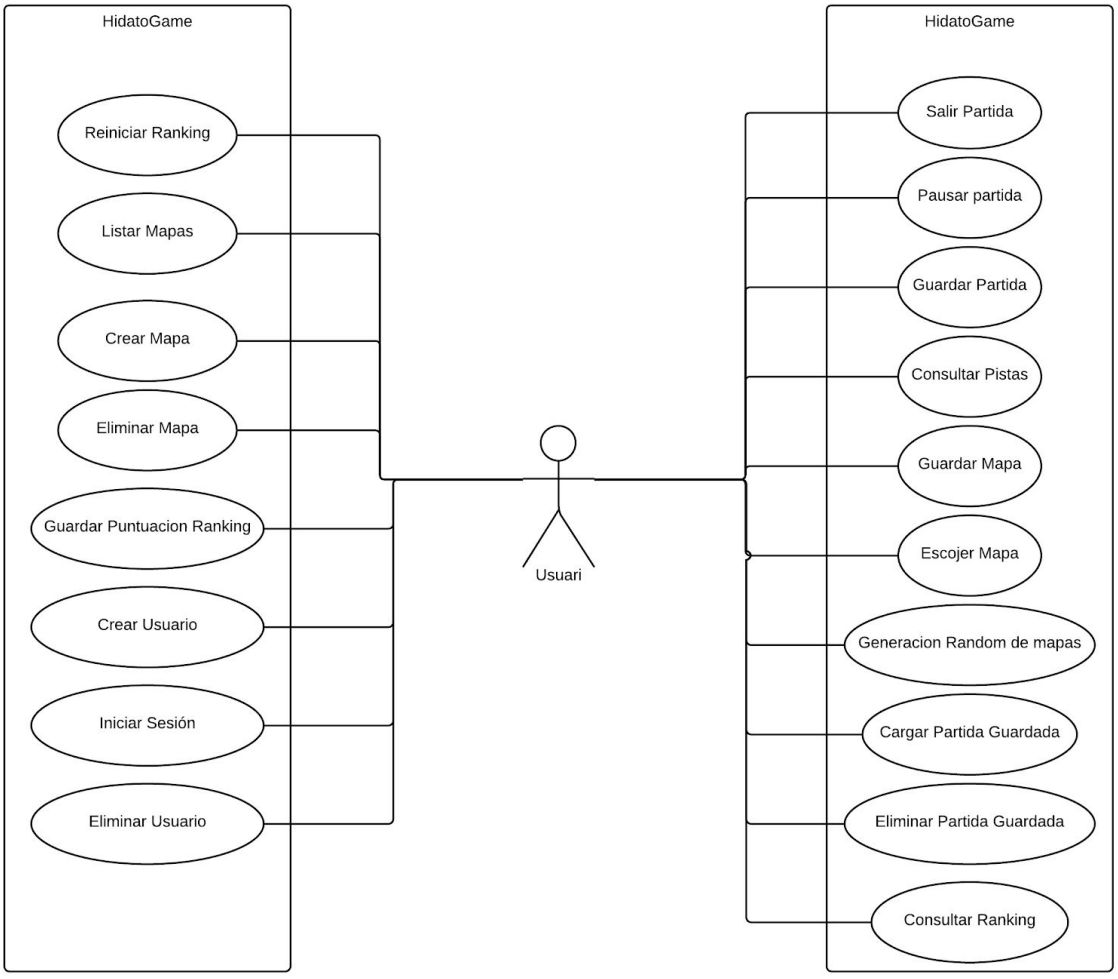
FUNCIONALIDADES OBLIGATORIAS:

- Añadir diferentes niveles de dificultad
- Sistema de ránking + records
- Permitir guardar la partida
- Controlar tiempo de partida (máquina y usuario)
- Definir bases de datos de Hidatos (mapas)
- Proponer Hidatos (+ validación máquina)
- Generación de mapas

FUNCIONALIDADES OPCIONALES:

- Escoger topología mapas
- Editor de mapas para usuarios.
- Categorizar por dificultad y topología mapa.

Diagrama casos de uso



CASOS DE USO:

- **JUGAR PARTIDA**
  - **Salir partida**: cuando un usuario quiere salir de la partida actual, accede al menú de pausa, y selecciona salir de la partida.
  - **Pausar partida**: cuando un usuario quiere pausar la partida actual, accede al menú de pausa, se para el contador de tiempo y se bloquea el tablero.
  - **Guardar partida**: cuando un usuario quiere guardar la partida actual, accede al menú de pausa, selecciona guardar partida y esta se almacena en el sistema y se sale de la partida.
  - **Consultar pista:** cuando un usuario desea consultar una pista para la partida, accede a la opción de pista y el sistema proporciona una pista.

Una Pista solo se proporciona si el usuario a colocado por lo menos la mitad de los números en el hidato.

- **INICIAR PARTIDA**
  - **Escoger mapa (bdd)**: Cuando un usuario desea escoger el mapa a jugar, accede a la lista de mapas, selecciona el que quiere jugar y se inicia la partida.
  - **Generación random de mapas**: Cuando un usuario desea jugar un mapa nuevo aleatorio el sistema se lo proporciona.
  - **Cargar partida guardada:** Cuando un usuario quiere continuar una partida ya empezada, accede al listado de sus partidas guardadas, selecciona la que desea y la partida se continúa.
  - **Eliminar partida guardada:** Cuando un usuario quiere eliminar una de sus partidas guardadas, accede al listado de sus partidas, selecciona la que desea eliminar y esta se elimina del sistema.

- **RANKING**
  - **Consultar ranking:** Cuando un usuario quiere consultar el ranking, accede al apartado de ranquing del menú principal y este se muestra.

- **GESTOR MAPAS**
  - **Listar mapas (bdd):** Cuando un usuario quiere consultar los mapas disponibles en el juego, accede a lista de mapas y estos se muestran en un listado.
  - **Crear mapa (editor):** Cuando un usuario desea crear un mapa nuevo, accede al menú de gestor de mapas, selecciona crear mapa y se abre un editor con las opciones para poder generarlo, en la primera pantalla del editor se le tienen que añadir las características básicas de un hidato: número de filas, columnas, tipo de casilla, adyacencia y un nombre. En la siguiente pantalla se abre el editor en si para poder rellenar las casillas del hidato en question. Si queremos guardar un hidato el sistema tiene un control de errores y te avisa si: 1) No están todas las casillas rellenadas, 2) Debe contener el primer y último número del hidato, 3) te avisa si no existe solución para el hidato generado.

- **FINALIZAR PARTIDA**
    - **Acabar partida:** Cuando el usuario finaliza la partida guarda la puntuación obtenida en el ranking si el hidato está bien resuelto, en caso contrario la partida continúa.

- **MENÚ DE USUARIOS**
    - **Crear usuario:** Cuando una persona quiere crear un usuario, introduce su username/nick y una contraseña y se creará un nuevo usuario con estos parámetros.
    - **Iniciar sesión:** Cuando un usuario desea iniciar sesión, selecciona su usuario en el menú principal e introduce su contraseña, si esta es válida, accede al juego.

Definición de conceptos.

Perfil de usuario: Una persona debe crear uno o más usuarios para jugar al juego. El usuario se escogerá un nombre o nick, que servirá para ponerle nombre a las puntuaciones del ranking o para guardar/cargar partidas creadas por el usuario.

Partida: Una partida se entenderá como la resolución de un mapa escogido previamente por el usuario o generado automáticamente al querer iniciar una partida. Una partida empezará con el mapa vacío, y terminará cuando el usuario complete el mapa de forma válida. En cualquier momento de la partida podremos acceder al menú de pausa, desde donde se podrá guardar el mapa, guardar la partida y salir al menú (guardando o no previamente la partida). También se podrá consultar el reloj de la partida (que estará visible desde la misma pantalla desde la que se juega la partida) y consultar las pistas disponibles.

Al acabar la partida, podremos enviar la puntuación al ranking, si lo deseamos.

Mapa: Hablaremos de mapas para referirnos a los hidatos a resolver en las partidas. Un mapa será un hidato, con su respondientes parámetros.

- Parámetros del mapa: Los parámetros del mapa definirán cada mapa de forma única. Los parámetros que definirán un mapa son: tipo de casilla (Quadrados, Triangulos, Hexagonos), adyacencias (costados o costados + ángulos), número de filas, número de columnas y dificultad. También guardaremos el nombre del usuario que ha creado el mapa.

- Dificultad de mapa: La dificultad se calcula en base a la cantidad de casillas que contienen un número o un interrogante y la cantidad de casillas adyacentes a estas que también tienen un número o interrogante

Reloj: Contador de tiempo (en segundos con precisión de dos decimales) que al empezar a jugar una partida nueva se inicializa a 0. Cuando se pausa la partida el reloj también se pausa. Cuando se guarda la partida se guarda el valor del reloj en su estado actual.

Pistas: Las pistas son ayudas a las cuales se podrán acceder desde la partida. Las pistas te indicarán si los números que has colocado hasta el momento en el tablero son correctos o no.

Puntuación de la partida: La puntuación de la partida en cuestión se calculará en base a los parámetros del mapa que se ha resuelto y en base al tiempo y a la cantidad de veces que se han consultado las pistas disponibles.

- La fórmula para definir la puntuación de una partida es:
  - Puntuación = 100000 * factorDificultad - factorPistas - factorTiempo * tiempoResolución

    - factorDificultad = 0.25 -> facil
      = 0.75 -> media
      = 1 -> dificil
    - factorTiempo = 100 -> facil
      = 75 -> media
      = 25 -> dificil
    - factorPistas = numeroPistas^2

Ranking: El ranking es un sistema de puntos en el cual podemos ver la puntuación total de cada usuario (la suma de la puntuación obtenida en todas las partidas que ha jugado).

Diagrama de Classes

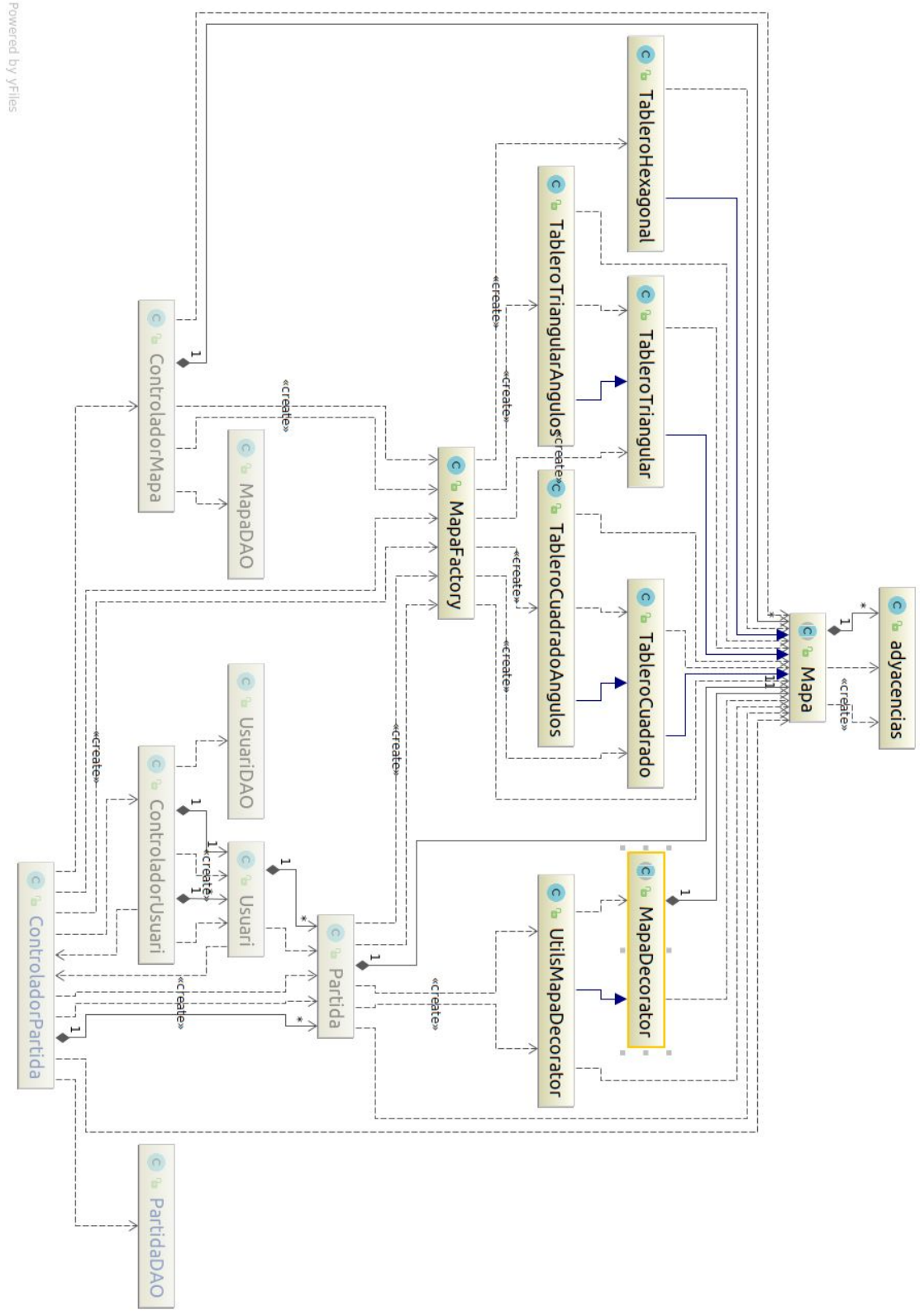
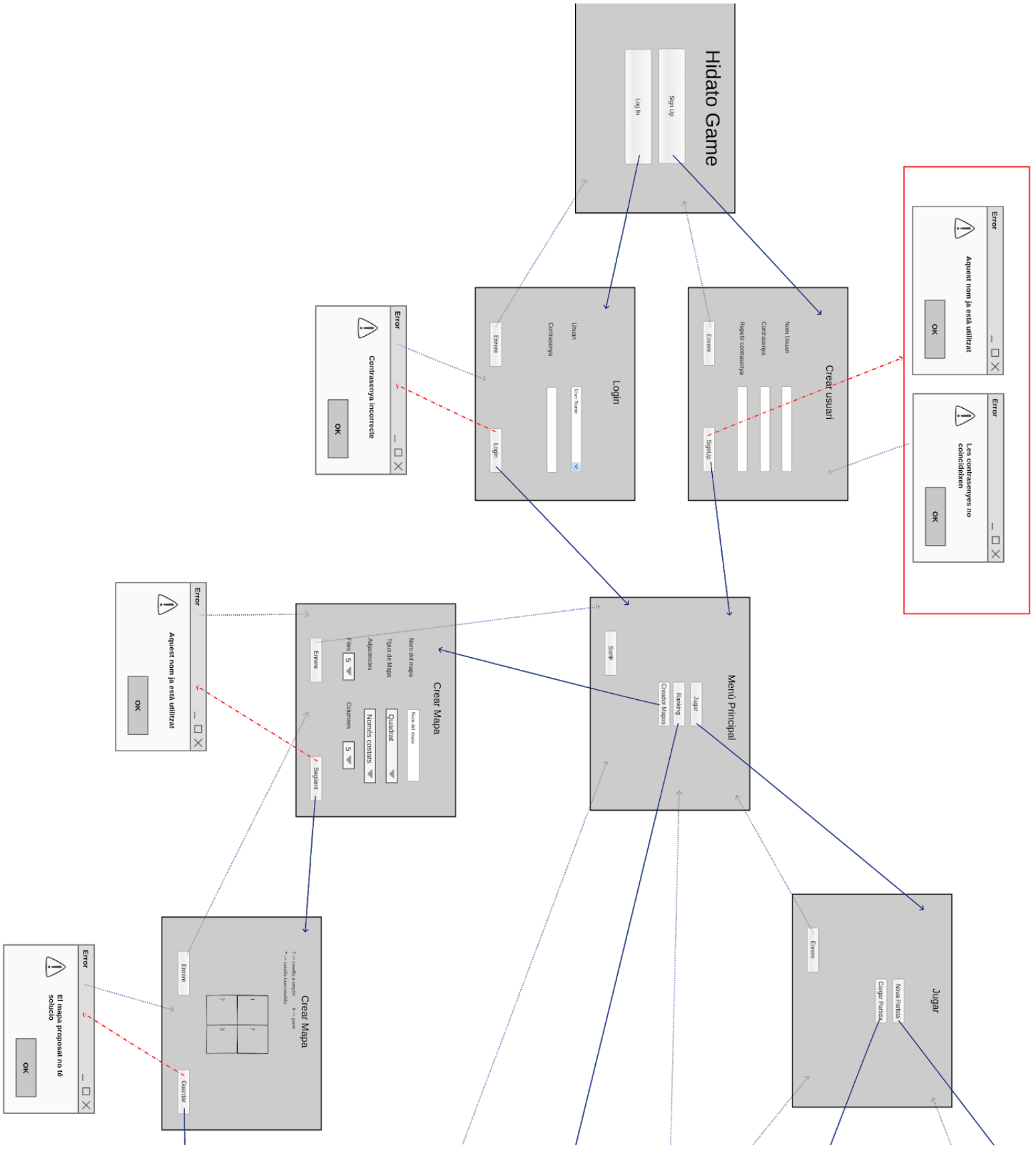
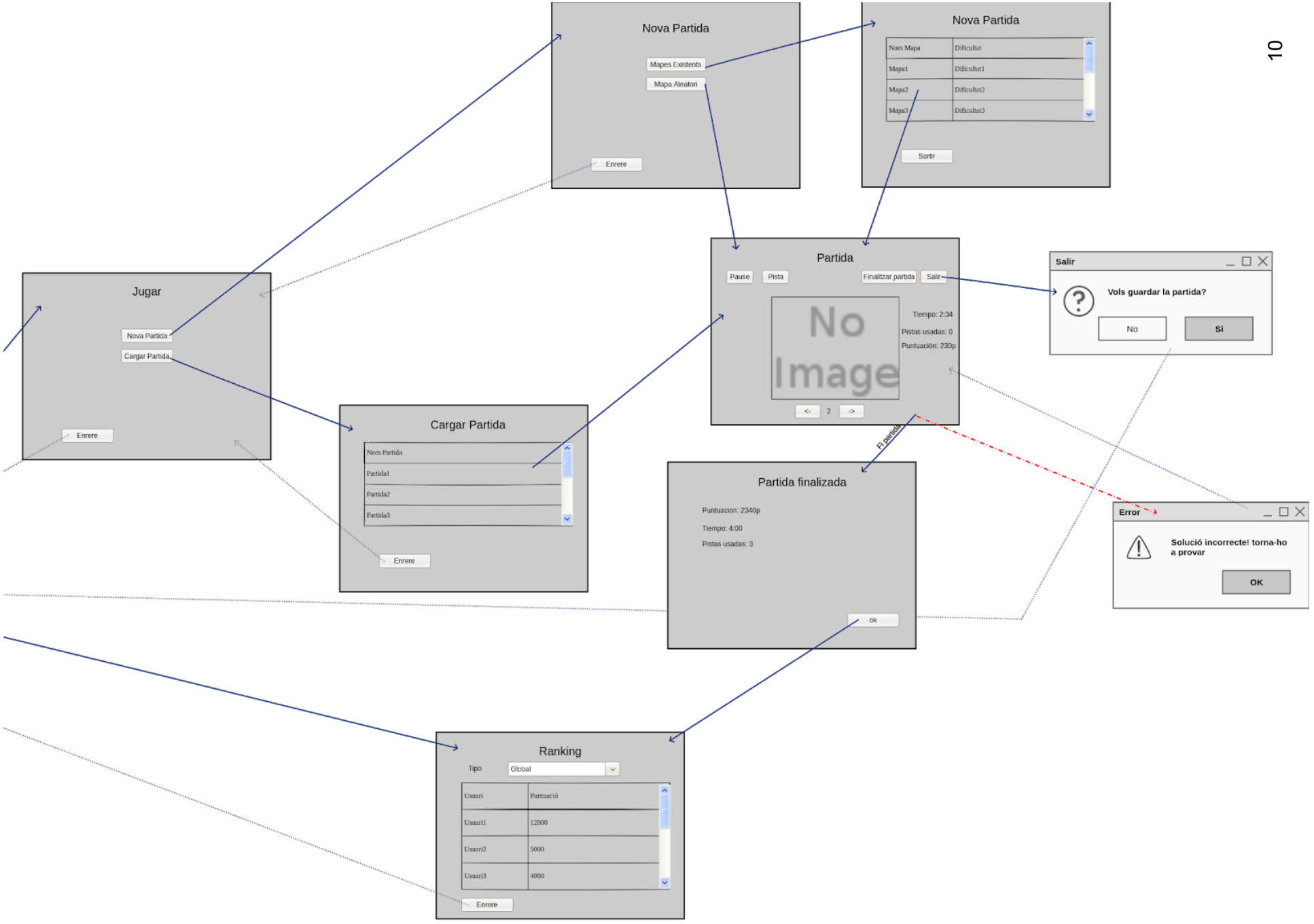


Diagrama completo en la memoria USB.

Diagrama de pantallas de la interfície




Nova Partida
Mapes Existents
Mapa Aleatori
Enrere
Nova Partida
Nom Mapa
Dificultat
Mapa1
Dificultat1
Mapa2
Dificultat2
Mapa3
Dificultat3
Sortir
Jugar
Nova Partida
Cargar Partida
Enrere
Partida
Pause
Pista
Finalitzar partida
Salir
No Image
Tiempo: 2:34
Pistas usadas: 0
Puntuación: 230p
<-
2
->
Salir
Vols guardar la partida?
No
Si
Cargar Partida
Nom Partida
Partida1
Partida2
Partida3
Enrere
Fi partida
Partida finalitzada
Puntuacion: 2340p
Tiempo: 4:00
Pistas usadas: 3
ok
Error
Solució incorrecte! torna-ho a provar
OK
Ranking
Tipo
Global
Usuari
Puntuació
Usuari1
12000
Usuari2
5000
Usuari3
4000
Enrere

Descripción de estructuras de datos y algoritmos obligatorios

Algoritmo de Resolución/validación (backtrackingResolució):

BacktrackingResolució es una función por fuerza bruta que calcula si un algoritmo puede ser resuelto y guarda la solución en uno de sus atributos. La solución se usa para calcular una Pista si el usuario la pide.

Primero calculamos la Tabla de Adyacencias del hidato, esta consiste en un vector donde en cada posición guardamos una Adycencia. Todas las casillas que tienen por valor un número o un interrogante se consideran Adyacencia. Vamos a guardar también su posición en el hidato y un identificador único para cada una (se calcula con las componentes fila y columna). Cada adyacencia tiene un booleano que indica si esta ha estado visitada y además tiene un vector de números que indican que posiciones en la Tabla de Adyacencias se encuentran Adyacencias a las cuales son adyacentes (es decir, que “se están tocando”). Cada Adyacencia tambien guarda que valor tiene en el hidato.

También tenemos un vector de números donde guardamos todos los números que por defecto ya existen en el hidato.

El backtracking consiste en calcular todos los posibles caminos que hay para ir de un número que existe en el hidato A al siguiente número que también se encuentra en el hidato B. Donde B es el número que se encuentra en la posición “i” del vector de números que ya se encuentran por defecto en el hidato y A es el de la posición “i+1”.

Estos caminos deben cumplir la condición de distancia, por ejemplo si B = 1 y A = 3, entonces solo busca caminos de distancia 2. Una vez ha encontrado todos los caminos que permiten llegar al destino A, marcaremos el primer camino como visitado (un camino es un vector de enteros, estos enteros indican que posición de la Tabla de Adyacencias hemos visitado) y lo añadiremos como parte de la solución final.

A continuación estamos en A y queremos llegar a D = 5. Entonces átodos los caminos posibles, y si no encuentra ninguno entonces deshará el backtracking, el camino de B a A lo quitará como posible parte de la solución y probará de hacer el BacktrackingResolucio con el siguiente camino que hemos calculado anteriormente.

Esta función devuelve true si se ha encontrado una resolución, en caso contrario devuelve false.

Algoritmo de Generación (generarHidato):

Este algoritmo genera un hidato de forma aleatoria que tiene como mínimo una posible resolución.

Primero definimos de forma aleatoria los parámetros que definen un hidato: número de filas, número de columnas, topología y tipo de adyacencias. Para número de filas y columnas se determina un rango.

Se define un valor aleatorio dentro de un rango (que va aproximadamente entre 1 cuarto de las casillas totales de la matriz y 3 cuartos) para asegurarnos de que hay una cierta cantidad de *, ?, # y números ya colocados. Éste valor indicará el número de casillas que han de ser números (el resto de casillas serán * o #).

A continuación utilizamos la función PathFinder correspondiente a la topología escogida, la cual va generando un camino. Este camino es de “?” y números (hay una probabilidad de que se muestre el número que va en la casilla). PathFinder utiliza una función que devuelve todas las posibles adyacencias para ese mapa y escoge una dirección aleatoria para ir encontrando el camino. También se asegura que el número de casillas que conforman el camino es el número que hemos designado anteriormente (aproximadamente entre 1 cuarto y 3 cuartos de las casillas totales del tablero). Como el camino siempre se realiza a través de las adyacencias disponibles, sabremos que siempre tiene solución. En caso de quedarse encerrado y no encontrar camino, el algoritmo descarta la solución parcial y vuelve a empezar (a efectos prácticos no afecta al rendimiento del programa en la mayoría de los casos)

Drivers

- **DriverPartida:** Driver que permite probar la clase Partida y su controlador.
    - Uso: Al iniciar el driver se crea un usuario y da la opción de escoger una partida con un mapa aleatorio o unos mapas simples de prueba. Entonces se puede jugar utilizando todas las funcionalidades que ofrece Partida.
    - Pre Condiciones: -
- **DriverPartidaDAO:** Permite testear funciones de PartidaDAO.
    - Uso: Se muestran unas instrucciones que nos permiten probar funciones de PartidaDAO
    - Pre Condiciones: -
- **DriverMapa:** Utilizado para testear las distintas operaciones del controlador y clase mapa.
    - Uso: simplemente al ejecutar el driver ya se mostrará un pequeño menú mostrando las distintas operaciones a ejecutar.
- **DriverMapaDAO**: es el driver que permite probar las operaciones de guardado y carga del disco de los mapas.
    - Uso: al ejecutarlo muestra un menú mostrando las distintas operaciones a probar.
- **DriverUsuari:** Utilizado para testear las funciones del ControladorUsuari y usuari
    - Uso: al ejecutarlo muestra un menú mostrando las distintas operaciones a testear.

Distribución del trabajo

Enric Bover

Capa Presentación:

mapaButton, mapaButtonQuadrat, mapaButtonTriangle, mapaFactoryButton, MapaView, QuadratButton, HexagonButton, TriangleUpButton, TriangleDownButton, PartidaS, mapaButtonHexagon

Implementación de la visualización e interación de los hidatos tanto a la hora de jugar como en su generación random.

Clases Dominio: adyacencias, MapaDecorator, UtilsMapaDecorator, De las subclases de Mapa: CalculoAdyacencias

Implementacion de funciones del algoritmo Resolución/Validación y Pista de un Mapa (con todo lo que supone: calcular la tabla de adyacencias, inicializar la tabla…).

Mathias Bertorelli:

PartidaDAO, ControladorPartida, Partida, Menu, CarregarPartida, MapesExistents, CreadorMapes, EditorMapes, Jugar, NovaPartida, GridEditor, GridEditorFactory, GridEditorQuadrats, GridEditorTriangles, GridEditorHexagons.
De la clase Mapa: holeChecker() i generarHidato(). De les subclasses de Mapa: els pathFinders, siguienteCasilla().

Víctor Juez

Clases:

- **Usuari**: Usuari, UsuariDAO, ControladorUsuari
- **Mapa**: ControladorMapa, MapaFactory, MapaDAO
- **Capa presentación**: Main, SingUp, Login
- **Tests:** UsuariDAOTest, ControladorUsuariTest, UsuariTest, TableroCuadradoAngulosTest, TableroCuadradoTest

Diseño general de la Arquitectura del Software del programa:

- Diseño de la estructura de las 3 capas y comunicación entre ellas a partir de los controladores.
    - Diseño de la estructura de controladores, una por clase modelo (ControladorMapa - Mapa, ControladorUsuari - Usuari ...)

- Implementación del acceso al disco y guardado de archivos.
- Diseño de la estructura de subclases de la clase Mapa y implementación del patrón factoria para la generación de sus instancias.