BoomEmoji

Análisis orgánico

Extracción de clases y relaciones a partir de las especificaciones del problema y representación de las mismas en un diagrama UML.

Una vez analizadas las especificaciones del supuesto práctico comenzamos con el diseño del modelo de datos. Las clases que se han extraído son las siguientes:

- **Configuración** (Config): esta clase almacena la configuración del juego que está iniciado o la configuración de las propiedades predeterminadas del juego que el usuario iniciará. Las propiedades de esta clase son:
    - Nivel: es un entero en el que se almacena el nivel de juego elegido por el usuario, el valor almacenado se corresponde con el número de personajes a posicionar en el tablero (NIVEL_PRINCIPIANTE=10, NIVEL_AMATEUR=30, NIVEL_AVANZADO=60)
    - Nivel del tablero: almacena, en un entero, el número de posiciones del tablero, haciéndolo corresponder con el nivel de juego (TABLERO_NIVEL_PRINCIPIANTE=8, TABLERO_NIVEL_AMATEUR=10, TABLERO_NIVEL_AVANZADO=12).
    - Imagen: imagen que se utilizará para el personaje mostrado en las casillas del tablero, por defecto se utiliza la imagen *android.R.drawable.progress_horizontal*.
    - Constantes de utilidad (todas de tipo entero) a nivel de clase:

      NIVEL_PRINCIPIANTE = 10;
      NIVEL_AMATEUR= 30;

NIVEL_AVANZADO = 60;
TABLERO_NIVEL_PRINCIPIANTE = 8;
TABLERO_NIVEL_AMATEUR = 10;
TABLERO_NIVEL_AVANZADO = 12;
IMAGEN_SIN_PERSONAJE = android.R.drawable.progress_horizontal;
IMAGEN1_NORMAL = android.R.drawable.presence_audio_away;
IMAGEN2_AMATEUR= android.R.drawable.presence_video_away;
IMAGEN3_AVANZADO_= android.R.drawable.presence_away;
IMAGEN1_NORMAL_PIERDE = android.R.drawable.presence_audio_busy;
IMAGEN2_AMATEUR_PIERDE= android.R.drawable.presence_video_busy;
IMAGEN3_AVANZADO_PIERDE = android.R.drawable.presence_busy;

- **Juego** (Juego): representa cada acto de jugar, es decir almacena la información del juego en curso y proporciona las funciones necesarias para que se desarrolle el juego. La componen los siguientes miembros:
    - Contexto (context): contexto de la actividad relacionada.
    - Configuración (config): objeto de la clase donde se almacenan todas las propiedades del juego en curso.
    - Tablero (tablero): lista de objetos de clase Personaje que representa a todas las posiciones del tablero. El número de elementos de la lista estará indicado por el nivel de juego asociado al tablero (8x8, 10x10 o 12x12).
    - Adyacentes (adyacentes): lista de objetos de la clase Personaje con los personajes del tablero adyacentes a un personaje dado. La lista se rellena cuando el jugador usa la función asociada.
    - Preparar el tablero (prepararTablero): se trata de un método (público) encargado de posicionar los personajes en el tablero. Se invoca cuando se inicia un nuevo juego y devuelve verdadero en caso de que la preparación haya sido correcta. Entre otras operaciones realiza las siguientes:
        - Genera posiciones aleatorias para los personajes en el tablero teniendo en cuenta el nivel de juego seleccionado.
        - Almacena en la propiedad *tablero* todos los personajes, los que tienen como imagen la imagen por defecto y los que contienen como imagen el personaje seleccionado por el usuario.
    - Posicionar un personaje (posicionarPersonaje): se trata de un método (privado) que se encarga de calcular la posición de los personajes en el tablero. Recibe 2 parámetros, el primero contiene la posición del personaje en la lista y el segundo la imagen correspondiente al personaje. Este método es invocado cuando se prepara el tablero. Su principal función es obtener, a partir de la posición en la lista de personajes, la fila y columna que le correspondería en el tablero (virtual). Tiene en

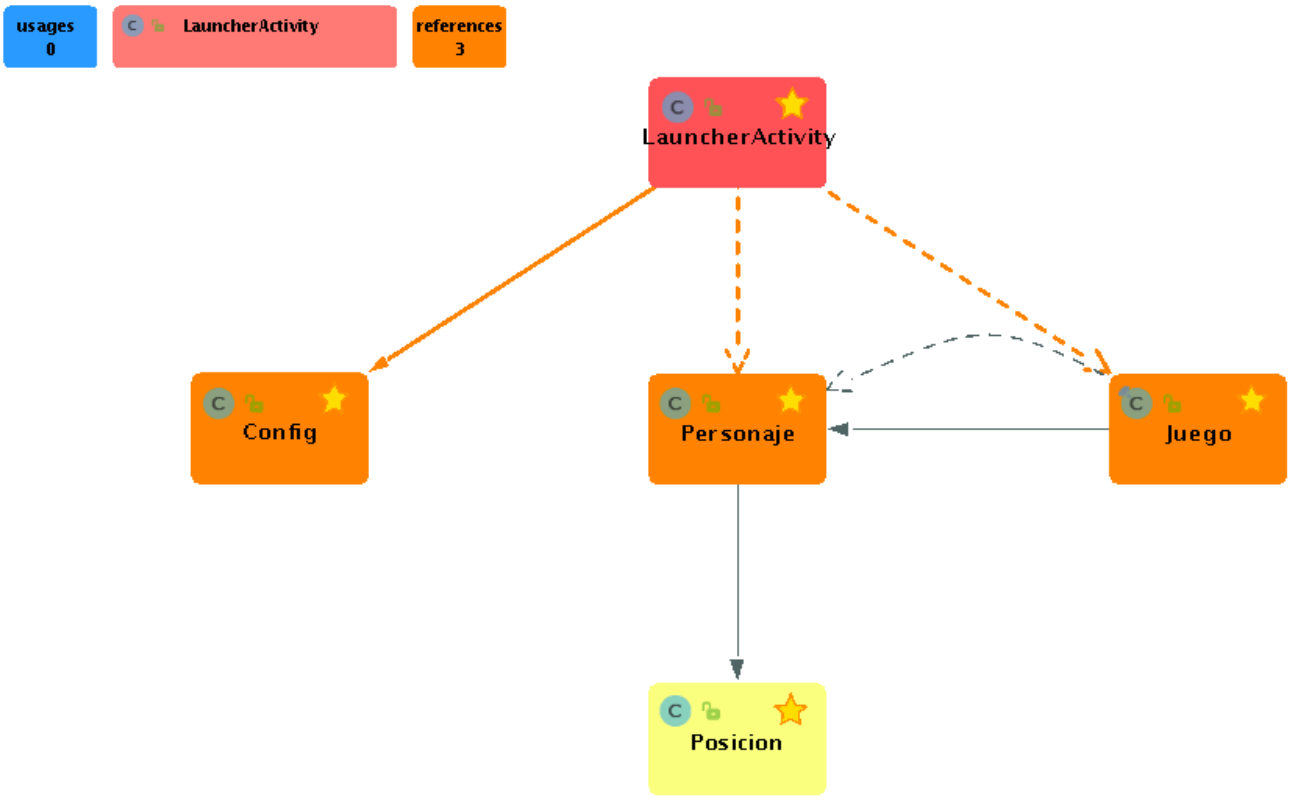
cuenta el nivel de juego seleccionado. Devuelve un objeto de la clase Personaje que contiene la configuración adecuada respecto de la posición en el tablero.
    - Calcular adyacentes (calcularAdyacentes): se trata de un método (público) que calcula las posiciones adyacentes a una dada. Recibe 2 parámetros, el primero con la fila de la posición del tablero de la que se quieren obtener los adyacentes, y el segundo con la columna. Modifica el personaje del tablero en la posición indicada en los parámetros para añadirle la lista de adyacentes.
- **Personaje** (Personaje): clase que se encarga de representar al personaje/figura del juego en una posición del tablero. Esta compuesta por los siguientes miembros:
    - Índice (índice): es una propiedad de sólo lectura y de tipo entero que almacena la posición del personaje en el tablero
    - Imagen (imagen): es una propiedad de sólo lectura y de tipo entero que almacena la referencia a la imagen del personaje.
    - Posición (posición): es una propiedad, de lectura y escritura, de la clase Posicion que almacena la posición del personaje en relación a la fila y columna del tablero.
    - Adyacentes (adyacentes): es una propiedad, de lectura y escritura, de tipo lista de personajes que almacena todos los personajes adyacentes.
    - Obtener fila (getFila): es un método (público) que devuelve la fila de un personaje en el tablero a partir de su posición dada como parámetro.
    - Obtener columna (getColumna): es un método (público) que devuelve la columna de un personaje en el tablero a partir de su posición dada como parámetro.
- **Posición** (Posicion): clase que almacena la posición relativa de un personaje en el tablero, guarda la fila y columna correspondientes. Está compuesta por:
    - Fila (fila): propiedad, de sólo lectura, de tipo entero que almacena la fila de la posición.
    - Columna(columna): propiedad, de sólo lectura, de tipo entero que almacena la columna de la posición.
- **Clase lanzadora** (LauncherActivity): es una clase de tipo actividad que controla la ejecución de la APP.

Las relaciones entre clases del modelo han quedado reflejadas en la descripción de cada una de las clases. No obstante, a continuación se exponen las mismas:

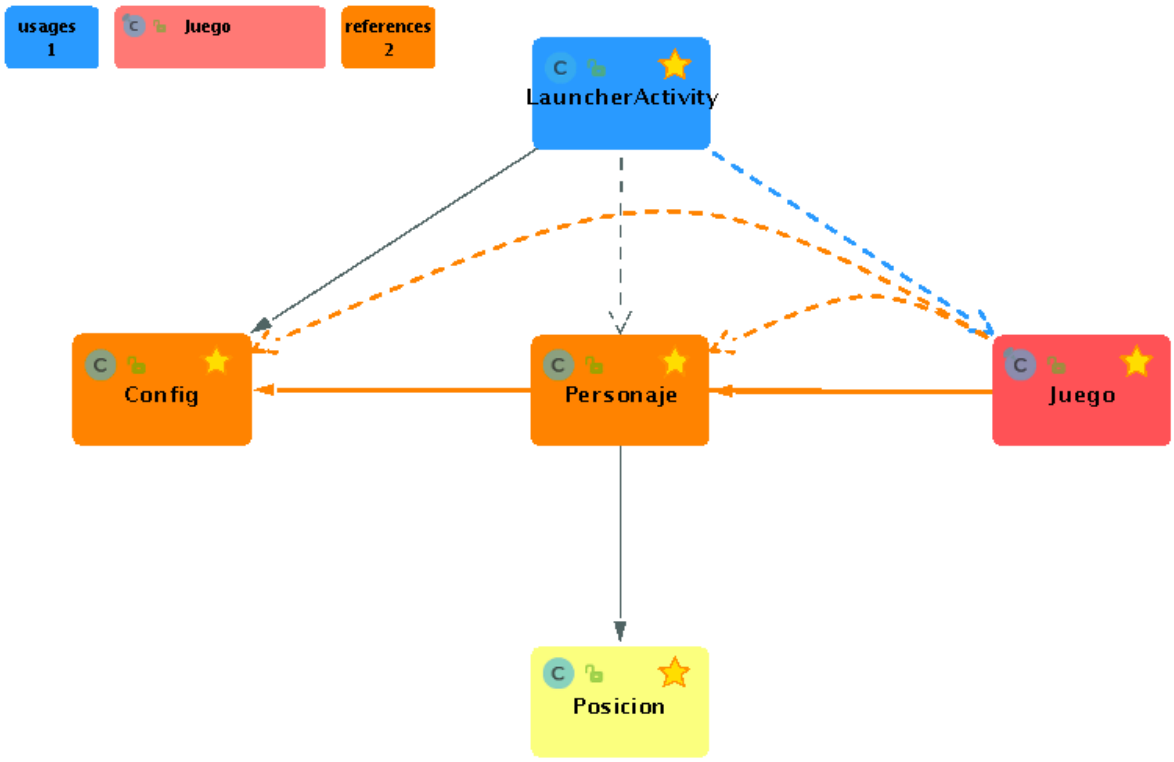
1. Un Tablero está formado por varios personajes (Personaje).
2. Un Personaje tiene una única posición (Posicion).
3. El Juego tiene un único Tablero y una única Configuracion de juego.

En las siguientes figuras se muestran los diagramas UML del modelo de datos propuesto.

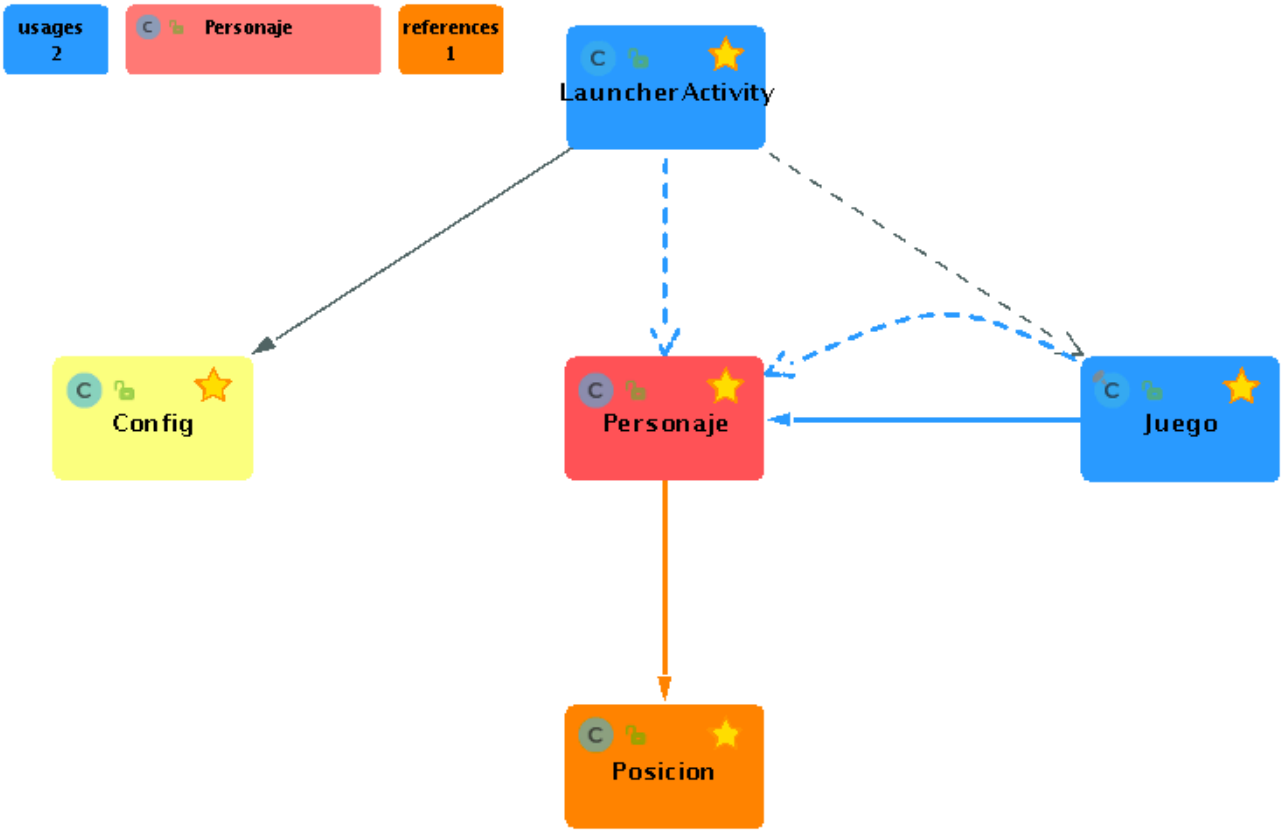
- **LauncherActivity**



- **Juego**



- **Personaje**



Análisis funcional

- A nivel funcional el juego se comporta de modo similar al juego 'Buscaminas'. La finalidad del juego es que el usuario vaya descubriendo posiciones del tablero sin que en las mismas haya un personaje. Si se selecciona una casilla del tablero se descubre el personaje oculto, si lo hay. En caso de que se descubra una casilla con personaje el juego habrá terminado y se mostrará la imagen del personaje en otro color o posición. Sino, el juego continua y se mostrará el número de personajes que hay en las posiciones adyacentes. Adicionalmente, si el jugador pulsa sobre una casilla donde no hay ningún personaje oculto se le mostrarán las casillas adjuntas donde no hay personajes ocultos. Si el jugador consigue eludir todos los personajes ocultos habrá ganado y se mostrarán todos los personajes ocultos con color o posición normales.
- La interfaz de usuario estará compuesta por:
    - Un menú de opciones:
        - Instrucciones: información sobre el juego a través de un cuadro de diálogo.

        - Nuevo juego: se inicia el juego tomando como configuración de juego la indicada por el jugador, si no lo ha hecho se utilizará la configuración por defecto (nivel principiante).
        - Configurar juego: se mostrará un menú con opciones excluyentes para seleccionar el nivel de juego (principiante, amateur, avanzado).
        - Seleccionar personaje: es una opción de menú visible en la barra de acción de la APP. Mostrará una caja de texto combinada con lista para seleccionar la imagen del personaje.

- Acciones del jugador:
    - Pulsación corta: muestra el personaje oculto en la casilla. Si había personaje oculto el jugador pierde, en caso contrario se muestra un número con los personajes ocultos adyacentes.
    - Pulsación larga: muestra todas las casillas del tablero adyacentes a la pulsada donde no hay personajes ocultos. Igualmente, si en la casilla pulsada hay un personaje oculto, el jugador pierde.