eman ta zabal zazu

Escuela Universitaria de Ingeniería Técnica Industrial de Bilbao

Grado en Informática de gestión y Sistemas de Información

Trabajo Fin de Grado

2014 / 2015

Moles' game

Memoria

Datos de la Alumna o del Alumno	**Datos del Director o de la Directora**
Nombre: Rubén	Nombre: Begoña
Apellidos: Alonso	Apellidos: Blanco Jauregui
	Departamento:
Fdo.:	Fdo.:
Fecha:	Fecha:

Índice

Índice de ilustraciones

1 Documento de objetivos del proyecto

1.1 Introducción

En los últimos años ha habido un gran crecimiento en el mercado de la telefonía móvil, es difícil encontrar alguna persona que no posea uno.

Dentro de la telefonía móvil nos podemos encontrar que cada móvil tiene un sistema operativo diferente, donde nos encontramos con Android, iOS, Blackberry, Windows Phone como los más populares pero hay otros sistemas menos conocidos y siguen apareciendo más.

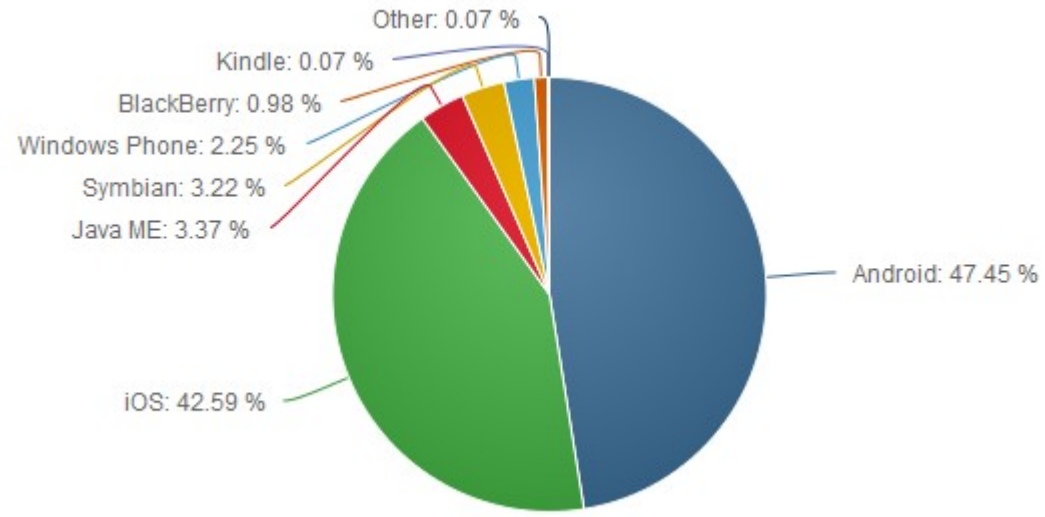


Ilustración 1: SO móviles más utilizados para 2015

[1]

Cada SO tiene sus propias características y a la hora de desarrollar un aplicación y queremos que este soportada en cada uno de estos sistemas, supone que debemos desarrollar casi desde cero para cada uno. Ahí es donde entra la tecnología web, hay una cosa que tienen en común todos los sistemas operativos y es un navegador web.

Teniendo esta característica común que es el navegador podremos desarrollar una aplicación que sea soportada en todos los sistemas operativos.

1 Ilustración 1: https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/ 25/02/2015 por Nicolás Montés

1.2 Descripción del proyecto

El proyecto consiste en desarrollar el mítico juego de los topos de las ferias, en el que van apareciendo topos a través de unos agujeros y hay que golpearlos antes de que vuelvan a desaparecer.

El juego se desarrollará en el lenguaje HTML5 que utilizando una de las nuevas etiquetas que se han añadido en el nuevo estándar, la etiqueta <*canvas*> que nos permite dibujar gráficos estáticos y animaciones a través de JavaScript.

1.3 Objetivos

El propósito del proyecto es realizar un juego en HTML5, y demostrar la gran cantidad de posibilidades que nos puede ofrecer HTML5, la tecnología web en general, a la hora de desarrollar una aplicación y que puede llegar a estar al nivel de un aplicación desarrollada en el lenguaje nativo del sistema operativo.

Así, los objetivos del proyecto son:

- Aplicar los conocimientos adquiridos durante el estudio a través de curso práctico.
- Ampliar estos conocimientos a través de la investigación y el aprendizaje para el desarrollo en páginas web.
- Lograr que la gente disfrute de la aplicación.

1.4 Arquitectura

El proyecto tendrá una arquitectura local, ya que el total de las funcionalidades se ejecutarán desde el dispositivo.

1.5 Herramientas y tecnologías a utilizar

A la hora de desarrollar un aplicación web debemos tener en cuenta la multitud de herramientas que están surgiendo, valorarlas y elegir cual son las más adecuadas para nuestra aplicación y para nosotros. Las herramientas que se utilizarán para el desarrollo del proyecto serán las siguientes.

1.5.1 Hardware

- PC con Ubuntu 14.04
- Móvil que disponga de navegador con soporte HTML5

1.5.2 Software

1. **Node.js**

Hasta ahora los entornos de programación en la capa del servidor estaban basados en lenguajes como PHP, Python..., y JavaScript era sólo un lenguaje orientado a la parte cliente. Node.js es un entorno de programación basado en JavaScript para la capa del servidor.

En este proyecto no se usará parte en el servidor pero es necesario para poder instalar y usar las herramientas que veremos a continuación.

2. **NPM(Node Package Manager)**

NPM es un gestor de paquetes para Node.js. Se usa a través de comandos para la consola y sirve para resolver las dependencias de una aplicación, así como para instalar aplicaciones de Node.js que se encuentre en el registro de NPM.

Es la herramienta que utilizaremos para instalar Grunt y Bower, y las dependencias y plugins que necesitemos para el desarrollo de la aplicación.

3. **Bower**

En el desarrollo de una aplicación web se utilizan muchas herramientas como frameworks, librerías,... Bower administra todo esto. El funcionamiento es sencillo, se necesita crear en el proyecto un archivo con el nombre *bower.json* donde escribiremos todas las cosas que necesitamos para el desarrollo de la aplicación y desde la consola deberemos ejecutar desde la consola y dentro de la carpeta donde se encuentre este archivo y ejecutar.

```
bower install
```

4. **Grunt**

Es un herramienta de automatización de procesos. Lo que conseguiremos con esta herramienta será compilar los archivos CoffeeScript, Jade, Stylus a JavaScript, HTML y CSS respectivamente.

El funcionamiento de Grunt es fácil, se basa en un fichero de configuración que puede ser escrito en JavaScript o en CoffeeScript, *gruntfile.js* o *gruntfile.coffee*. En este fichero debemos escribir que tareas queremos que sean automatizadas y para que ficheros deben ser ejecutadas.

Grunt cuenta con una gran cantidad de plugins que nos permiten automatizar tareas y en la documentación de cada uno de éstos se explica las distintas opciones que tiene y que debemos incluir en el gruntfile para realizar las funciones tal y como deseemos.

En nuestro proyecto utilizaremos plugins para compilar CoffeeScript, Jade, Stylus y otros que nos servirán para concatenar archivos y para ofuscar el código JavaScript.

5. KineticJS

Kinetic es una librería creada por Eric Rowell que nos ayudará a dibujar dentro del contexto 2d de canvas. Tiene diferentes clases para poder dibujar de manerá sencilla las diferentes posibilidades que nos ofrece canvas. También nos ayudarán a la hora de crear animaciones y transiciones, asi como añadir eventos tanto para escritorio como para móvil a los diferentes objetos que estarán contenidos en el canvas.

6. Cordova CLI

Es un plataforma que sirve para crear aplicaciones nativas móviles utlizando HTML, CSS y JavaScript. Nos ofrece diferentes plugins para poder acceder a ciertos recursos del móvil como la cámara, los contactos y tenemos una lista con unos cuantos más.

1.5.3 Lenguajes de programación

Como todas las páginas web los lenguajes de programación a utilizar son HTML, CSS y JavaScript, en concreto se utilizará HTML5 que es la ultima revisión de este lenguaje y provee un gran potencial, así como CSS3.

De hecho para el proyecto no se utilizarán estos lenguajes en si, se utilizará un lenguaje similar que más tarde se compilará a cada uno de estos.

1. **Jade**

Es un lenguaje para escribir plantillas HTML. En Jade en vez de utilizar el lenguaje de etiquetas que conocemos en HTML que utiliza un etiqueta de apertura *<etiqueta>* para abrir una sección y otra *</etiqueta>* para cerrar, se utiliza la indentación y sólo se escribe el nombre de la etiqueta. Una etiqueta empieza en el momento que aparece escrita y acaba justo cuando empieza otra en esa misma linea. Todas las etiquetas que tengan una indentación sobre otra padre estarán contenidas dentro de esta.

Ejemplo.

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <h1>Hello World!!!</h1>
  </body>
</html>
```

Ilustración 2: HTML

Jade

```
doctype html

html
  head
    title Example
  body
    h1 Hello World!!
```

Ilustración 3: Jade

2. **Stylus**

Stylus es un preprocesador de CSS, en el cual tenemos la posibilidad de utilizar variables, funciones, operadores (+, -, *)... así como muchas otras características que nos facilitan mucho a la hora de escribir un archivo CSS.

Al contrario de CSS en Stylus no es necesario escribir los 2 puntos (:) después de una propiedad CSS ni el punto y coma (;) al final, así como tampoco son necesarios los corchetes. Para definir que propiedades pertenecen a cada componente del DOM solo habría que indentar esa propiedad.

Ejemplo.

CSS:

```css
body {
  background-color: #000;
  color: #fff;
  padding: 1em;
}
body article {
    color: #000;
    padding: 1em;
}
```

Ilustración 5: CSS

Stylus:

```
WHITE = #fff
BLACK = #000
AIR = 1em

body
  background-color BLACK
  color WHITE
  padding AIR
  article
    color BLACK
    padding AIR
```

Ilustración 4: Stylus

3. CoffeeScript

Es el lenguaje de programación que se compilará después a JavaScript. Se ha elegido este lenguaje por su facilidad de comprensión al leerlo.

Se utiliza la identificación en vez de llaves para definir el principio el final de las funciones y métodos y no se utiliza el punto y como (;) para definir el final de las lineas, así como simplifica la sintaxis en algunas sentencias como los *for* …, *switch*, con eso conseguimos que los programas se escriban en número reducido de líneas.

Ejemplo.

JavaScript*:*

```javascript
/*Función que muestra uno a uno los
elementos de un array por la consola
 */
var array, showInConsole;

array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

showInConsole = function() {
  var elem, _i, _len;
  for (_i = 0, _len = array.length; _i < _len; _i++) {
    elem = array[_i];
    console.log(elem);
  }
};
```

Ilustración 6: JavaScript

CoffeeScript*:*

```
###Función que muestra uno a uno los
elementos de un array por la consola###

array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

showInConsole = ->
  for elem in array
    console.log elem
```

Ilustración 7: CoffeeScript

1.6 Alcance del proyecto

En este apartado se analizará las partes de las que constará el proyecto así como la estimación de los tiempos que se dedicarán a cada una de las partes.

1.6.1 Fases del proyecto

- **Gestión:** En esta fase del proyecto se buscará información sobre herramientas que existan para poder realizar el proyecto, así como aplicaciones que puedan ser similares para ver el comportamiento. También se definirán las tareas de las que se compondrá el proyecto y la estimación de los tiempos.
- **Análisis:** Se estudiará que funcionalidades tendrá la aplicación y, en base a ellas, se elegirán las herramientas que necesitáramos.
- **Diseño:** Aquí se diseñara la aplicación los más óptima posible teniendo en cuenta que podrá ser soportada en dispositivos con distintas dimensiones y sistemas operativos.
- **Implementación:** En esta fase plasmaremos el algoritmo y diseño de la fase anterior a código.
- **Pruebas:** A medida que se vayan implementando las diferentes funcionalidades se harán pruebas para comprobar que todo avanza correctamente, y corregir las cosas que no funcionen como estaba previsto.

1.6.2 Estructura de descomposición del trabajo

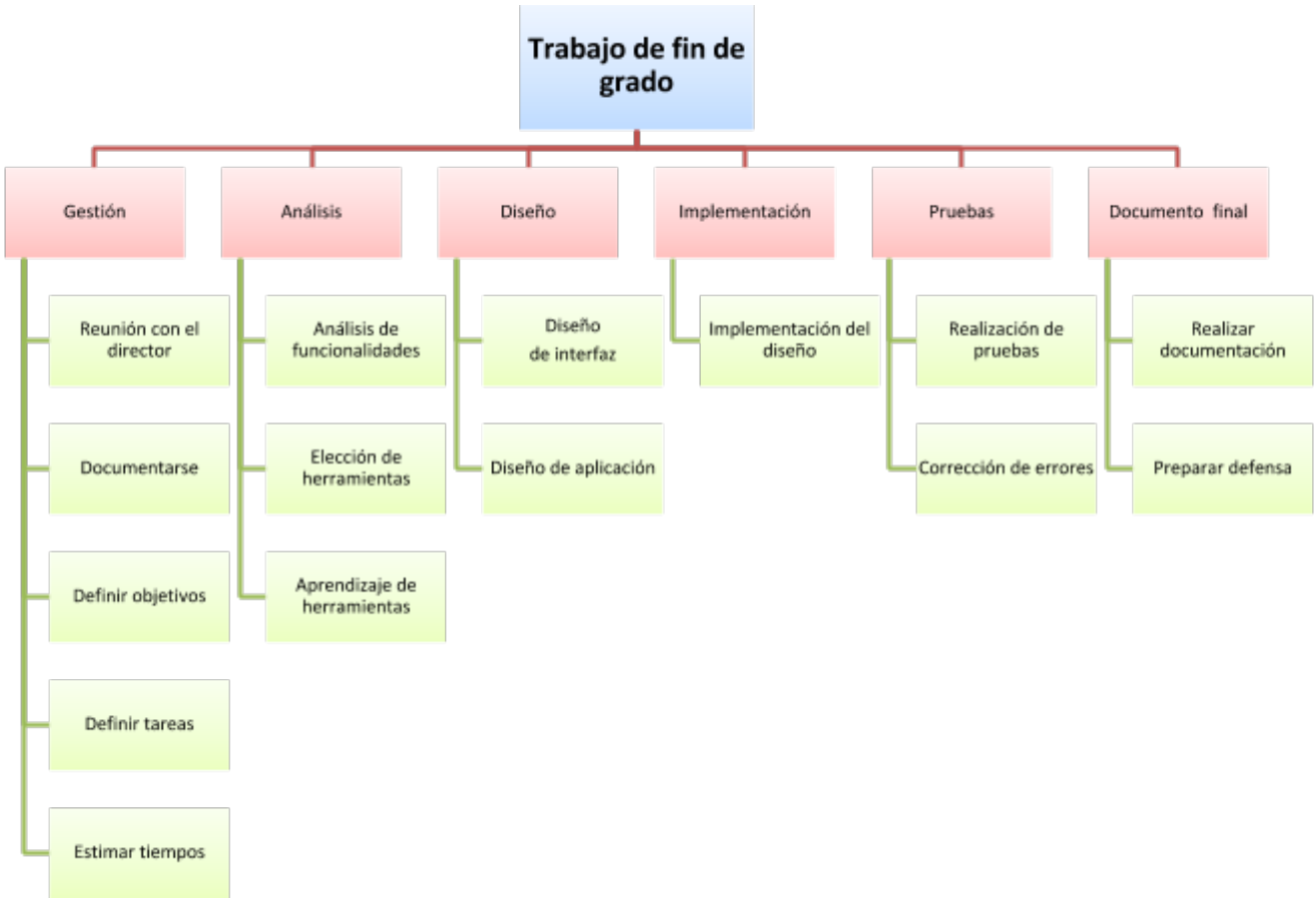


Ilustración 8: EDT

1.6.3 Tareas

1.6.3.1 Gestión

Paquete de trabajo: Reunión con el director

Duración: 1 hora

Descripción

Realizar reunión con el director con el fin de resolver dudas, objetivos, funcionalidades y herramientas.

Recursos

Tener idea clara del proyecto

Salidas

Acta de la reunión

Paquete de trabajo: Documentarse

Duración: 5 horas

Descripción

Documentarse sobre las posibilidades de HTML5 así como de las diferentes herramientas que existen.

Recursos

Ninguno

Salidas

Nuevos conocimientos adquiridos.

Paquete de trabajo: Definir objetivos

Duración: 1 hora

Descripción

Decidir los objetivos que debe cumplir el proyecto.

Recursos

Tener conocimientos sobre el lenguaje HTML5.

Salidas

Objetivos del proyecto.

Paquete de trabajo: Definir tareas

Duración: 2 horas

Descripción

Identificar cada una de las tareas a realizar.

Recursos

Saber que va a hacer la aplicación.

Salidas

Las tareas que se van a realizar durante el proyecto.

Paquete de trabajo: Estimar tiempos

Duración: 3 horas

Descripción

Analizar cada una de las tareas y estimar el tiempo que dedicaremos a cada una de ellas.

Recursos

Tener las tareas definidas.

Salidas

Las estimación de los tiempos.

1.6.3.2 Análisis

Paquete de trabajo: Análisis de las funcionalidades

Duración: 10 horas

Descripción

Definir las funcionalidades que tendrá la aplicación, en base a las posibilidades que nos ofrece el lenguaje.

Recursos

Conocer las posibilidades que nos ofrece el desarrollo web.

Salidas

Funcionalidades de la aplicación.

Paquete de trabajo: Elección de las herramientas.

Duración: 4 horas

Descripción

Elegir las herramientas que mejor se adecuen para la implementación de las funcionalidades elegidas.

Recursos

Conocer las herramientas de las que disponemos

Precedentes

Documentarse sobre las herramientas que existen.

Salidas

Funcionalidades de la aplicación.

Paquete de trabajo: Aprendizaje de herramientas.

Duración: 6 horas

Descripción

Documentarse más a fondo sobre las herramientas que vamos a utilizar y aprender a utilizarlas correctamente.

Recursos

Conocer las herramientas de las que disponemos

Precedentes

Haber escogido las herramientas.

Salidas

Mayor conocimiento sobre las herramientas que vamos a utilizar.

1.6.3.3 Diseño

Paquete de trabajo: Diseño de interfaz

Duración: 12 horas

Descripción

Diseñar las distintas pantallas que compondrán la interfaz de la aplicación.

Recursos

Tener una idea clara de las funcionalidades que se van a implementar.

Precedentes

Ninguno.

Salidas

Interfaz de la aplicación

Paquete de trabajo: Diseño de la aplicación.

Duración: 50 horas

Descripción

Diseñar el algoritmo teniendo en cuenta todo lo aprendido durante el proceso de documentación.

Recursos

Tener claras las funcionalidades que vamos a implementar, la interfaz diseñada y conocer las herramientas que vamos a utilizar.

Precedentes

Análisis

Salidas

Algoritmo de la aplicación.

1.6.3.4 Implementación

Paquete de trabajo: Implementación de la aplicación.

Duración: 200 horas.

Descripción

Implementar el algoritmo que hemos obtenido en la fase de diseño.

Recursos

Ninguno.

Precedentes

Diseño de la aplicación.

Salidas

Implementación de la aplicación.

1.6.3.5 Pruebas

Paquete de trabajo: Realización de pruebas

Duración: 30 horas

Descripción

Diseñar las pruebas y probarlas contra la implementación para sacar a relucir posibles fallos.

Recursos

Ninguno

Precedentes

Implementación de la aplicación.

Salidas

Resultado de las pruebas.

Paquete de trabajo: Corrección de errores

Duración: 20 horas.

Descripción

Corregir los errores que nos hayan aparecido en la realización de las pruebas.

Recursos

Conocer las herramientas de las que disponemos

Precedentes

Realizar las pruebas.

Salidas

Aplicación funcional.

1.6.3.6 Documento final

Paquete de trabajo: Realizar documentación

Duración: 20 horas.

Descripción

Realizar la documentación sobre todo el proceso del proyecto.

Recursos

Ninguno

Precedentes

Implementación

Salidas

Memoria del proyecto.

Paquete de trabajo: Preparar defensa

Duración: 10 horas.

Descripción

Preparar la defensa del proyecto ante el tribunal.

Recursos

Memoria del proyecto.

Precedentes

Implementación de la aplicación.

Salidas

Defensa del proyecto.

1.7 Planificación temporal

Nombre	Fecha Inicio	Fecha Fin	Días	Horas
Proyecto fin de carrera	**25/04/15**	**11/07/15**	**77**	**374**
Gestión	**25/04/15**	**28/04/15**	**3**	**12**
Reunión con el director	25/04/15	26/04/15	1	1
Documentarse	26/04/15	27/04/15	1	5
Definir objetivos	27/04/15	28/04/15	1	1
Definir tareas	27/04/15	28/04/15	1	2
Estimar tiempos	27/04/15	28/04/15	1	3
Análisis	**30/04/15**	**07/05/15**	**7**	**40**
Definir funcionalidades	30/04/15	02/05/15	2	12
Elegir herramientas	02/05/15	03/05/15	1	6
Aprendizaje herramientas	03/05/15	07/05/15	4	22
Diseño	**08/05/15**	**23/05/15**	**15**	**62**
Diseñar Interfaz	08/05/15	10/05/15	2	12
Diseñar aplicación	11/05/15	23/05/15	12	50
Implementación	**24/05/15**	**03/07/15**	**40**	**180**
Implementar el diseño	24/05/15	03/07/15	40	180
Pruebas	**28/06/15**	**03/07/15**	**5**	**50**
Realizar pruebas	28/06/15	03/07/15	5	30
Corrección errores	29/06/15	03/07/15	4	20
Documento final	**04/07/15**	**11/07/15**	**7**	**30**
Realizar memoria	04/07/15	08/07/15	4	20
Planificar defensa	09/07/15	11/07/15	2	10

Ilustración 9: Planificación temporal

1.7.1 Gannt

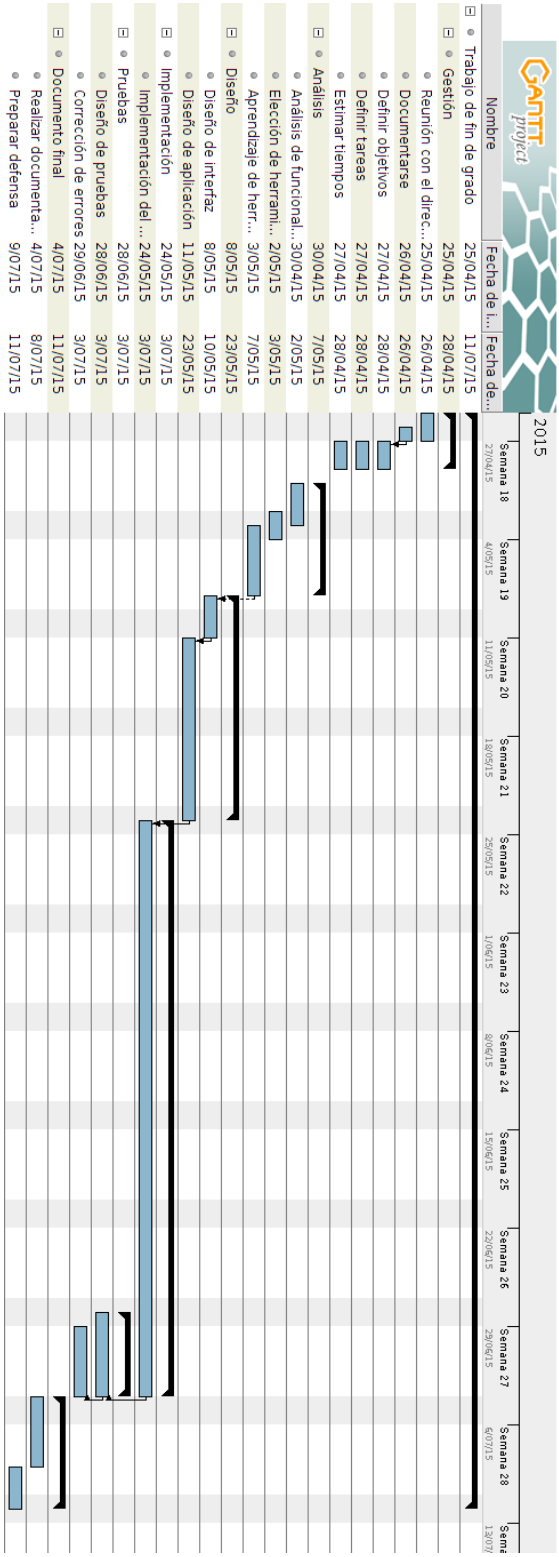
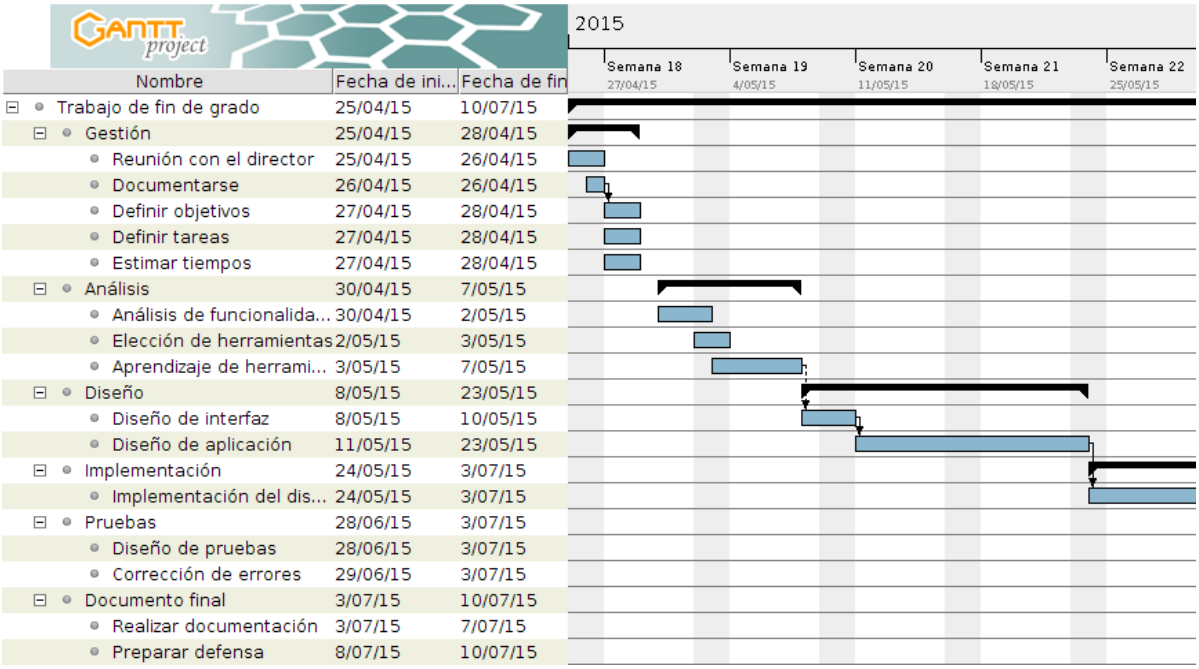


Ilustración 10: Gannt

Se dividirá en partes para que se más sencillo de ver.

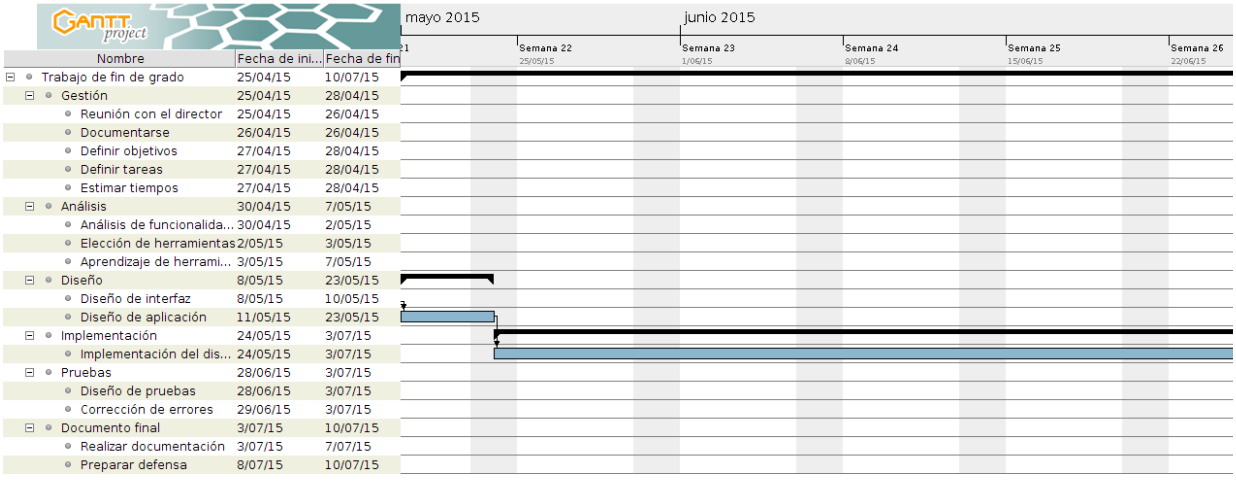
(Gestión, Análisis y Diseño)



Nombre	Fecha de ini...	Fecha de fin
Trabajo de fin de grado	25/04/15	10/07/15
Gestión	25/04/15	28/04/15
Reunión con el director	25/04/15	26/04/15
Documentarse	26/04/15	26/04/15
Definir objetivos	27/04/15	28/04/15
Definir tareas	27/04/15	28/04/15
Estimar tiempos	27/04/15	28/04/15
Análisis	30/04/15	7/05/15
Análisis de funcionalida...	30/04/15	2/05/15
Elección de herramientas	2/05/15	3/05/15
Aprendizaje de herrami...	3/05/15	7/05/15
Diseño	8/05/15	23/05/15
Diseño de interfaz	8/05/15	10/05/15
Diseño de aplicación	11/05/15	23/05/15
Implementación	24/05/15	3/07/15
Implementación del dis...	24/05/15	3/07/15
Pruebas	28/06/15	3/07/15
Diseño de pruebas	28/06/15	3/07/15
Corrección de errores	29/06/15	3/07/15
Documento final	3/07/15	10/07/15
Realizar documentación	3/07/15	7/07/15
Preparar defensa	8/07/15	10/07/15

Ilustración 11: Gannt (Gestión, Análisis, Diseño)

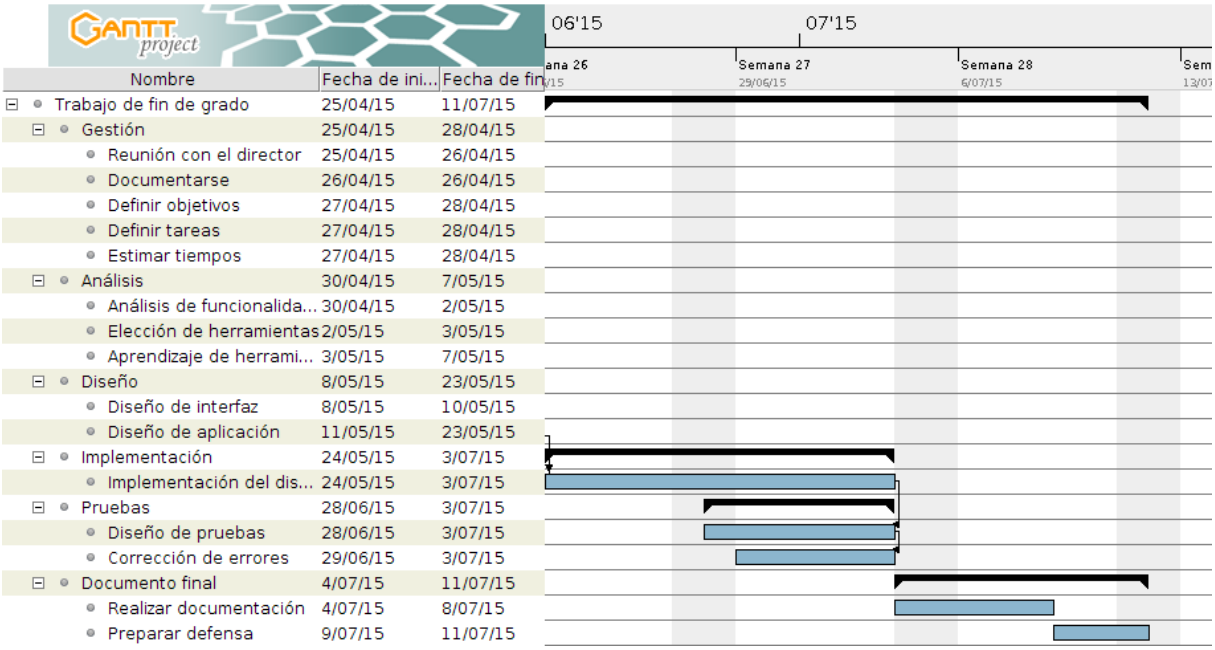
(Implementación)



Nombre	Fecha de ini...	Fecha de fin
Trabajo de fin de grado	25/04/15	10/07/15
Gestión	25/04/15	28/04/15
Reunión con el director	25/04/15	26/04/15
Documentarse	26/04/15	26/04/15
Definir objetivos	27/04/15	28/04/15
Definir tareas	27/04/15	28/04/15
Estimar tiempos	27/04/15	28/04/15
Análisis	30/04/15	7/05/15
Análisis de funcionalida...	30/04/15	2/05/15
Elección de herramientas	2/05/15	3/05/15
Aprendizaje de herrami...	3/05/15	7/05/15
Diseño	8/05/15	23/05/15
Diseño de interfaz	8/05/15	10/05/15
Diseño de aplicación	11/05/15	23/05/15
Implementación	24/05/15	3/07/15
Implementación del dis...	24/05/15	3/07/15
Pruebas	28/06/15	3/07/15
Diseño de pruebas	28/06/15	3/07/15
Corrección de errores	29/06/15	3/07/15
Documento final	3/07/15	10/07/15
Realizar documentación	3/07/15	7/07/15
Preparar defensa	8/07/15	10/07/15

Ilustración 12: Gannt (Implementación)

(Implementación, Pruebas, Documento final)



Nombre	Fecha de ini...	Fecha de fin
Trabajo de fin de grado	25/04/15	11/07/15
Gestión	25/04/15	28/04/15
Reunión con el director	25/04/15	26/04/15
Documentarse	26/04/15	26/04/15
Definir objetivos	27/04/15	28/04/15
Definir tareas	27/04/15	28/04/15
Estimar tiempos	27/04/15	28/04/15
Análisis	30/04/15	7/05/15
Análisis de funcionalida...	30/04/15	2/05/15
Elección de herramientas	2/05/15	3/05/15
Aprendizaje de herrami...	3/05/15	7/05/15
Diseño	8/05/15	23/05/15
Diseño de interfaz	8/05/15	10/05/15
Diseño de aplicación	11/05/15	23/05/15
Implementación	24/05/15	3/07/15
Implementación del dis...	24/05/15	3/07/15
Pruebas	28/06/15	3/07/15
Diseño de pruebas	28/06/15	3/07/15
Corrección de errores	29/06/15	3/07/15
Documento final	4/07/15	11/07/15
Realizar documentación	4/07/15	8/07/15
Preparar defensa	9/07/15	11/07/15

Ilustración 13: Gannt (Implementación, Pruebas, Documento final)

1.8 Riesgos

En este punto se identificarán los posibles riesgos que puedan ocurrir y suponer retrasos a la hora de la realización del proyecto.

1.8.1 Dificultades para realizar la implementación

- **Prevención:** Dedicar tiempo al aprendizaje de las herramientas que se utilizarán para la implementación del proyecto.
- **Plan de contingencia:** Buscar solución en documentación y/o ejemplos.
- **Probabilidad:** Alta
- **Impacto:** Medio

1.8.2 Problemas técnicos o perdida de equipo

- **Prevención:** Realizar copias de seguridad de la documentación y utilizar un servidor para el control de versiones para el código.
- **Plan de contingencia:** Recuperar las copias de seguridad hechas,
- **Probabilidad:** Baja
- **Impacto:** Alto

1.8.3 Problemas personales

- **Prevención:** Debido a su naturaleza, sería imposible preverlo.
- **Plan de contingencia:** Buscar solución en documentación y/o ejemplos.
- **Probabilidad:** Baja
- **Impacto:** Depende del tipo de problema.

2 Análisis de antecedentes

El formato elegido del juego se basa en el típico juego de los topos que se podía encontrar en las ferias, en el que había que golpear con un martillo los topos que iban apareciendo en los agujeros sin dejar que se escondan.

Este juego ha sido portado a distintas plataformas como PC y móviles, un ejemplo de estás aplicaciones son:

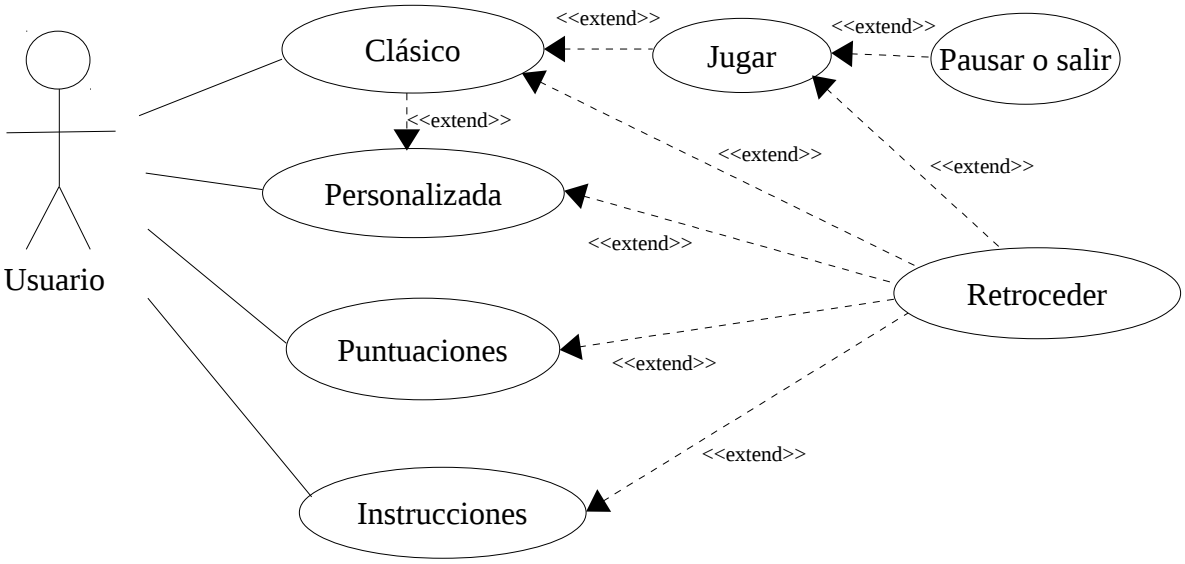
- **Mole!!!:** Este juego está disponible en Play Store para la plataforma Android, donde hay que golpear a los topos para ir aumentando de niveles.
- **Aplasta el topo:** Este juego también está disponible para plataforma Android, a diferencia del anterior juego este no tiene niveles sino un tiempo que se va consumiendo y tienes que conseguir el máximo de puntuación en ese tiempo. Hay distintos topos que te pueden dar tiempo y más puntos, así como restarlo.
- **Whack a Mole:** Es un juego para la plataforma Windows Phone. Tiene 2 versiones, una clásica que consiste en golpear a los topos, sin golpear las lombrices que aparecen y va aumentando la velocidad en la que aparecen. Y el modo “arcade” que es golpear el máximo número de topos en 60 segundos.
- **Whac a Mole:** Este es un versión más moderna de este juego, llevado al 3D para las plataformas de Apple (iPhone, iPod Touch, iPad).

Hit The Mole no dispondrá de niveles de dificultad sino que la dificultad irá aumentando progresivamente durante la partida, a medida que se vayan golpeando topos el intervalo en el que van apareciendo de sus agujeros irá disminuyendo. Al igual que alguno de los ejemplos anteriores dispondrá de distintos tipos de topos que darán ayudas o perjudicarán la partida.

La diferencia principal que aportaremos al juego será dar la capacidad de poder utilizar funcionalidades como la cámara y la galería para poder personalizar la imagen que saldrá como topo. Además, la implementación se hará en HTML5 y eso nos proporcionará un portabilidad a diferentes plataformas sin tener que reescribirla para cada una de ellas.

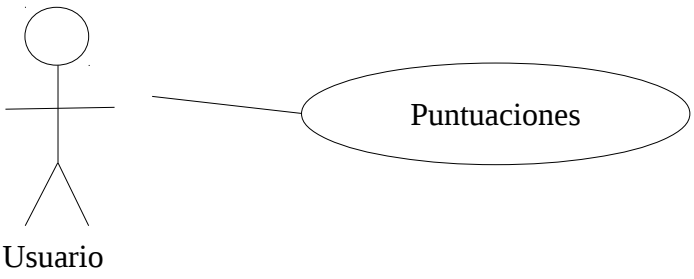
3 Captura de requisitos

3.1 Casos de uso



3.2 Casos de uso extendidos

3.2.1 Ver puntuaciones



Nombre: Puntuaciones

Descripción: Se podrá visualizar las ultimas puntuaciones y compartir en las redes sociales la* puntación más alta.

Actores: Usuario

Precondiciones: Haber jugado alguna partida para tener alguna puntuación para visualizar

Flujo de Eventos:

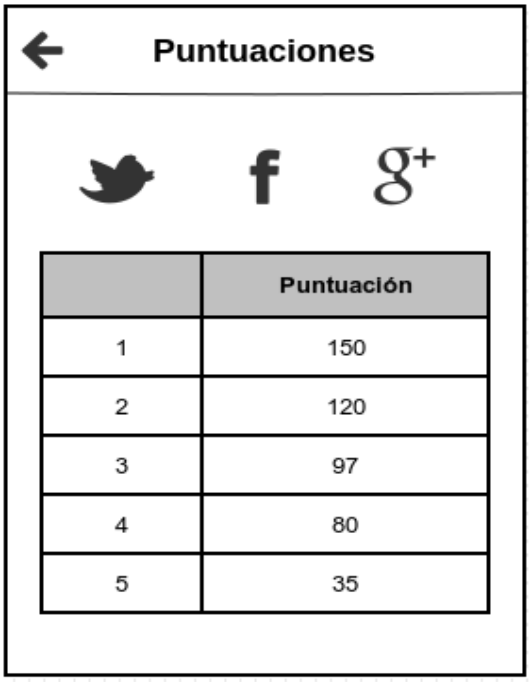
1. El jugador inicia la aplicación
2. Pulsar el botón de “Puntuaciones”
3. ¿Quiere compartir la máxima puntuación?

   Si.

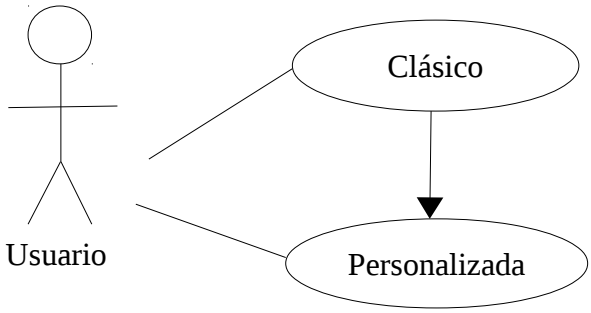
   Pulsar en el botón de la red social en la que se quiera compartir.

Poscondiciones: Ninguna

Interfaces gráficas:



3.2.2 Clásico



Nombre: Clásico

Descripción: Podemos elegir las dimensiones que va a tener el tablero

Actores: Usuario

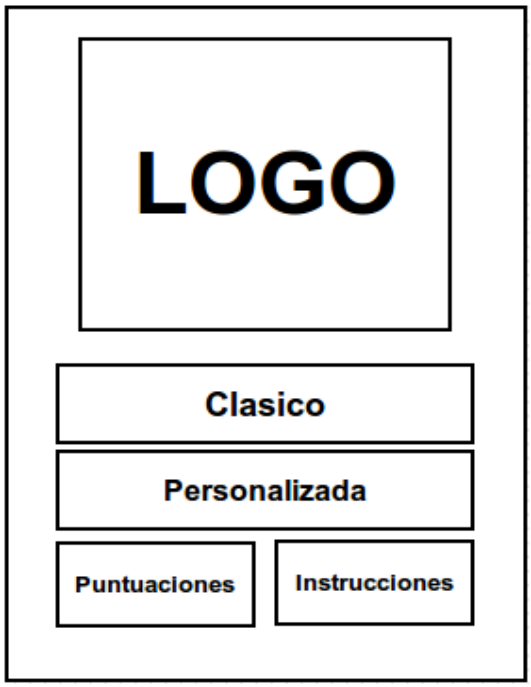
Precondiciones: Ninguna.

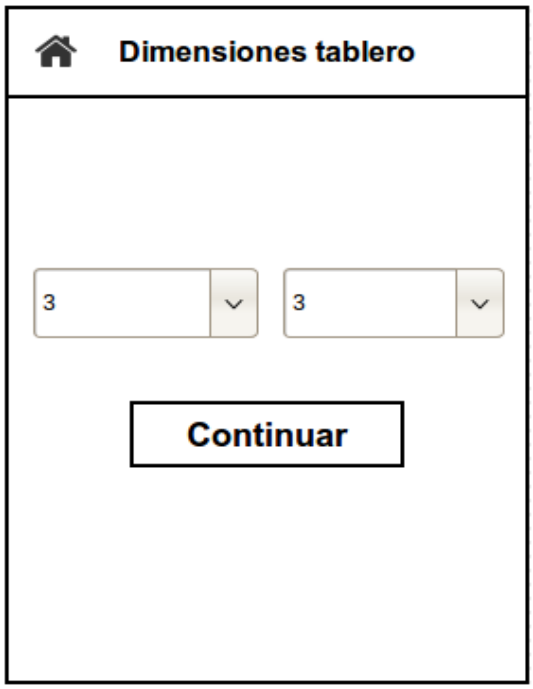
Flujo de Eventos:

1. Pulsar en “Clásico” o en el botón de jugar una vez personalizado los topos
2. Elegir dimensiones número de filas y columnas
3. Pulsar “Continuar”
4. Se inicializa el tablero con las dimensiones elegidas

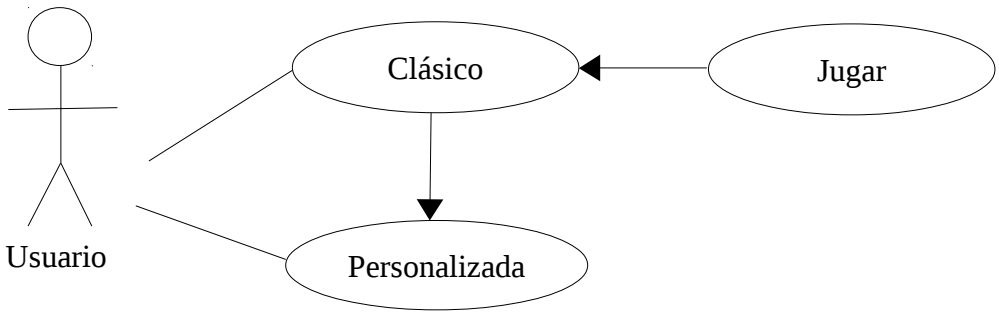
Postcondiciones: Ninguna

Interfaces gráficas:





3.2.3 Jugar



Nombre: Jugar

Descripción: Se desarrolla todo el proceso de juego

Actores: Usuario

Precondiciones: Ninguna.

Flujo de Eventos:

1. Pulsar el botón “Jugar”
2. Se inicia el [Intervalo]

   [Intervalo]

   1. Pulsar topos
   2. ¿Es el topo correcto?
      1. Si

         Sumar puntos.

         1. ¿Has alcanzado nuevo nivel?
            1. Si

               Aumentamos velocidad
      2. No.
         1. Restamos vidas
         2. ¿Quedan vidas?
            1. No

               Fin de la partida

   [Fin de intervalo]

[Una vez finalizada la partida]

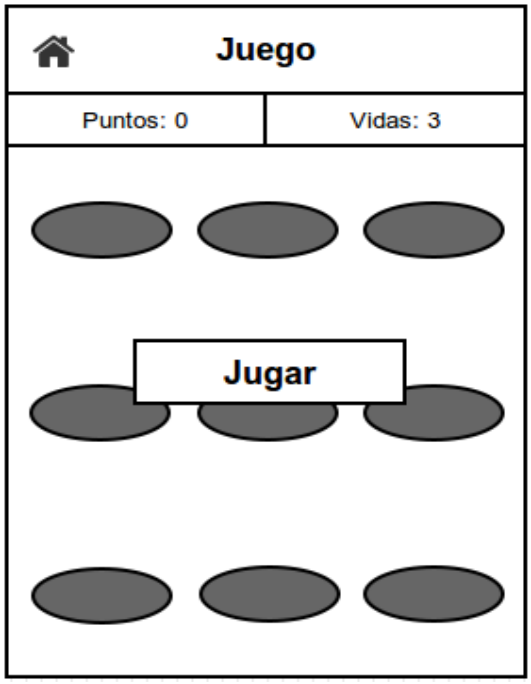
3. Guardar puntuación
4. ¿Quieres volver a jugar?
   1. Si

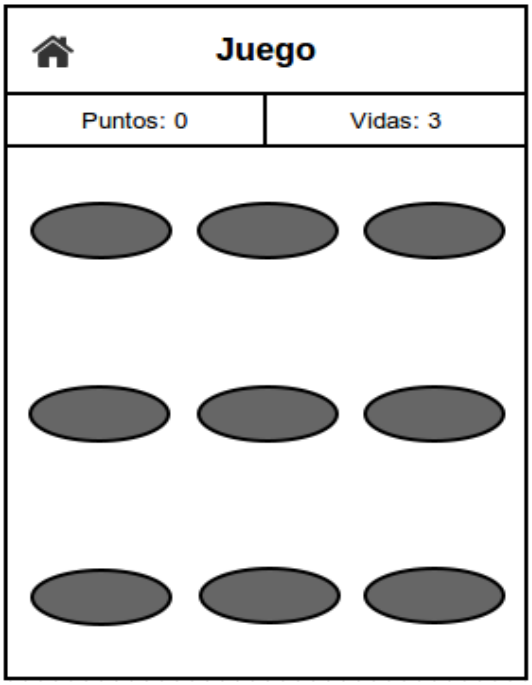
      Reinciamos partida y volveriamos al punto 1.
   2. No

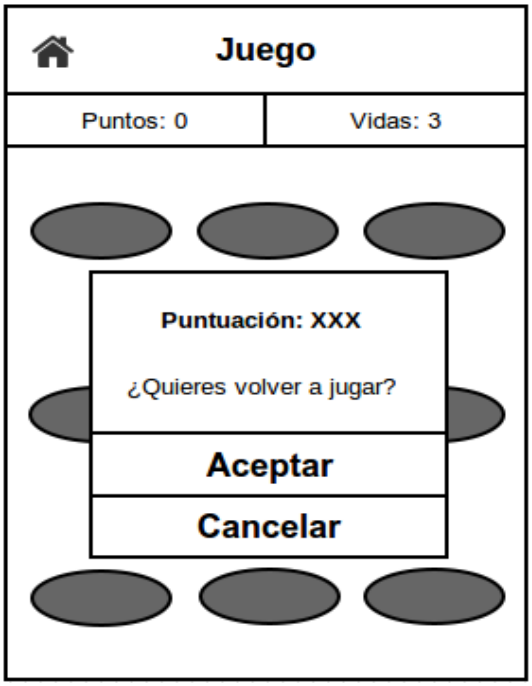
      Salimos a la pantalla principal

Poscondiciones: Ninguna

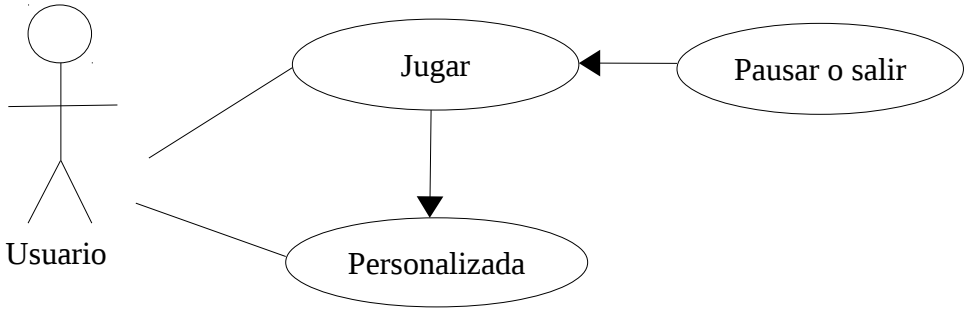
Interfaces gráficas:







3.2.4 Pausar o salir



Nombre: Jugar

Descripción: Se desarrolla todo el proceso de juego

Actores: Usuario

Precondiciones: Ninguna.

Flujo de Eventos:

1. Pulsar el botón con el símbolo de la casa.
2. ¿Queremos salir de la partida?
    1. Si

       Salimos a la pantalla principal
    2. No.

       [si no ha empezado la partida]

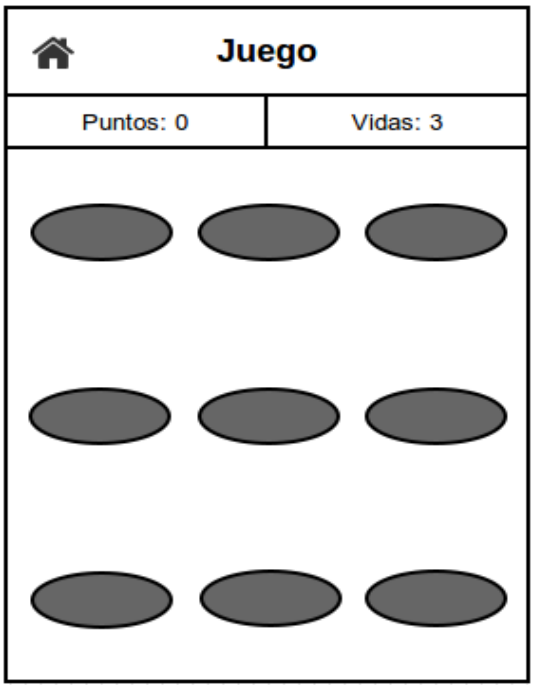
       1. Cerramos la pantalla de salir

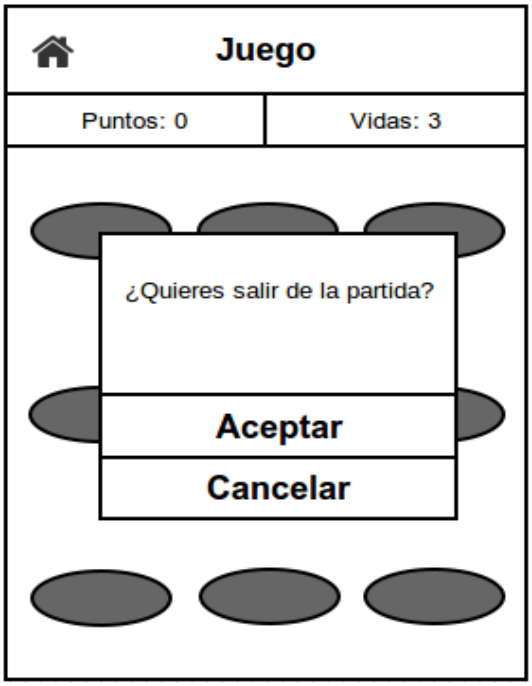
       [si ha empezado la partida]

       2. Reiniciamos el [Intervalo] para seguir jugando

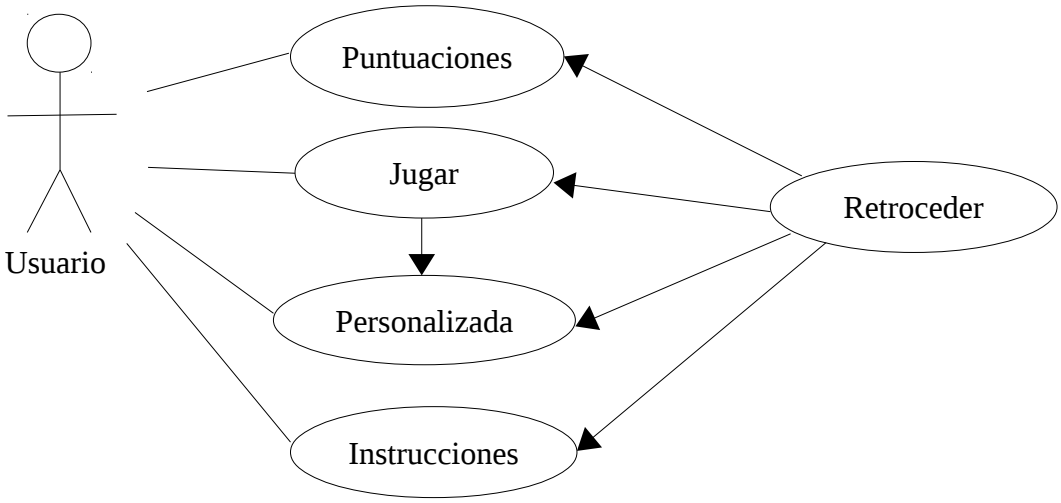
Poscondiciones: Ninguna

Interfaces gráficas:





3.2.5 Retroceder



Nombre: Retroceder

Descripción: Controlar la navegación de la aplicación cuando se pulsa el botón de “Atrás” en el navegador así como el de un dispositivo móvil.

Actores: Usuario

Precondiciones: Ninguna.

Flujo de Eventos:

1. Pulsar el botón de retroceder el dispositivo.
2. ¿Estamos en la pantalla inicial?
    1. Si

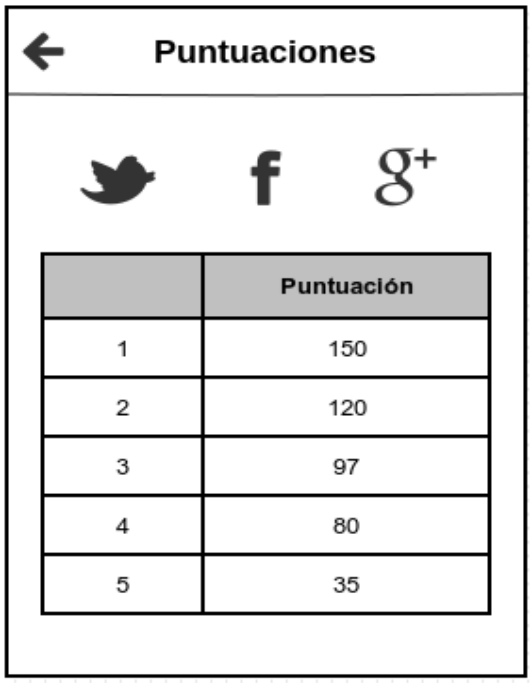
       No hace nada
    2. No
        1. ¿Estamos en la pantalla de juego?
            1. Si

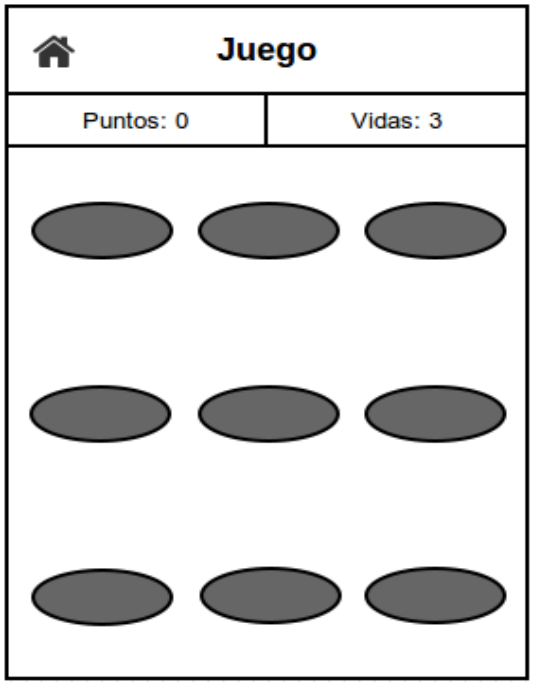
               Lanzamos menú de pausa o salir
            2. No

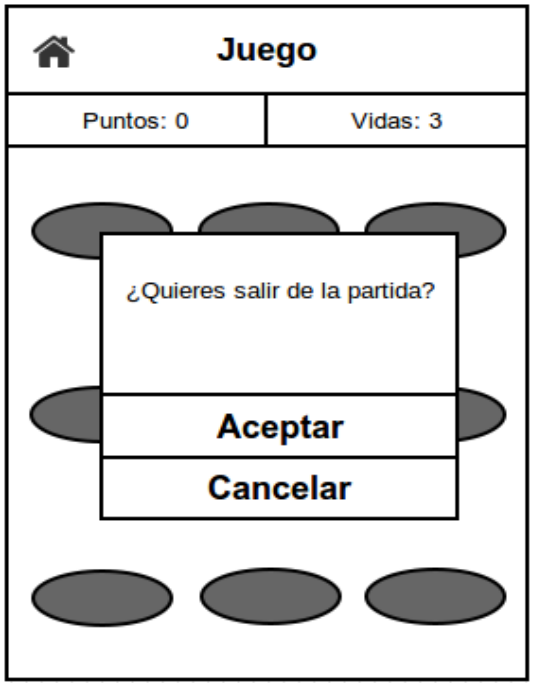
               Volvemos a la pantalla inicial

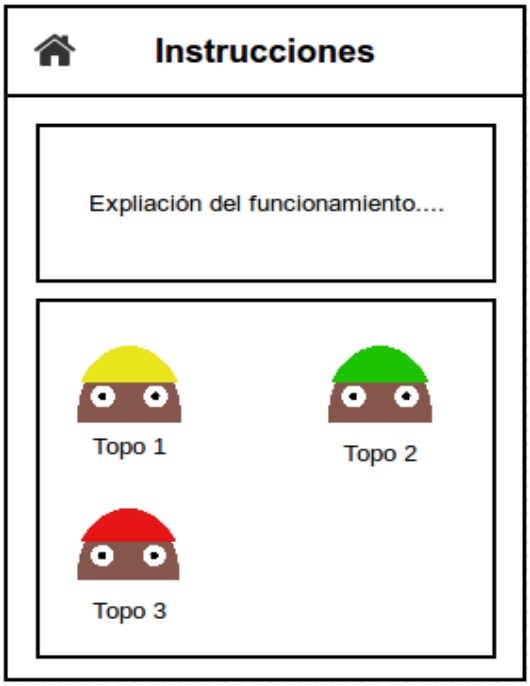
Poscondiciones: Ninguna

Interfaces gráficas:

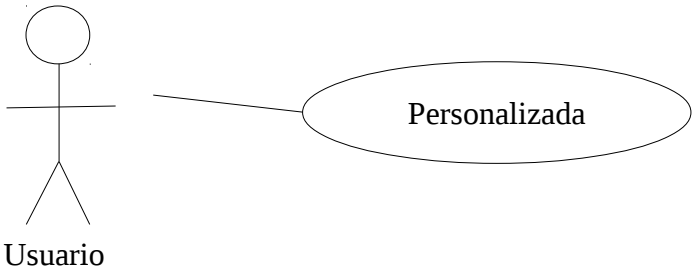








3.2.6 Personalizada



Nombre: Personalizada

Descripción: Se podrá sacar una foto con la cámara del dispositivo para que aparezca en la cara del topo la foto sacada, en vez del topo por defecto.

Actores: Usuario

Precondiciones: Disponer de cámara en el dispositivo.

Flujo de Eventos:

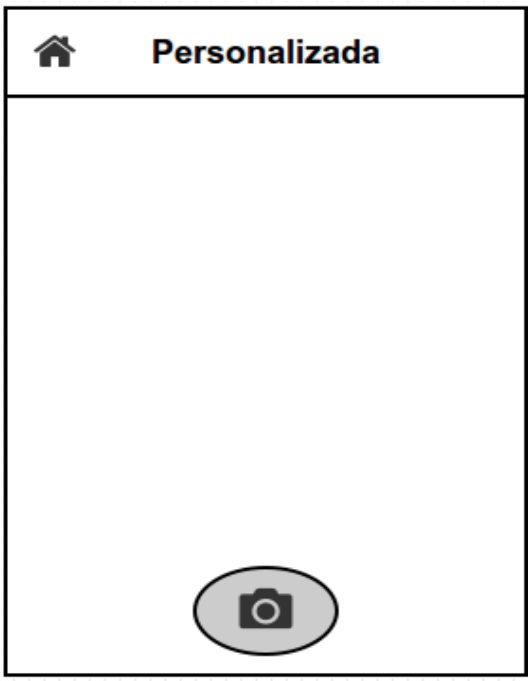
1. Pulsar el botón de “Personalizada”
2. Pulsar el botón con el símbolo de la cámara para sacar la foto.
3. ¿Quieres sacar una nueva foto?
    1. Si.

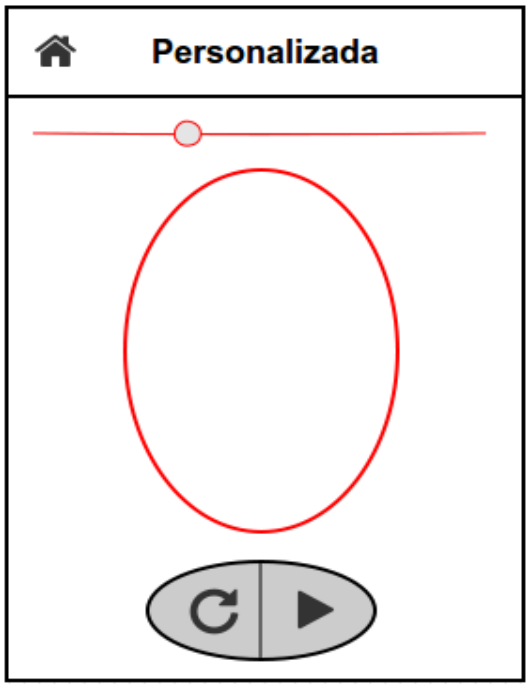
       Pulsar el botón de Nueva foto para repetir la foto.
    2. No.

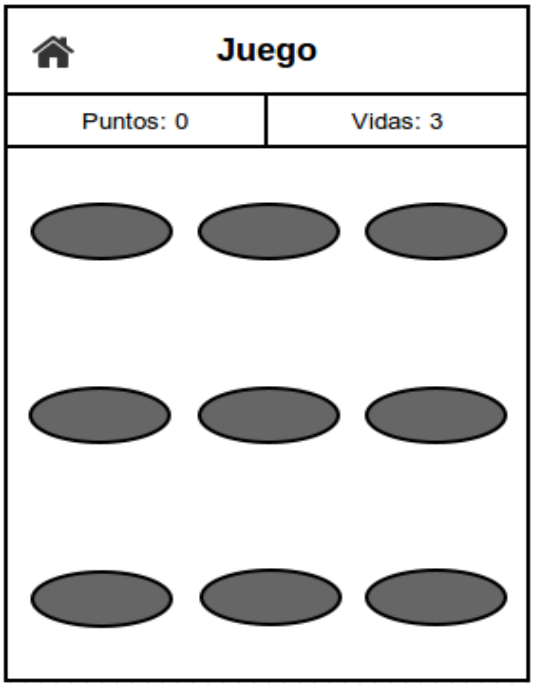
       Centrar la imagen dentro de la elipse y pulsar el botón de jugar para comenzar la partida.

Poscondiciones: Ninguna

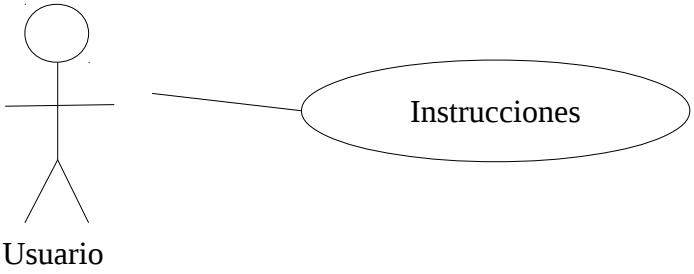
Interfaces gráficas:







3.2.7 Instrucciones



Nombre: Instrucciones

Descripción: Puede verse la forma en la que está pensado el juego para funcionar, así como las modalidades que hay.

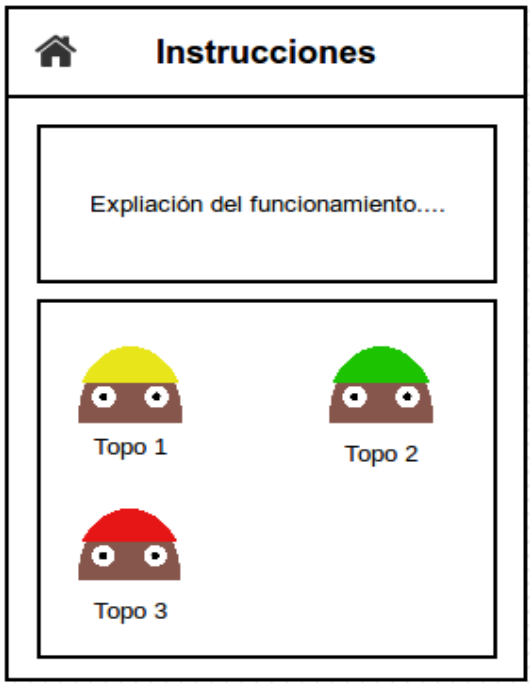
Actores: Usuario

Precondiciones: Ninguna.

Flujo de Eventos:

1. Pulsar el botón de “Instrucciones”
2. Pulsar el icono para volver atrás.

Interfaces gráficas:



4 Análisis y diseño

4.1 Modelo de dominio

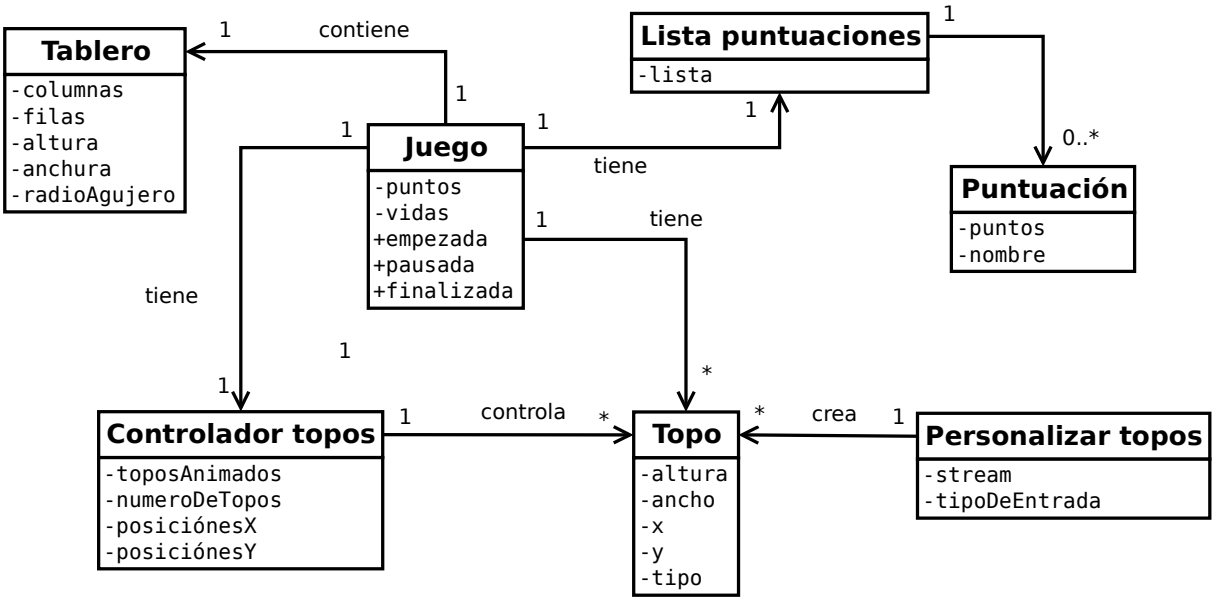


Ilustración 14: Modelo de dominio

4.2 Diagrama de clases

Aquí se representa el diagrama completo de clases, señalando las relaciones que hay entre ellas. En los apartado siguientes se detallarán cada una de las clases que lo conforman.

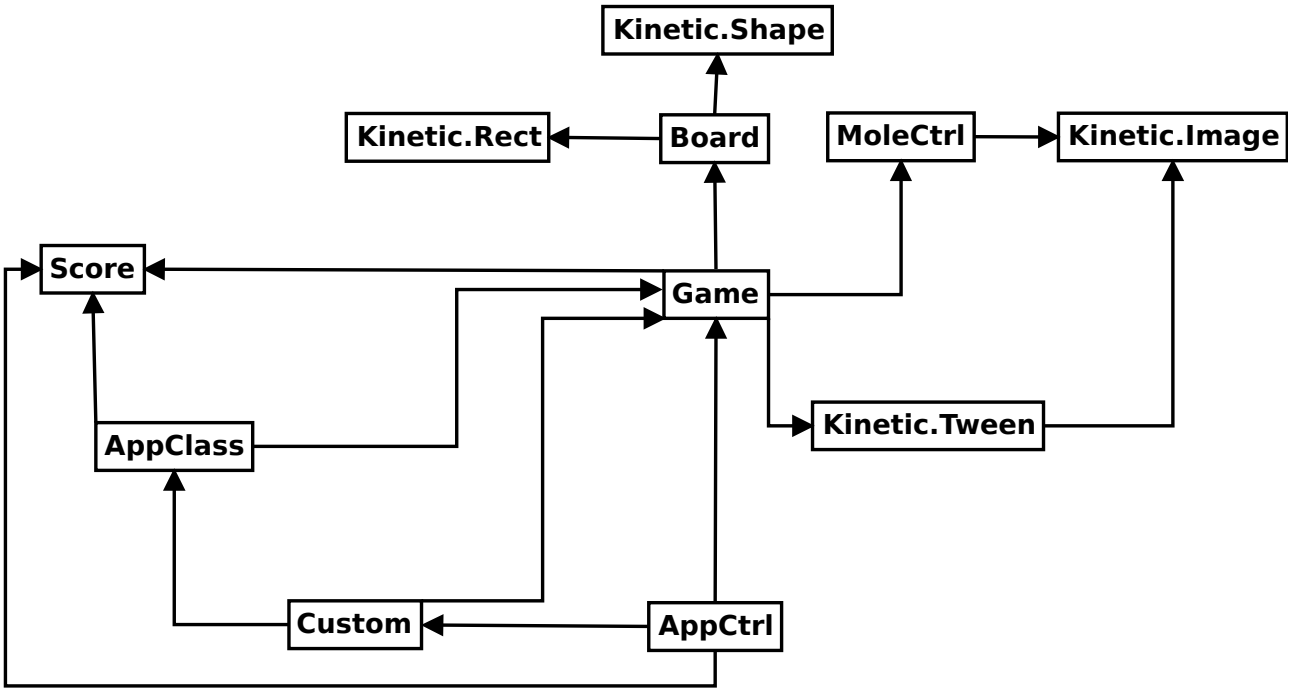


Ilustración 15: Diagrama de clases

4.2.1 AppClass

Esta clase nos sirve para obtener objetos del DOM, asi como poder mostrar mensajes de confirmación, alertas y tenemos métodos que no serán utiles a la hora de desarrollar la aplicación.

AppClass
- _window
+ dom(selector) + isArray(object) + confirm(title, message, confirm, error) + loading() + error(type, name, callback) + hide() - _removeClass(_class) - _show(_window)

4.2.2 AppCtrl

La clase AppCtrl es la que nos ayudara para pasar de una pantalla a otra y ejecutar el método de inicialización de esa pantalla si es que lo tuviera, así como de administrar los eventos de retroceso como pueden ser el botón de ir hacia atrás de un móvil como del navegador.

AppCtrl
+ transitionTime + views - actualView - backButton
+ route(id, img, args...) + onRoute(event) + onBackButton() + onPopState(event)

4.2.3 Board

Board únicamente se encargará de dibujar el tablero con sus agujeros, por los que aparecerán los topos.

Board
- columns - rows - height - width - radius - scale
+ drawBoard() + drawBaseBoard() + drawEllipse(x, y)

4.2.4 Score

En está clase encontraremos todos los métodos necesarios para controlar las puntuaciones, como guardar una puntuación o borrar todas.

Score
- key - number
+ onClear(event) + get() + isInScoreList(points) + save(points) + init() - shareMessage(points)

4.2.5 Custom

Está clase se encargará de que podamos personalizar las imágenes de nuestros topos, nos dará posibilidad de sacar una foto con la cámara del dispositivo o con API getUserMedia de HTML5 en caso de que el dispositivo sea capaz de soportarla.

Custom
- stream - sourceType
+ init() + userMedia() + onPhoto() + onRepeat() + onChange(event) + onPlay() + _toggleVideo(active) + _togglePhoto(active)

4.2.6 MoleCtrl

MoleCtrl se encargara de todo lo que tenga que ver con el control de los topos, como cual será el siguiente topo, si será bueno o malo y se encargara de dibujarlo.

MoleCtrl
- animatedMoles - numMoles - width - radius - columns - rows
+ drawMole(x, y, id, height, image, good) + nextMole() + removeFromAnimated(id) + calculatePositions()

4.2.7 Game

Esta clase se encargará de ejecutar el proceso de jugar en si, se encargará de llamar a las correspondiente clases para dibujar el tablero, los topos y también se encargará de

Game
- points - lives - posX - posY + columns + rows + started + paused + finished
+ init(img, reset) + loadImages(img, callback) + reset() + onStart(event) + onClickMole() + changeVelocity() + play() + startInterval() + onFinishTween() + pause() + finish() + destroyTweens()

4.2.8 Kinetic

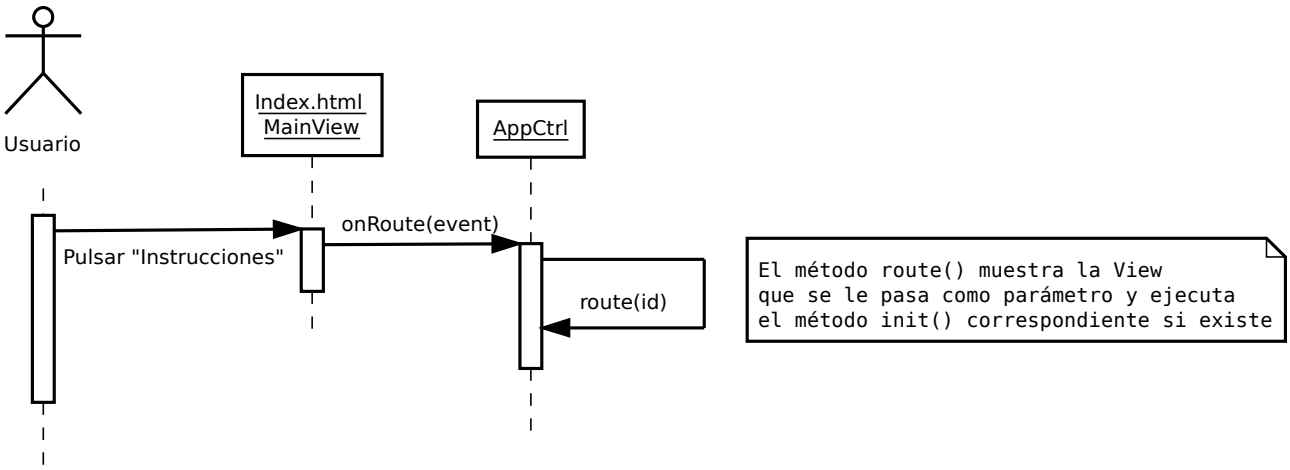
Las que empiezan por Kinetic pertenecen a la librería Kineticjs que se esta utilizando para disminuir la dificultad a la hora de dibujar sobre *canvas*. Para encontrar más información sobre las clases que conforman esta librería se pueden encontrar aquí:

- Kinetic.Tween: http://agavestorm.com/kineticjs/Kinetic.Tween.html
- Kinetic.Image: http://agavestorm.com/kineticjs/Kinetic.Image.html
- Kinetic.Shape: http://agavestorm.com/kineticjs/Kinetic.Shape.html
- Kinetic.Rect: http://agavestorm.com/kineticjs/Kinetic.Rect.html

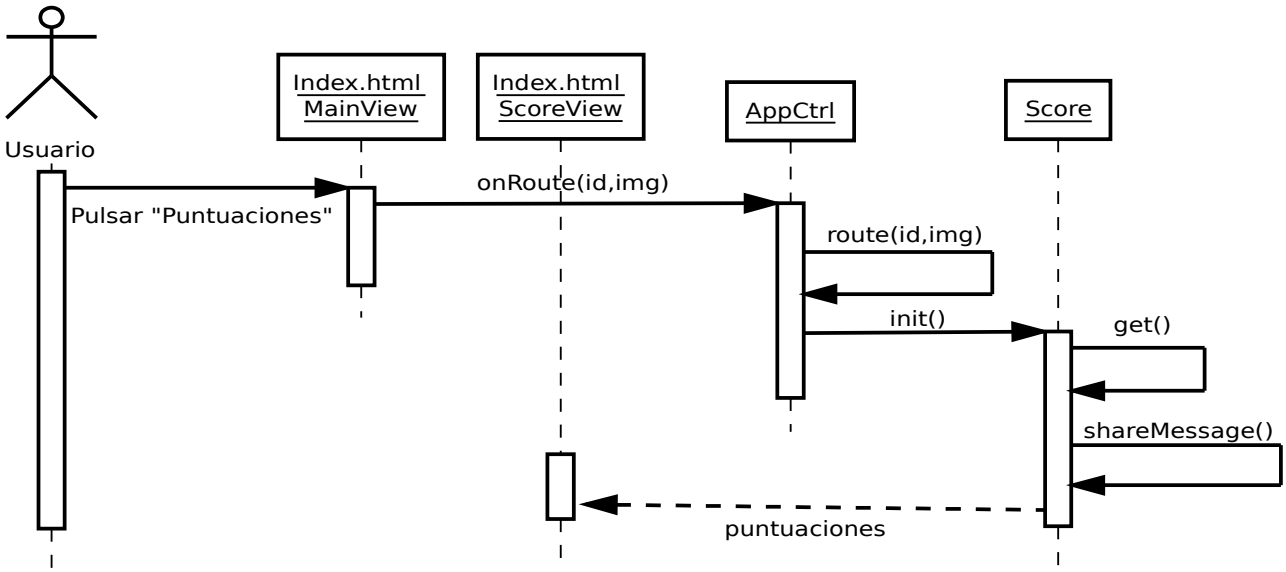
5 Diagramas de secuencia

Se mostrarán todos los diagramas de secuencia de las diferentes funcionalidades de las que consta toda la aplicación.

5.1 Ver instrucciones

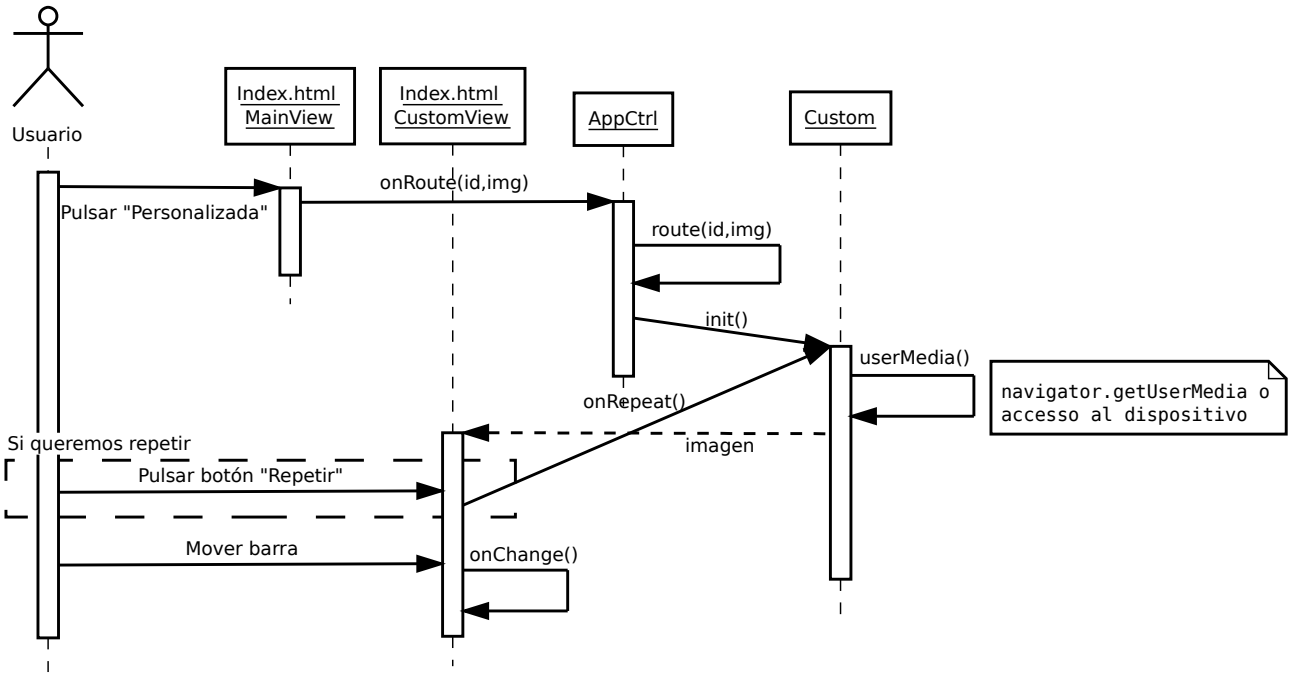


5.2 Listar puntuaciones



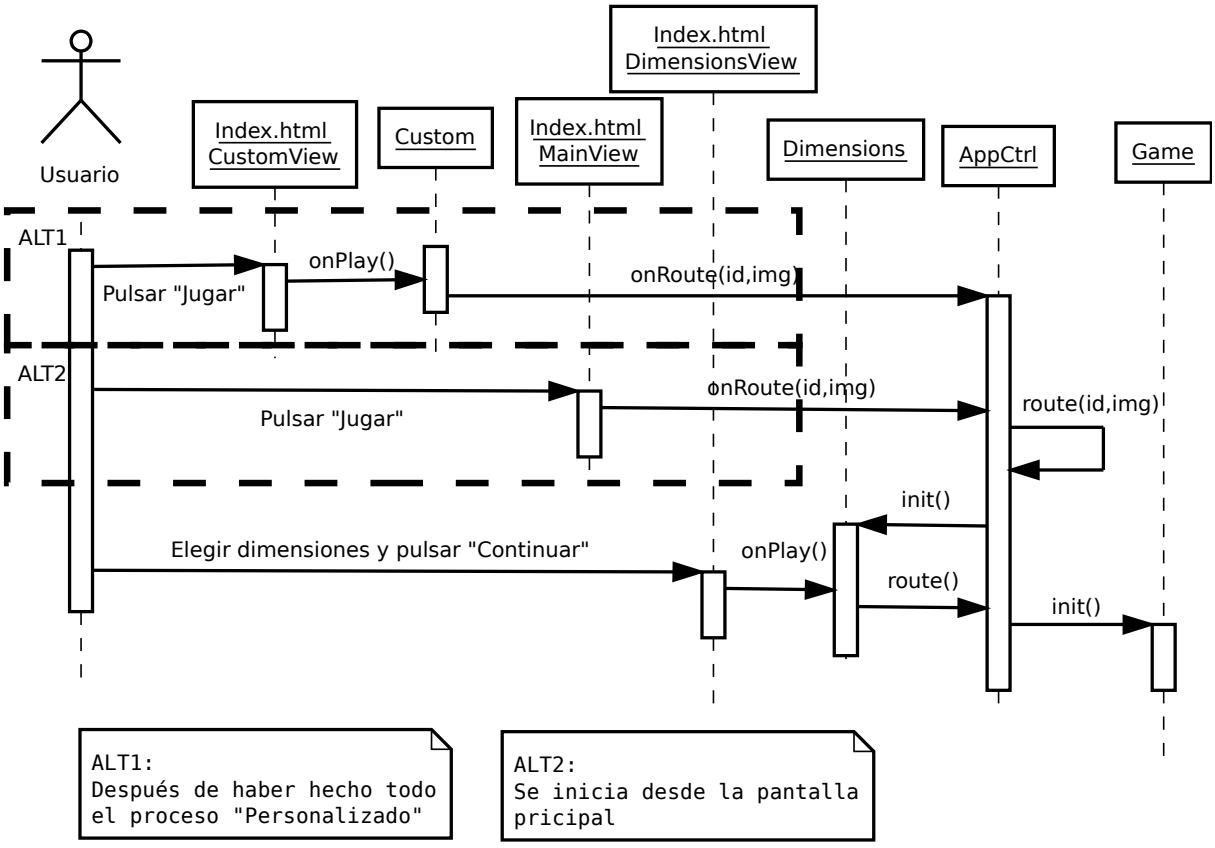
5.3 Preparar topos personalizados (Custom)

En este diagrama aparece el proceso que se sigue para poder poner una imagen sacada con la cámara del dispositivo como si fuera el topo. Podremos ajustar la imagen agrandarla o moverla para que encaje perfectamente.



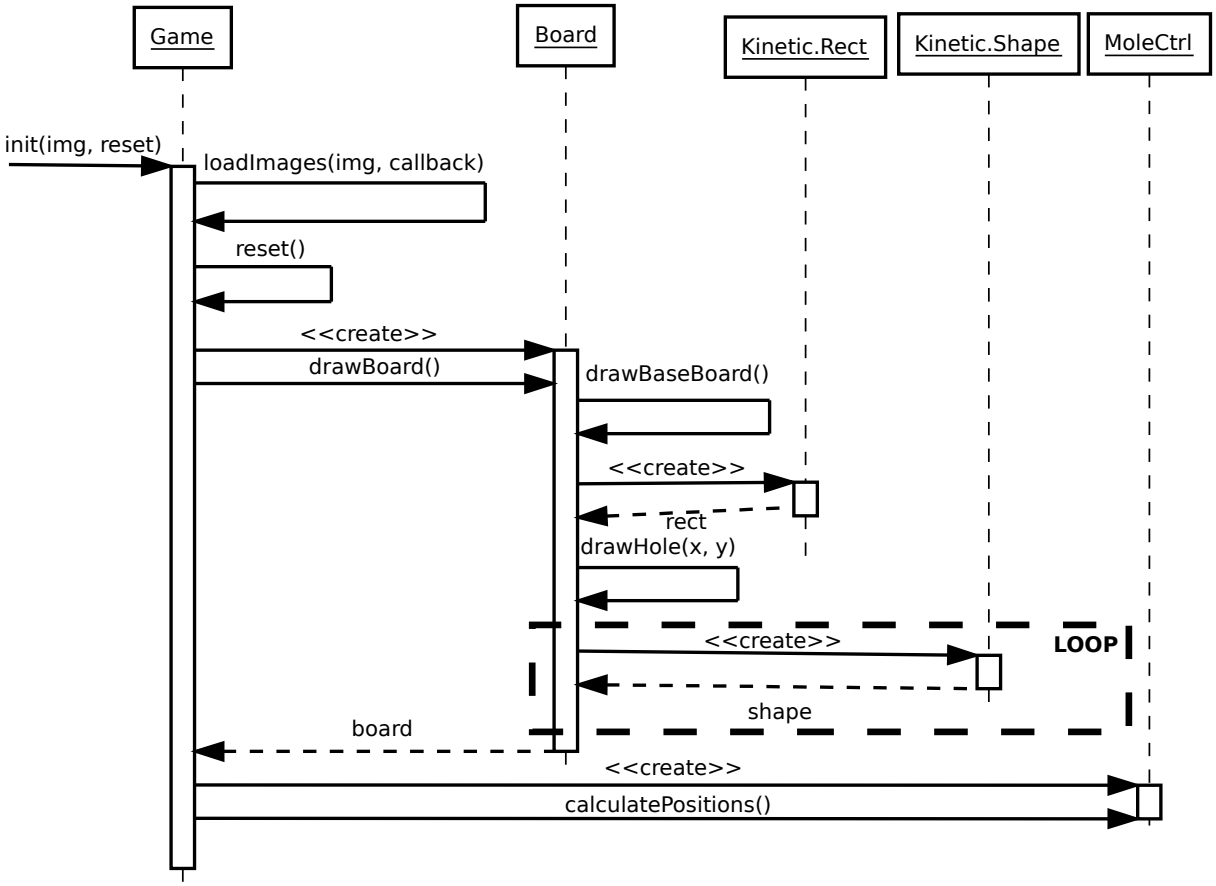
5.4 Ir a pantalla de juego

Vemos el proceso que se sigue cuando pulsamos el botón de jugar, bien pulsando el botón de jugar de la pantalla principal o pulsando el botón de jugar que aparece después de realizar el proceso de personalización de los topos.

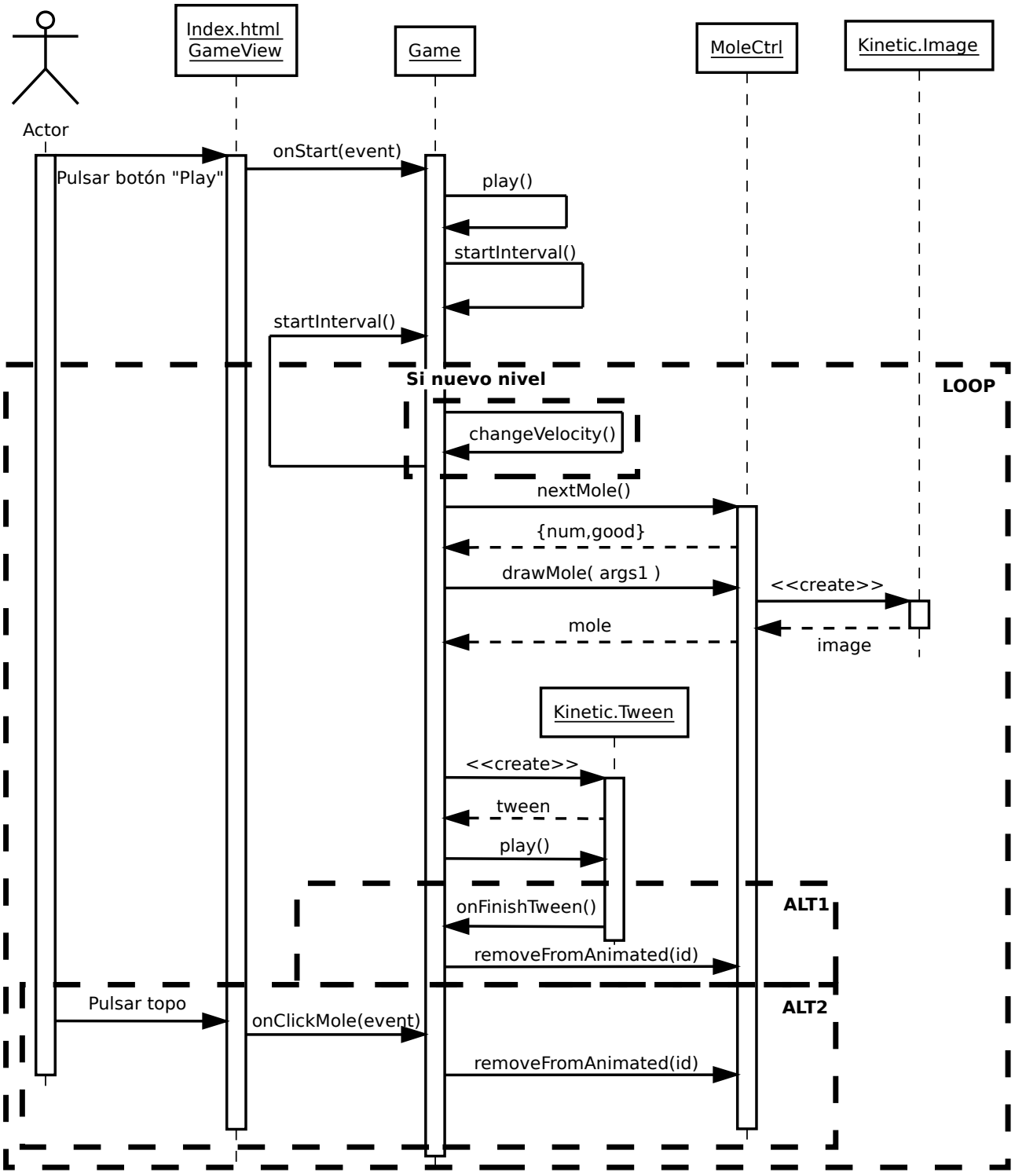


5.5 Iniciar pantalla de juego

Este diagrama es la continuación del siguiente, son los métodos que se ejecutan para crear la pantalla donde aparecerán los topos, se cargan las imágenes las personalizadas o por defecto, se dibuja el tablero, se crea la clase que dibujará los topos y se calculan posiciones donde aparecerán.



5.6 Jugar


Actor
Index.html
GameView
Game
MoleCtrl
Kinetic.Image
Pulsar botón "Play"
onStart(event)
play()
startInterval()
startInterval()
LOOP
Si nuevo nivel
changeVelocity()
nextMole()
{num,good}
drawMole(args1)
<<create>>
mole
image
Kinetic.Tween
<<create>>
tween
play()
ALT1
onFinishTween()
removeFromAnimated(id)
ALT2
Pulsar topo
onClickMole(event)
removeFromAnimated(id)

Este es el diagrama que explica todo el proceso de jugar y para que quede más claro es mejor explicar algunas partes del diagrama.

En el diagrama nos encontramos la alternativa "*si nuevo nivel"*, los niveles van determinados por la cantidad de puntos, entonces esta alternativa quiere decir que si hemos llegado a una cantidad de puntos determinada por un atributo de la clase, se aumentará la velocidad.

El método *nextMole*, devuelve un objeto que nos indica la posición en la que aparecerá el nuevo topo y que tipo es.

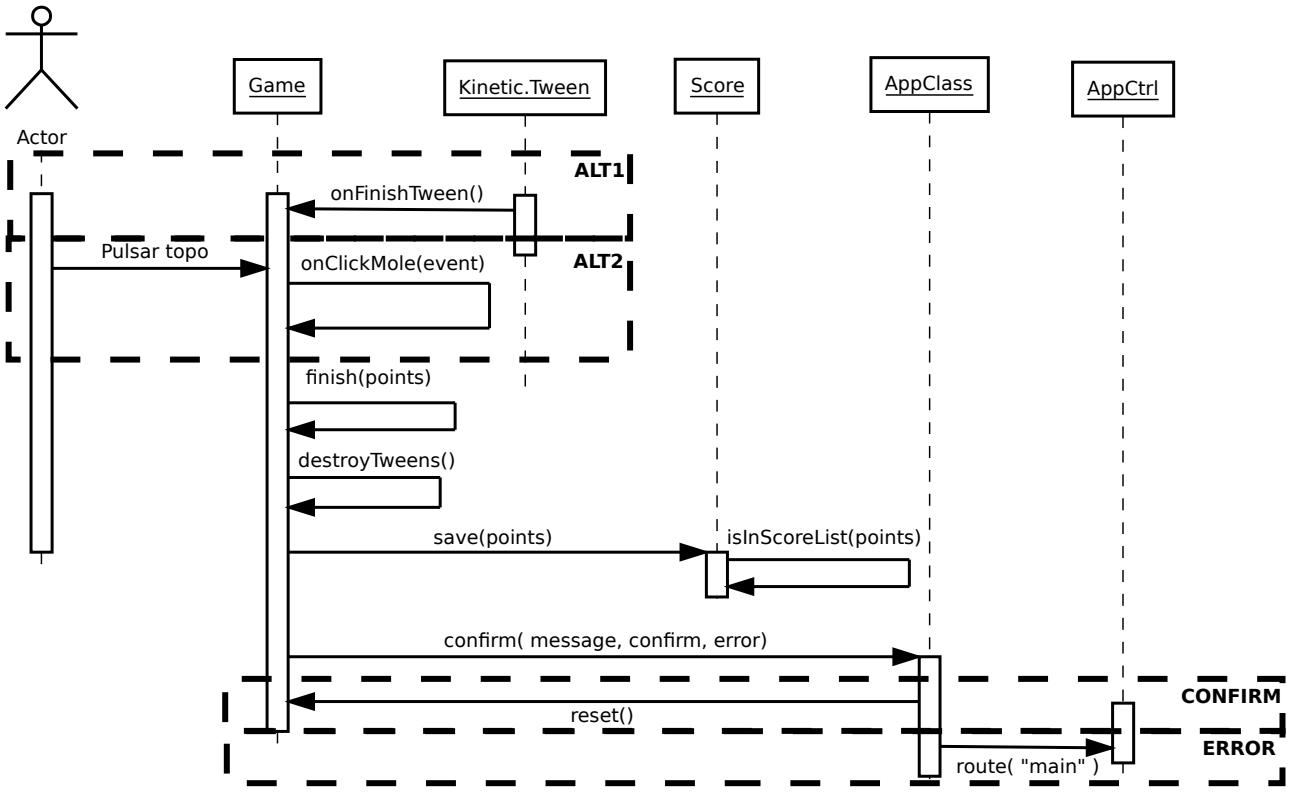
ALT1. Se ejecuta cuando no pulsamos un topo

ALT2. Se ejecuta cuando hemos pulsado un método.

ALT1 y ALT2 serán los métodos encargados de dar puntos, quitar vidas o terminar el juego dependiendo de lo que hayamos hecho en el juego.

5.7 Fin de la partida

En diagrama es la continuación del diagrama anterior, el proceso describe los métodos que conllevarían a acabar la partida.

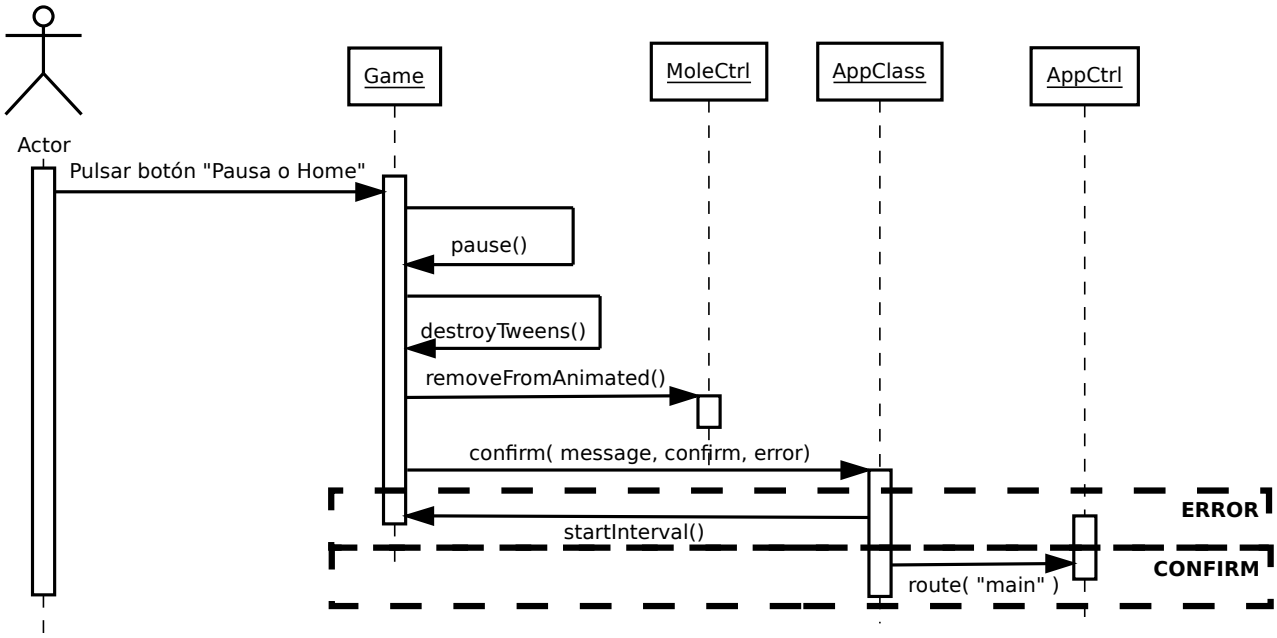


Imaginemos que sólo disponemos de una vida para continuar y por alguna razón ya se por tocar el topo inadecuado (ALT2) o por no pulsar al que debiéramos (ALT1) el juego guardaría los puntos conseguidos en la partida y nos aparecería un mensaje para volver a jugar o no.

En caso de decir si se ejecutaría *reset* y volveríamos al diagrama del punto **5.5**, en caso contrario volveríamos a la pantalla inicial.

5.8 Pausar o salir de la partida

Cuando pulsamos el botón de salir de la partida nos saldrá un mensaje para confirmar si queremos salir, si es error se reanudará el juego y en caso afirmativo, saldremos a la pantalla inicial.



6 Desarrollo

En este apartado se mostrará el proceso que se ha llevado acabo durante la implementación. El proceso consiste en irse planteando pequeños objetivos que nos permitan ir avanzando progresivamente y con mayor facilidad. Para explicar este apartado seguiremos el mismo método explicando poco a poco las partes que se han ido desarrollando.

El proyecto esta desarrollado utilizando la combinación de los 3 lenguajes básicos para el desarrollo web que son HTML, CSS y JavaScript. También se ha utilizado la librería Kineticjs, que nos ayudará al dibujo en 2D que nos aporta el elemento *canvas* de HTML5, y también utilizaremos Apache Cordova para construir aplicaciones nativas usando los lenguajes mencionados anteriormente.

En el desarrollo en vez de utilizar los lenguajes mencionados, se utilizará una variante de ellos que son Jade, Stylus y CoffeeScript como ya se ha mencionado en el apartado **1.5.3**

6.1 Interfaz de usuario e interacción de pantallas

Para la interfaz de usuario, como en toda página aplicación web, se utilizará HTML y CSS, HTML nos servirá para crear los elementos de los que dispondrá la aplicación y CSS nos aportará el estilo que le daremos a cada una de los elementos, podemos personalizar casi totalmente cada uno de nuestros elementos a nuestro gusto.

6.1.1 Responsive Design

Para el desarrollo de la aplicación se ha seguido el model de "*Responsive Desgin"* que sirve para definir aquellas páginas web que se adaptan a los distintos tipos de pantalla y diferentes dispositivos, sin tener que diseñar diferentes aplicaciones para cada uno.

6.1.1.1 Cajas flexibles

Para este diseño se ha utilizado la cajas flexibles que nos aporta CSS3 y que nos permite distribuir los elementos que están contenidos dentro de otro, pudiendo tener dimensiones flexibles para adaptarse al tamaño de la pantalla. Así, tendremos otras propiedades que podremos utilizar en cada uno de lo elementos contenidos para que se comporten de la manera que deseemos.

Existen distintas versiones y hemos utilizado una versión de 2009 y la que está actualmente para tener un mayor soporte en los distintos navegadores, siempre intentando utilizar la versión

Version 2009: *display: box*

Versión actual: *display: flex*

Todas las pantallas de la aplicación siguen este esquema creado por cajas flexibles.

```
body > section {
  display: box;
  box-orient: vertical;
  display: flex;
  flex-direction: column;

body > section > header,
body > section > footer {
  box-flex: 0;
  flex: 0 1 auto;
}
body > section > header {
  box-ordinal-group: 1;
  order: 1;
}
```

```
body > section > nav {
  box-ordinal-group: 2;
  order: 2;
}
body > section > article {
  box-flex: 1;
  box-ordinal-group: 3;
  flex: 1;
  order: 3;
}
body > section > footer {
  box-ordinal-group: 4;
  order: 4;
}
body > section > header,
body > section > footer {
  height: 3em;
  line-height: 3em;
}
```

Todas las *sections* dispondrán de cajas flexibles y se distribuirán en forma de columna donde el *header, nav, y footer* al tener la propiedad *flex: 0 1 auto* o no disponer de la propiedad no serán flexibles por eso podremos establecerles una altura fija. La única propiedad flexile será el elemento *article*.

Con la propiedad *order y box-ordinal-group* especificamos el orden en el que se dispondrán los elementos independientemente del orden en el código fuente.

6.1.1.2 Media Queries

Otra de las novedades que se utilizan de CSS3 en el proyecto son las *media queries* que son una consulta al medio, es decir, nos permite poner condiciones sobre el ancho, orientación y algunas opciones más sobre el medio en el que se está mostrando la aplicación y si esa condición se cumple el navegador utilizará los estilos definidos dentro de esa condición.

```
@media screen and (max-width:400px)
  .hide-mobile
    display none


@media screen and (max-width:400px)
  #main
    #title
      font-size 7em
    .img
      width 3em
      height 5em


@media screen and (max-width:350px)
  #main
    #title
      font-size 5em
```

En la aplicación únicamente se utilizan 3 *media queries*, los cambios que se hacen son cuando la pantalla es menor de 400px o 350px. No se muestran los elementos que tengan las clase *.hide-mobile* cuando la pantalla es menor de 400px y se cambia los tamaños del titulo y de los elementos que contengan la clase *.img* en los distintos tamaños de pantalla.

6.1.2 Pantallas de la aplicación

En este apartado se explica como se han desarrollado las diferentes pantallas de la aplicación y la navegación entre ellas.

6.1.2.1 Pantallas

Todas las pantallas de la aplicación han sido implementadas en mismo archivo HTML, index.html, cada una de las pantallas de la aplicación esta representada por un elemento *section* y cada una de ellas se mostrará o se ocultará según vayamos navegando de una a otra. Que se muestren o no dependerá de la clase *.show* que la pantalla sea visible o no visible si no tienen como atributo esa clase.

Cada *section* o pantalla deberá disponer de un identificador que será necesario para poder navegar entre pantallas.

6.1.2.2 Direccionamiento de pantallas

Como se ha comentado en el punto anterior cada pantalla deberá disponer de un identificador único para poder navegar entre ellas.

```
<section id="main" transition="pop" class="show">
	<button data-view="score" class="max-width">Puntuación</button>
</section>

<section id="score" transition="pop">
</section>
```

Estas dos sections son parte del código de la aplicación y servirá para explicar el funcionamiento de navegación entre pantallas. Como se puede ver cada una tiene un identificador único y la pantalla activa es la del identificador “main” porque contiene la clase *.show* como se ha explicado antes.

Otra de las cosas que se debe tener en cuenta es el atributo *data-view* que contiene el botón y su valor es el identificador de la otra pantalla, esto quiere decir que si se pulsa este botón navegaremos hacia la pantalla con el identificador “score”.

¿Quién se encarga de cambiar de pantalla? La clase AppCtrl, la cual se encarga de toda la navegación de la aplicación.

Los métodos de la clase AppCtrl que se encargan de la navegación son el método *onRoute(event)* y *route(id)*

Cuando se inicia la aplicación y se crea la clase AppCtrl el constructor de la clase será el encargado de añadir eventos a todos los elementos que contengan el atributo *data-view*.

```
sections = App.dom("[data-view]");
for (i = 0, len = sections.length; i < len; i++) {
  e = sections[i];
  e.addEventListener("click", this.onRoute, false);
}
```

Teniendo todos los eventos creados cuando pulsamos sobre un elemento que contiene el atributo se ejecuta la función de la clase *onRoute* la cual se encarga de obtener el valor del atributo *data-view,* cuyo valor será el *id* de la pantalla a la que navegaremos, y de llamar al método *route* mandándole ese valor.

```
AppCtrl.prototype.onRoute = function(event) {
   var element, id;
   element = event.currentTarget;
   id = element.attributes["data-view"].value;
   return this.route(id);
};
```

Una vez se llamado el método *route* este se encarga de hacer una consulta al DOM para buscar la pantalla que tiene la clase *show* y quitársela para que no se muestre y ñadirsela a la que queremos mostrársela.

```
  showed = App.dom(".show")[0];
  showed.setAttribute("direction", "out");
  setTimeout(function() {
    return showed.classList.remove("show");
  }, this.transitionTime);
  toShow = App.dom("#" + id)[0];
  toShow.setAttribute("direction", "in");
  return toShow.classList.add("show");
```

Como podemos ver en el código también se cambia el atributo “direction”. Este atributo simplemente nos sirve para asignar una animación cuando se cambia entre pantallas. Esta animación se crea a través de la nueva especificación CSS3.

Con la propiedad @keyframes de CSS3 podemos crear las animaciones y después se puede asignar esta animación añadiendo a un elemento el atributo *animation* y luego el nombre de la animación creada seguida de las demás propiedades que afectan a la animación.

```
@keyframes popIn {
  from {
    transform: scale(0);
  }
  to {
    transform: scale(1);
  }
}
```

```
@keyframes popOut {
  from {
    transform: scale(1);
  }
  to {
    transform: scale(0);
    opacity: 0.5;
  }
}
```

```
section[transition="pop"][direction="in"] {
  animation: popIn 500ms cubic-bezier(0.65, 0.05, 0.36, 1);
  z-index: 2;
}
section[transition="pop"][direction="out"] {
  animation: popOut 500ms cubic-bezier(0.65, 0.05, 0.36, 1);
  z-index: 1;
```

6.1.2.3 Retroceso

En este apartado se explicará como se ha gestionado la navegación entre pantallas cuando se pulsa el botón de retroceso tanto del navegador como el de un dispositivo móvil.

Para manejar el botón de un dispositivo móvil ha sido fácil controlar la navegación ya que Cordova un evento para detectar cuando se pulsa este botón, para ello hemos añadido el evento al documento y

```
document.addEventListener "backbutton", @onBackButton
```

Cuando detecte este evento se lanzará el método *onBackButton* que redirigirá a la página principal, en caso de que estemos en la pantalla de la partida pausará el juego, que nos da la opción de salir a la pantalla principal y si ya se encuentra en la pantalla inicial se cerrará la aplicación.

```
onBackButton: (event) =>
    backbutton = true
    event.stopPropagation()
    if actualView is "main"
      navigator.app.exitApp()
    else
      if actualView is "game"
        App.game.pause()
      else
        @route "main"
```

Para el caso de que nos encontremos en un navegador podemos manipular el historial añadiendo entradas. Cuando añadimos la entrada podemos añadirle un estado que podremos consultar cuando se lanza el evento *popstate* que significa se está utilizando el historial para navegar en la aplicación en vez de la aplicación en sí.

Cuando iniciamos la aplicación creamos la primera entrada en el historial.

```
window.history.pushState "back", null, null
```

El parametro "*back*" sería el estado que le asignamos a esa entrada del historial para comprobar que queremos hacer cuando detectemos esa entrada. Las otras dos entradas sería el título y la nueva ruta que queremos que se cargue pero en este caso no nos son necesarios.

Cuando pulsamos el botón hacía atrás se lanza el evento *popState* y con ello el método *onPopState* asociado a ese evento.

```
window.onpopstate = @onPopState

onPopState: (event) =>
    if event.state is "back"
      window.history.pushState "back", null, null
      if actualView is "game"
        App.game.pause()
      else
        if actualView isnt "main"
          @route "main"
```

La única diferencia con respecto al evento de un botón del dispositivo es que en este método al ir hacia atrás desaparece esa entrada en el historial, con lo que debemos crear una otra vez para la próxima que se pulse hacia atrás.

6.2 Interfaz e interacción de la partida

Para la interfaz de la partida donde utilizaremos el elemento *canvas* para dibujar, se ha elegido utilizar la librería Kineticjs por la facilidad que nos da al crear los distintos objetos y la posibilidad de poderles añadir eventos a cada uno de ellos, tanto para pantallas táctiles como para eventos del ratón en un PC.

6.2.1 Contenedor principal

Para utilizar Kinetic necesitamos establecer un contenedor principal para contener todos los demás objetos que creemos con esta librería.

Para establecer las dimensiones que queramos debemos definir las medidas mediante JavaScript no podemos ayudarnos de medidas relativas como los porcentajes para que se ajuste a cada pantalla.

Como queremos la pantalla de juego este ajustada al elemento *article* que se encuentra en esa pantalla y para ello obtenemos las medidas del elemento a través de JavaScript.

```
this.height = App.dom("#game-zone")[0].clientHeight;
this.width = App.dom("#game-zone")[0].clientWidth;
```

Una vez que tenemos las medidas donde queremos encajar nuestra pantalla de juego creamos creamos con Kinetic el contenedor que abarcará las distintas capas.

```
mainStage = new Kinetic.Stage({
    container: "board",
    height: this.height,
    width: this.width
});
```

Este será el contenedor que contenga todas las etiquetas *canvas* en la que vamos a dibujar. Podemos ver que el elemento que contendrá a éste es el que tiene como identificador "*board*", que no es el mismo del que hemos cogido las medidas, la razón de que esto sea así es que al asignar al *Stage* un contenedor elimina todos los hijos de éste e incluye el creado por Kineticjs. El elemento *board* si tendrá el mismo tamaño que *game-zone* pero éste tendrá más elementos hijos que no deben desaparecer.

6.2.2 Lógica de tablero

En este apartado se explicará cual ha sido la implementación que se ha llevado para crear el tablero con los agujeros por los que aparecerán los diferentes topos.

6.2.2.1 Tablero base

Para crear el tablero primero crearemos la base y después le añadiremos los agujeros. Para la base utilizaremos la clase *Kinetic.Rect* que nos ayudará a dibujar un rectangulo base sobre los que dibujar los agujeros.

```
boardLayer = new Kinetic.FastLayer;
board = this.drawBaseBoard();
boardLayer.add(board);

Board.prototype.drawBaseBoard = function() {
  var board;
  return board = new Kinetic.Rect({
    height: this.height,
    width: this.width,
    fill: "#DEB887"
  });
};
```

Primero debemos crear un *Layer* que son capas que luego añadiremos al contenedor principal para que se vean. Luego creamos el rectangulo con las medidas que consideremos para que ocupen todo el contenedor principal y le añadimos un color. Una vez creado este rectangulo se lo añadiremos al *Layer. FastLayer* es un tipo de capa al que no se le pueden añadir eventos y se dibuja más facilmente.

6.2.2.2 Dibujar agujeros

Para dibujar los agujeros debemos tener más medidas en cuenta, como el número de filas y columnas que vamos a tener, el radio que utilizaremos para dibujar los agujeros, así como el espacio que queremos que haya entre agujeros.

```
columns = App.game.columns
rows    = App.game.rows
radius  = (@width/columns)/4
scale   =
  x: 1.5
  y: 0.25


drawBoard: ->
  ...
  for x in [radius*2..width] by width/columns
    for y in [height/rows - radius..height] by height/rows
      hole = @drawHole x, y
      boardLayer.add hole

    boardLayer
```

El código anterior está escrito en CoffeeScript en vez de JavaScript por que considero que va a ser más explicativo para este caso.

Para calcular las posiciones donde se colocarán los primero nos basaremos en el número de columnas y de filas de las que constará el tablero.

En nuestro caso el tablero será de 3 columnas por 3 filas, pero no se verán las lineas simplemente se seguirá el esquema para explicar el método que se ha seguido para situar todos los agujeros.

El método que calcula las posiciones el método *drawBoard* en el *for*. Cada uno de los agujeros tendrá su centro en la mitad de cada una de las celdas que están con borde más negro y el radio de las circunferencia será la distancia de uno de los cuadrados, con lo que la circunferencia ocupará la distancia de 2 de esos cuadrados.

Entonces la distancia en la que se encuentra el centro del primer agujero se encuentra en *radius*2* y haremos que en cada vuelta la distancia aumente una distancia equivalente al ancho de la columna *width/columns* y así obtendremos todas las posiciones en x para que coincidan con el centro de cada una de las casillas.

```
for x in [radius*2..width] by width/columns
```

Para calcular las posiciones en *y* queremos que el agujero quede en la parte baja de cada una de las celdas y por ello empezamos en *height/rows*, la distancia total de la altura entre el número de filas, y en cada vuelta aumentamos la posición *y* en es misma distancia. Al calcular estas distancias de esta manera los agujeros que se encuentran en la última fila sólo tienen visible la parte superior del circulo, para solucionarlo reducimos esta distancia en la misma que el radio y así son completamente visibles.

```
for y in [height/rows - radius..height] by height/rows
```

Una vez que vamos calculando le enviamos las pociones al método *drawHole(x, y)* el cual es el encargado de dibujar los agujeros.

```
drawHole : (x, y) ->
  hole = new Kinetic.Shape
    x: x
    y: y
    drawFunc: (ctx) ->
      ctx = ctx._context
      ctx.beginPath()
      ctx.scale scale.x, scale.y
      ctx.arc 0, 0, radius, 0, Math.PI * 2, false
      ctx.closePath()
      ctx.fill()
      ctx.restore()
```

El método creará el objeto que representa el agujero que se va a dibujar. Este objeto lo deberemos añadir al *FastLayer* y este después lo añadiremos al contenedor principal *Stage* que será cuando se dibuje.

Como podemos ver se ha cambiado la escala a la hora de dibujar el agujero, la razón es para que se muestre como una elipse en vez de como un circulo. Se elige escalar “x” a 1.5 para que al final acabe ocupando el espacio equivalente a la distancia de 3 de los cuadros mostrados en la table de referencia, y escalamos “y” a 0.25 por una elección de estética.

6.2.3 Dibujar topos

En esta sección explicaremos como se ha implementado el tamaño que van a tener los topos a la hora de dibujarlos.

Para calcular el ancho que tienen que tener los topos para que aparezcan encima de los agujeros simplemente se ha calculado el doble del radio del agujero *width = radius * 2*. Para calcular la altura que debían tener sin que estos tapasen los agujeros que tienen por encima o los topos superiores sobresalieran del limite del tablero, se calcula que ocupasen la mitad del espacio que hay entre las distintas filas de agujeros.

```
height = App.game.height / (2 * rows)
```

En nuestro ejemplo que serían 3 filas cada topo ocuparía 1/6 de la altura total, en caso de que hubiera más filas mediante la anterior operación se calcularía automáticamente.

El método que crearía el objeto que más tarde se añadira al *Layer* y este después se añadirá al contenedor principal para que se dibuje el topo sería el siguiente. Los atributos que contienen y el valor de cada uno se explicarán en los siguiente apartados.

```
drawMole: (num, image, good) ->
    mole = new Kinetic.Image
      id: num
      image: image
      x: posX[num] - radius
      y: posY[num] - radius
      height: height
      width: width
      good : good
      scaleY: 0.01
      shadowColor: "black"
```

6.2.4 Calcular posiciones de los topos

Cuando se inicia el juego se llama a la clase *MoleCtrl* y después al método *calculatePositions* el cual el es encargado de establecer las posiciones en las que se dibujarán los topos. El sistema es igual al que seguimos cuando se quiere dibujar los agujeros pero con una diferencia, los topos deben ser dibujados encima de los agujeros.

Para calcular la posición en "*y*" en la que deben aparecer simplemente debemos restar la altura calculada del topo a "*y*".

```
for x in [radius*2..widthBoard] by widthBoard/columns

  for y in [heightBoard / rows – moleHeight .. heightBoard] by heightBoard / rows

    posX.push x

    posY.push y
```

Como hemos visto en el método que dibuja los topos simplemente le pasamos como primer parámetro un número, que indica el número del agujero del que debe aparecer, y con este número podremos asignar las coordenadas cogiendo los valores que tengan en esa posición posX y posY

```
x: posX[num] - radius
y: posY[num] - radius
```

La razón para que le restemos el radio a la posición es que ha diferencia del circulo que el punto origen es el centro de la forma, la de una imagen empieza en la esquina superior izquierda.

6.2.5 Inicio de la partida

Lo primero que hacemos al iniciar una partida es cargar las imágenes que van a representar a los topos. Si el método *init* de la clase *Game* recibe imágenes personalizadas se cargarán esas imágenes y en caso contrario, las imágenes por defecto.

```
loadImages: (img, callback) ->
  App.loading()
  unless img?
    img = ["static/images/mole.png"
      "static/images/mole-good.png"
      "static/images/mole-bad.png"]

  @moleNorm = new Image
  @moleNorm.onload = =>
    @moleGood = new Image
    @moleGood.onload = =>
      @moleBad = new Image
      @moleBad.onload = ->
        App.hide()
        callback?()
      @moleBad.onerror = _error
      @moleBad.src = img[2]
    @moleGood.onerror = _error
    @moleGood.src = img[1]
  @moleNorm.onerror = _error
  @moleNorm.src = img[0]
```

Y después se resetean distintos valores como las vidas, los puntos, los niveles y el intervalo por el que van apareciendo.

```
reset: =>
    lives  = 3
    points = 0
    @intervalTime = 1000
    @levels = [10, 20, 40, 50, 60, 80, 100, 120]
```

6.2.6 Mejorar animaciones

Con el método anterior se había calculado la posición en la que aparecían los topos pero tenía un problema, no era atractivo visualmente que los topos aparecieran de repente encima del agujero, había que darle la sensación para que pareciera que los topos surgían de los agujeros. Para ello Kineticjs nos ofrece la clase *Kinetic.Tween.*

Con esta clase se puede asignar una animación a los distintos atributos de un elemento, que en este caso serán los atributos *scaleY* y *height* del topo.

Para conseguir este efecto lo que se ha hecho es escalar la altura del topo a 0 y bajar la altura a la que aparece el topo a la misma en la que dibujamos los agujeros con lo que el método para calcular esta posición quedaría finalmente asi:

```
for x in [radius*2..widthBoard] by widthBoard/columns
  for y in [heightBoard / rows .. heightBoard] by heightBoard / rows
    posX.push x
    posY.push y
```

Como se ha visto el atributo *scaleY* tiene valor 0.01 en vez de 0. La razón de que esto sea así es que Kinetic tiene un *"bug"* que en ciertos navegadores cuando se quiere asignar una animación a un atributo del objeto y este empieza desde 0 falla.

Para asignar animación a los atributos de un elemento, se asignan a través de la clase *Kinetic.Tween* y en este caso quedaría de esta manera.

```
scale = new Kinetic.Tween
        node: mole
        duration: transitionTime
        scaleY: 1
        y: mole.getY() - moleHeight
```

Asignamos al atributo node el elemento que queremos animar en este caso es *mole,* el objeto que representa al topo, luego especificamos en el atributo "*duration*" el tiempo que queramos que transcurra desde que se inicia la transición hasta que termina. Luego

especificamos los atributos que vamos a escalar.

Recordando que el punto de origen para escalar o dibujar una imagen con Kineticjs, lo que tenemos que hacer para que de la sensación que aparece del agujero se debe ir aumentando la escala hasta 1, para que quede del tamaño original, a la vez que se va cambiando la posición del topo.

Si solo se animaría la escala del dibujo lo único que haría sería estirar la imagen hasta su tamaño original hacía abajo, entonces se quedaría debajo debajo del agujero y es no es lo que se quiere.

6.3 Puntuaciones

Para guardar las puntuaciones no se utiliza una base de datos, ya que se ha decidido guardar sólo las 5 máximas puntuaciones y sería una complejidad extra innecesaria ya que podemos aprovecharnos del objeto *localStorage* que nos aporta HTML5.

```
localStorage.setItem(“clave”, “datos a guardar”)
```

Para guardar un dato necesitamos pasar una clave y el luego el dato que vayamos a guardar. El tipo de datos que podemos guardar son objetos de tipo númerico o “*strings*” y en este caso, se necesita guardar una lista de las 5 máximas puntuaciones, pero no se puede guardar el tipo “array” y para ello lo que se hace es transformar a “string” la lista para poder guardarla.

```
score.push points
score = JSON.stringify score
localStorage.setItem key, score
```

Utilizamos el método *“stringify”* del objeto JSON para transformar la “string”. Para luego volver a convertirlo a un objeto array y poder a acceder a sus métodos simplemente utilizamos el método *“parse”* del mismo objeto JSON.

```
score = JSON.parse score
```

6.4 Cordova CLI

Como ya se ha mencionado Apache Cordova es una plataforma que nos permitirá crear aplicaciones nativas a partir de HTML, CSS y JavaScript. Nos permite utilizar funcionalidades como la cámara, la galería, la vibración de un dispositivo y todo ello con una sola implementación sin tener que variar nada para un sistema operativo o otro.

6.4.1 Instalación y uso

Como las demás herramientas que hemos utilizado, Grunt, Bower... necesitamos tener instalado Node.js y NPM(Node Package Manager) y debemos ejecutar el siguiente comando para instalar Cordova.

```
npm install -g cordova
```

Una vez que se ha terminado la instalación debemos crear un proyecto. Para esto vamos a la carpeta donde queramos crear el proyecto y ejecutamos:

```
cordova create <nombre de la carpeta> <nombre del paquete> <nombre de la app>
```

Para añadir al proyecto las ditintas plataformas para las que queremos que la crear la aplicación debemos ejecutar:

```
cordova platform add firefoxos
cordova platform add android
cordova platform add browser
```

Estas son las plataformas que se han añadido a este proyecto. Esto genera dentro de la carpeta *platforms* del proyecto una carpeta de cada sistema operativo, que contiene los archivos y configuraciones necesarias para poder instalar o utilizar en cada uno de esos sistemas.

En este proyecto se utilizará el *"plugin"* de la cámara que nos permitirá acceso a la cámara del dispositivo. Para ello hay que ejecutar:

```
cordova plugin add cordova-plugin-camera
```

Si queremos ver los “plugins” que hay creados por Apache Cordova podemos verlos aquí con su respectiva documentación.

http://cordova.apache.org/docs/en/5.0.0/cordova_plugins_pluginapis.md.html#Plugin%20APIs

Para que los “*plugins*” funcionen correctamente debemos añadir a nuestro HTML el script *cordova.js*.

Dentro de la carpeta del proyecto se puede encontrar la carpeta *www*, en esta carpeta es donde debemos alojar todos los archivos que necesitamos en nuestras, tanto los archivos CSS, HTML, JavaScript, imágenes.

Para preparar la aplicación para cada una de las plataformas que hemos añadido tan solo debemos ejecutar:

```
cordova prepare
```

Con este conseguimos que nuestra aplicación que se encuentra dentro de la carpeta *www* se prepare para cada una de las plataformas que se encuentras dentro de la carpeta *platforms.*

6.4.2 Configuración

La configuración de la aplicación para las distintas plataformas es bastante sencillo, tan sólo debemos editar el archivo config.xml que se encuentra en la carpeta raíz del proyecto. En el siguiente enlace podemos encontrar las diferentes opciones de configuración que disponemos.

http://cordova.apache.org/docs/en/5.0.0/config_ref_index.md.html#The%20config.xml%20File

En este archivo es dónde debemos escribir el nombre de la aplicación, la descripción, el icono que aparecerá cuando se instale la aplicación en un dispositivo.

En este archivo también se ha añadido un par de opciones concretas para nuestra aplicación. Una de ellas es que la aplicación siempre se muestre con la pantalla en vertical y no se voltee cuando giremos el móvil, ya que debido al diseño planteado la pantalla para golpear topos se vería muy reducida. Y la segunda opción, es que la aplicación ocupe toda la pantalla del dispositivo incluyendo la barra de estado, donde aparece la hora y el estado de la batería, para conseguir mayor espacio para nuestra aplicación.

```
<preference name="FullScreen" value="true" />
<preference name="Orientation" value="portrait" />
```

6.5 Personalización de los topos

Para personalizar la apariencia de los topos se necesita tener acceso a la cámara y la galería del dispositivo, para ello hemos instalado el plugin de la cámara que nos proporicona Apache Cordova.

Para acceder a la galería o a la cámara a través del plugin simplemente debemos acceder a la función que se añade al objeto *navigator*. La llamada sería la siguiente:

```
navigator.camera.getPicture onSuccess, onFail, options
```

Cuando la ejecución de la función es exitosa se lanza la función *onSuccess* pasandole cómo parámetro los datos de la imagen. En caso de que la función falle se ejecuta *onFail* pasándole cómo parámetro el error. Para decidir desde donde se quiere obtener la foto o cómo queremos recibir los datos de la imagen, se debe pasar estás decisiones a través del objeto options. En nuestro caso el objeto *options* sería el siguiente:

```
options = {
  sourceType: type
  destinationType: Camera.DestinationType.URL
  correctOrientation: true
}
```

type tendrá el valor *Camera.PictureSourceType.CAMERA* en caso de que el usuario quiera acceder a la cámara o el valor *Camera.PictureSourceType.PHOTOLIBRARY* si quisiera obtener la imagen de una que tuviera ya disponible en su galería de imágenes.

En la opción *destinationType* tenemos el valor *Camera.DestinationType.URL* y significa que si la obtención de la imágen es correcta queremos que el párametro que se envie a la función *onSuccess* sea la ruta completa de donde está guardada la imagen.

Una vez que hemos recibido la ruta podemos crear un objeto imagen a partir de ella de la siguiente manera.

```
img = new Image
img.onload = =>
  @image = new Kinetic.Image
    image: img
    height: @stage.getHeight()
    width: @stage.getWidth()
    clearBeforeDraw: true
    draggable: true

  @stage.add @layer
  @layer.add @image, @ellipse
  @layer.draw()
img.src = data
```

Dentro de método *onload*, método que se lanza cuando la imagen se ha cargado correctamente, crearemos una imagen con *Kinetic.Image* que nos permitirá una mejor manipulación de la imagen para personalizarla.

6.5.1 Personalización en el buscador

Para crear los topos personalizados no se ha recurrido a las funcionalidades que nos proporciona Apacha Cordova. La razón por lo que se ha decidido ésto es que para mostrar la el botón para acceder a la galería o mostrar el vídeo de la cámara para sacar una foto simplemente se añade el elemento *video* o *input* al final del cuerpo de la aplicación sin incluirles tan siquiera un identificador para poder personalizarlo.

Con lo cual se ha decidido implementar esta parte por cuenta propia y hacer una diferencia, a la hora de la ejecución, entre si la aplicación se esta ejecutando en un buscador o es una aplicación nativa del dispositivo. Para saber si donde se está ejecutando la aplicación comprobamos si la dirección desde donde se ejecuta contiene "*http*", ésto significa que se está ejecutando desde un navegador.

```
isNative: ->
  index = document.URL.indexOf "http"
  if index is -1
    true
  else
    false
```

7 Pruebas

Se han realizado pruebas en la aplicación para determinar si el resultado de la implementación es correcto. Las pruebas se han ido realizando a medida que se iba desarrollando y al ser una aplicación web, se han hecho pruebas en distintos navegadores y dispositivos.

7.1 Interfaz e interacción de pantallas

7.1.1 Responsive Design

Acción	Dispositivo	Resultado
Los elementos se adaptan y distribuyen en la pantalla según lo planificado	Chrome v44.0.2403.61 beta	Correcto
	Firefox v38.0	Los elementos salen descolocados y en orientación invertida.
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	Los elementos salen descolocados y en orientación invertida.

El problema surge al utilizar la ultima versión del propiedad *display: flex* para distribuir de forma uniforme los que constan la aplicación.

Solución: La solución fue añadir una versión anterior sobre esta propiedad CSS para que se ejecutará ésta en caso de que la nueva versión no estuviera implantada en el dispositivo o navegador.

7.1.2 Direccionamiento de pantallas

Acción	Dispositivo	Resultado
Se ejecuta la animación cuando navegamos entre las distintas pantallas.	Chrome v44.0.2403.61 beta	Aparece la nueva ventana pero no se ejecuta ninguna animación.
	Firefox v38.0	Aparece la nueva ventana pero no se ejecuta ninguna animación.
	BQ Aquaris E5 4G Android v4.4.4	Aparece la nueva ventana pero no se ejecuta ninguna animación.
	Samsung S3 Mini Android v4.1.2	Aparece la nueva ventana pero no se ejecuta ninguna animación.

El problema surgía cuando al añadir los atributos que ejecutan la animación entre las pantallas ya se había quitado la clase *show* que es la que hace que se vea una pantalla o otra.

Solución: La solución ha sido añadir el atributo que ejecuta la animación para que desparezca la pantalla y pasado el tiempo de animación quitarle la clase *show*, y para mostrar la nueva pantalla añadimos ambos atributos a la vez ya que lo que se quiere es que se empiece a ver la pantalla desde el mismo momento que empieza la animación.

7.2 Interfaz e interacción de la partida

7.2.1 Lógica del tablero

Acción	Dispositivo	Resultado
Se dibuja correctamente tanto la base del tablero como los agujeros.	Chrome v44.0.2403.61 beta	Correcto
	Firefox v38.0	Correcto
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	Correcto

En este caso no ha habido ningún problema, la única corrección que se hizo fue bajar la escala en “y” a la hora de dibujar los agujeros para que quedarán mejor estéticamente.

7.2.2 Dibujar topos

Acción	Dispositivo	Resultado
Los topos se dibujan correctamente encima de sus agujeros.	Chrome v44.0.2403.61 beta	Aparecen desplazados hacia la derecha.
	Firefox v38.0	Aparecen desplazados hacia la derecha.
	BQ Aquaris E5 4G Android v4.4.4	Aparecen desplazados hacia la derecha.
	Samsung S3 Mini Android v4.1.2	Aparecen desplazados hacia la derecha.

La razón de que se produzca este error es que en vez de tener en cuenta que el punto desde donde se empieza a dibujar la imagen es desde la esquina superior izquierda y no desde el centro de la imagen como podría ser a la hora de dibujar un circulo.

Solución: Como se calculo que la posición central sería el doble del radio que utilizábamos al dibujar, con lo cual la solución es fácil, reducimos esa distancia al radio utilizado para dibujar los agujeros.

7.2.3 Animación de los topos

Acción	Dispositivo	Resultado
Se ejecuta la animación que hace aparecer al topo del agujero.	Chrome v44.0.2403.61 beta	Correcto
	Firefox v38.0	No se ven aparecer los topos para si que se ejecuta la función por que se restan las vidas.
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	Correcto

El problema es un *"bug"* o fallo que tiene la librería Kineticjs en Firefox. El fallo es que cuando se anima la escala de un objeto y la escala inicial es 0 no funciona.

Solución: La solución ha sido en vez de poner una escala inicial de 0, poner un muy pequeña pero sin que llegue a ser 0, con lo cual el nuevo valor de la escala es 0.01

7.2.4 Pausar el juego

Acción	Dispositivo	Resultado
Comprobar que se pausa bien el juego y continua normalmente cuando lo reanudamos.	Chrome v44.0.2403.61 beta	Aparecen topos solamente en unos pocos agujeros y si no se golpean no desaparecen.
	Firefox v38.0	Aparecen topos solamente en unos pocos agujeros y si no se golpean no desaparecen.
	BQ Aquaris E5 4G Android v4.4.4	Aparecen topos solamente en unos pocos agujeros y si no se golpean no desaparecen.
	Samsung S3 Mini Android v4.1.2	Aparecen topos solamente en unos pocos agujeros y si no se golpean no desaparecen.

En este caso han surgido 2 problemas, uno el que solo aparecen topos en unos pocos agujeros y segundo que no desaparecen si no se les golpea.

Soluciones:

1. Cuando se pausa el juego se eliminan los topos que están fuera de sus agujeros en ese momento para que al reanudar no tener topos ya salidos y no nos de tiempo a pulsarles y nos resten vidas. El problema es que no se eliminaban esos topos de la lista en la que aparecen los topos que están fuera, con lo que al reanudar la partida esos agujeros constan como ocupados aunque no haya nada en ellos. La solución ha sido vaciar la lista de los topos que están fuera cuando se pausa el juego.

2. Tenemos un variable *"paused"* que indica si la partida esta pausada o no. Y cuando se pausa, la función que se ejecuta cuando va a desparecer el topo utiliza esta variable para que si el juego esta pausado no se ejecute. El problema surgía que cuando se vuelve a reanudar el juego, no se cambiaba el valor de esta variable y no se ejecuta la función para ocultarlos.

7.2.5 Elegir dimensiones tablero

Acción	Dispositivo	Resultado
Dibujar correctamente los agujeros a partir de distintas dimensiones del tablero.	Chrome v44.0.2403.61 beta	Correcto
	Firefox v38.0	Correcto
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	Correcto

Todo sale correctamente en esta prueba.

Acción	Dispositivo	Resultado
Los topos aparecen correctamente y del tamaño correcto	Chrome v44.0.2403.61 beta	El tamaño es correcto pero las posiciones son incorrectas a partir de la primera vez.
	Firefox v38.0	El tamaño es correcto pero las posiciones son incorrectas a partir de la primera vez.
	BQ Aquaris E5 4G Android v4.4.4	El tamaño es correcto pero las posiciones son incorrectas a partir de la primera vez.
	Samsung S3 Mini Android v4.1.2	El tamaño es correcto pero las posiciones son incorrectas a partir de la primera vez.

Antes de implementar esta opción las dimensiones eran fijas y por tanto las posiciones de los topos se mantenían de una partida a otra y no se borraban las posiciones de la partida anterior y no había problema, en cambio, ahora hay que cambiarlas.

Solución: Cuando calculamos las nuevas posiciones de los topos debemos asegurarnos de que no hay posiciones guardadas y en caso de haberlas borrarlas.

7.3 Personalizar topos

Acción	Dispositivo	Resultado
Cargar imagen para personalizar tanto a partir de la cámara como de la galería.	Chrome v44.0.2403.61 beta	La imagen capturada por la cámara se carga bien, la obtenida de la galería no.
	Firefox v38.0	La imagen capturada por la cámara se carga bien, la obtenida de la galería no.
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	Se carga la imagen pero con la orientación incorrecta.

El error que surge en los navegadores es que debido a temas de seguridad cuando seleccionamos un archivo del sistema el navegador nos oculta la verdadera ruta donde se encuentra el archivo con lo que no podemos crear la imagen al igual que cuando lo hacemos en el dispositivo móvil.

Solución: La solución ha sido leer el archivo a partir de la API FileReader y el método *readAsDataURL* que devuelve una URL que representa los datos del archivo y de está manera ya podemos crear la image.

```
reader = new FileReader
reader.onload = (file) =>
  imageUri = file.target.result
  @onPhoto imageUri
reader.readAsDataURL file
```

Para resolver el problema de la orientación cuando obteniamos la foto a través de la cámara simplemente se ha añadido una opción más al método que lanza la cámara, que gira la la imagen a la orientación correcta durante la captura.

```
correctOrientation: true
```

Acción	Dispositivo	Resultado
Arrastrar y agrandar la imagen.	Chrome v44.0.2403.61 beta	Correcto
	Firefox v38.0	Correcto
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	Se puede agrandar pero al arrastrar la imagen queda una copia por debajo que no mueve.

El error surge por un problema en el sistema operativo android que utiliza la versión WebKit 534.30 para renderizar las páginas web y no borra la imagen después de la primera vez que se dibuja, con lo que la imagen que dibujamos primero es la que se queda por duplicado.

Solución: para la solución proponen un retraso a la hora de dibujar la imagen, para todos los casos probados con un retraso de 5ms es suficiente.

Acción	Dispositivo	Resultado
Crear circulo donde podremos recortar la imagen para ajustar a los topos personalizados.	Chrome v44.0.2403.61 beta	Correcto
	Firefox v38.0	Correcto
	BQ Aquaris E5 4G Android v4.4.4	Correcto
	Samsung S3 Mini Android v4.1.2	No se puede recortar la imagen.

El problema surge que para recortar la imagen se utiliza la propiedad *globalCompositeOperation=”destination-in”* que lo que hace es que cuando dibujemos ese circulo lo que este en la capa de destino sólo se mantenga lo que quede en el interior de este circulo.

Solución: De momento no se ha encontrado la solución para este problema. Asi que de momento se ha optado que la versión mínima de Android para utilizar este aplicación de forma nativa sea la versión 4.4. Abriendo la aplicación desde el navegador funciona igual que el navegador de un ordenador de mesa.

8 Conclusiones

Se expondrán las conclusiones del proyecto, tanto a nivel personal como en el cumplimento de objetivos.

8.1 Cumplimientos de objetivos

Debido a diversos problemas de factor personal no se ha cumplido las previsiones temporales previstas. Durante la duración del proyecto han surgido eventos que me ocupaban por completo.

El análisis y el diseño ha ocupado más tiempo de la planificación, debido a que se han encontrado una inmensidad de herramientas para el desarrollo web que no se esperaban encontrar. El inicio de la implementación se retraso un poco debido a los problemas personales comentados anteriormente y, además, también se invirtieron más horas de las esperadas, aunque no haya sido un gran exceso de horas, debido a que no se profundizó lo suficiente en el tiempo de aprendizaje.

A continuación se muestra un gráfico con las diferencias entre las horas estimadas y las horas reales aproximadas al final del proyecto.

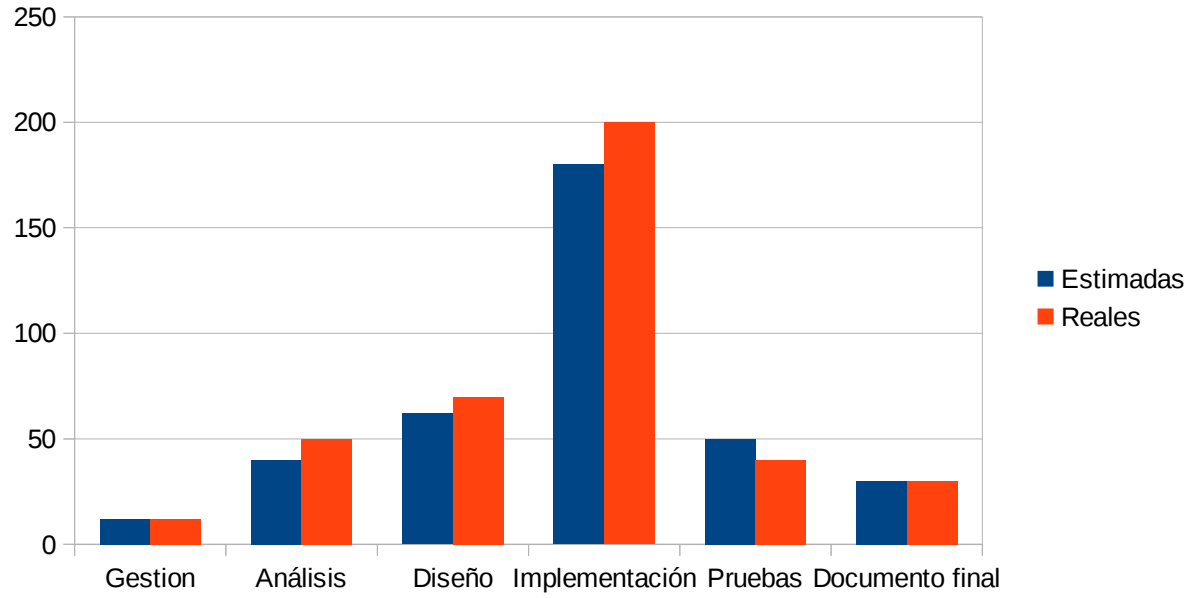


Ilustración 16: Horas estimadas y reales

8.2 Conclusiones personales

La realización del proyecto ha supuesto un gran avance a nivel personal. La investigación, el autoaprendizaje y las habilidades de programación han mejorado notablemente. Ha sido satisfactorio poder ver que he podido gestionar un proyecto desde su inicio hasta el final autonomamente, así como poder afrontar y resolver los diversos problemas que han surgido a lo largo de todo el desarrollo del proyecto.

También se han despertado inquietudes de profundizar más en el tema desarrollo web, ya que he descubierto la infinidad de posibilidades que nos puede llegar ofrecer el mundo de las aplicaciones web.

La sensación final es de satisfacción personal.

9 Líneas futuras

Durante el desarrollo de la aplicación se han pensado distintas funcionalidades que podrían ampliar la aplicación.

- Crear nuevos modos de juego para que los usuarios no se cansen de la aplicación al ser monótona.
- Que los topos ofrezcan distintas ventajas o desventajas a las ya implementadas.
- Crear una clasificación mundial o estatal.
- Añadir soporte WebGL para poder mostrar gráficos en 3D.

10 Bibliografía

Información

- https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/

Documentación

- **Grunt**
  - http://gruntjs.com/getting-started
  - https://www.npmjs.com/package/grunt-contrib-watch
  - https://www.npmjs.com/package/grunt-contrib-coffee
  - https://www.npmjs.com/package/grunt-contrib-stylus
  - https://www.npmjs.com/package/grunt-contrib-jade
  - https://www.npmjs.com/package/grunt-contrib-uglify
  - https://www.npmjs.com/package/grunt-contrib-copy
  - https://www.npmjs.com/package/grunt-contrib-concat
- **Kineticjs**
  - http://agavestorm.com/kineticjs/
- **Cordova**
  - http://cordova.apache.org/docs/en/5.0.0/index.html
  - http://cordova.apache.org/docs/en/5.0.0/cordova_events_events.md.html#Events
  - https://github.com/apache/cordova-plugin-camera/blob/master/README.md

Ejemplos

- **Kineticjs**
  - http://www.html5canvastutorials.com/kineticjs/ (Not maintained)