Trivial de festa major

Antoni Rosa Ruiz
Programa en Desarrollo de aplicaciones para dispositivos móviles

Carlos Caballero González

22/06/2015

FICHA DEL TRABAJO FINAL

Título del trabajo:	**Trivial de festa major**
Nombre del autor:	**Antoni Rosa Ruiz**
Nombre del consultor:	**Carlos Caballero González**
Fecha de entrega (mm/aaaa):	06/2015
Área del Trabajo Final:	**Desarrollo de aplicaciones para dispositivos móviles**
Titulación:	*Plan de Estudios del Estudiante*

Resumen del Trabajo (máximo 250 palabras):

El proyecto plantea las diferentes fases del desarrollo de una aplicación móvil (app) con el objetivo final de obtener un producto que cumpla las objetivos planteados.
La app se plantea como **un juego de preguntas y respuestas múltiples para uno o varios jugadores** simultáneos.
El juego permitirá crear **partidas individuales y con otros jugadores al mejor de un número determinado de rondas**, dónde se obtendrán puntos por cada respuesta correcta y por ser el primero en responder. Ganará el jugador con más puntos al finalizar todas las rondas.
Para que este tipo de juego sea interesante debe haber un repositorio de preguntas suficientemente grande y variado. Por este motivo se estima que sería interesante que la app tuviera también **un módulo de edición colaborativo para la redacción de preguntas**. Para asegurar la calidad de las preguntas provenientes del módulo de redacción colaborativa, deberán ser votadas por un mínimo de usuarios antes de ser incorporadas automáticamente al repositorio. El sistema de votaciones, para evitar vandalismo, se podría basar en **un sistema de métricas de la reputación de los usuarios que, a cuantas más preguntas aceptadas, más valiera su voto** (más «karma»).
La temática del juego se quiere centrar en las festividades locales de la ciudad de Mataró, pero el desarrollo de los diferentes módulos (El *engine* para permitir la mecánica de juego, el módulo de introducción de preguntas, etc.) es independiente de este hecho, ya que esto sólo afectará al *artwork* y detalles finales del juego.

Abstract (in English, 250 words or less):

The project aims at developing of a mobile application that meets the initial objectives.

The application will consist of trivia game for one or multiple simultaneous players.

Each game should have several rounds of questions that players can answer alone or competing against other players. When playing with other users for each round a score will be awarded for every correct answer and some extra point will go to the faster user to answer. The player with the highest score after all the rounds wins the game.

To make the game more interesting there should be a large and varied enough repository of questions. In order to accomplish it the app also aims at developing a module for collaborative editing and writing questions. To ensure the quality of the questions from this collaborative module, all questions must be voted by some users before being incorporated into the game repository.

To prevent vandalism, the voting part of the collaboration module should be based on a system of user reputation metrics: A vote from a user with more accepted questions is worth more than this of other users with less accepted questions.

Another objective is targeting the contents to promote the local festivities in the city of Mataró, but the development of the various modules (the engine to allow the game play, the module for writing questions, etc.) can be done independently of the as it will only affect the final artwork details.

Palabras clave (entre 4 y 8):

Juego, Trivial, Multijugador, Phonegap, PHP

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

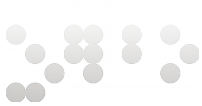
La realización de un juego como Trabajo de Final de Postgrado (TFC) le da un punto de interés lúdico al proyecto. La temática del juego centrada en las festividades locales de la ciudad de Mataró, muy queridas por los locales, se plantea como una aportación interesante a la promoción de la cultura local del municipio en el contexto de las nuevas tecnologías.

Cabe resaltar que la festividad cuenta ya con diferentes aportaciones de ciudadanos que han contribuido mediante blogs o páginas web que se usaran para generar un *set* inicial de información, aun y que uno de los objetivos de la app es que el contenido sea generado por los propios los usuarios. Explorar maneras para hacer interesante la edición de información de forma colaborativa, ideado con tinte de *gamificación*, aporta, además, un matiz interesante y actual al desarrollo.

Por otro lado, el tipo de juego no será muy exigente en cuanto a recursos gráficos y por esto, parece adecuado para un desarrollo no nativo en Phonegap. Phonegap ofrece el atractivo adicional de que el juego podría publicarse en diferentes plataformas de forma más o menos inmediata, sin ninguna o pocas adaptaciones. Finalmente, el tipo de proyecto en su concepción parece realizable en el contexto de este TFP, ni que sea para poner los cimientos de la aplicación.

1.2 Objetivos del Trabajo

En términos generales la promoción de la cultura local en el contexto de las nuevas tecnologías es uno de los objetivos implícitos del proyecto. Para conseguirlo se intentará conseguir una aplicación que responda a una mecánica de juego concreta y que, eventualmente, pudiera ser publicada en la tiendas (*markets)* de apps de la principales plataformas móviles.
Se pretende que el uso del juego pueda aportar a partes iguales diversión y conocimientos.
Así mismo se plantea que la app verse enriquecida mediante la aportación de los propios usuarios mediante la aportación de nuevas preguntas y anécdotas. Para conseguirlo se propone un sistema de



puntos y clasificaciones que aporten elementos de *gamificación*[1] en los contenidos aportados por el usuario

1.3 Enfoque y método seguido

El método de trabajo viene prefijado según las entregas de las prácticas de evaluación continuada en las que se encuadran el proyecto. A grandes rasgos, su puede argumentar que el desarrollo del proyecto corresponde con el del modelo en cascada que comprende tareas de Análisis, Diseño, Implementación y Verificación.
A continuación se detallan brevemente la fases:

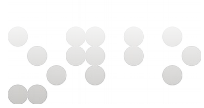
1. **Estudio previo y análisis de los requerimientos**: En esta fase se analiza y desglosan los requerimientos. En este caso se basa en el tema escogido para la aplicación del proyecto.
2. **Planificación**: En esta fase de planifica la duración de las diferentes partes del diseño y implementación del proyecto.
3. **Diseño de la arquitectura técnica**: En esta fase se estudia y propone una arquitectura física para dar cabida al servicio desarrollado de acuerdo al desarrollo de software.
4. **Diseño funcionalidad/Diseño la gráfico**: En esta se se estudia y propone la funcionalidad concreta de acuerdo con sus módulos lógicos de programación y la implementación concreta que estas funcionalidades tendrán de cara al usuario final.
5. **Implementación**: Esta ase comprende tanto **la codificación** de la solución como la puesta en marcha de los diferentes **elementos arquitectónicos** que la comprendan (Repositorios de código, Servidores, BD, etc...) y diseño **gráfico de la solución**.
6. **Pruebas de validación de la solución**: Esta fase, **no estrictamente secuencial** ya que parte de ella empieza durante la implementación comprende el conjunto de **pruebas y validaciones** que se llevarán a cabo para validar que se cumple con los requerimientos y objetivos marcados

El punto 6 no ha sido ejecutado pero se contempla como posibles tareas futuras del proyecto ya fuera del alcance se esta entrega.

1.4 Planificación del Trabajo

Esta es una aproximación en la planificación, que viene prefijada según el de las entregas de las prácticas de evaluación continuada.

1 Djaouti, D., Alvarez, J., and Jessel, J. P. (2010) Can Gaming 2.0 help design Serious Games?: a comparative study. Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games, pp 11-18



Se describen de forma genérica las características que podrán entregarse por la fecha y las horas o esfuerzo aproximado de las mismas.

Planificación según 2 entregas

- Tercera Práctica de Evaluación Continuada. 27/04/2015
  Tiempo total: 27 días

  Tareas:

  1. Identificar las funcionalidades y flujos del usuario. *35% esfuerzo*
  2. Crear la interfaces mediante wireframes y aplicando los patrones de diseño de la plataforma destino. *15% esfuerzo*
  3. Diseño visual de la app *20%*
  4. Realización de un prototipo o wireflow *30%*

- Cuarta Práctica de Evaluación Continuada. 21/06/2015
  Tiempo total: 58 días

  Tareas:

  1. Preparación del desarrollo 2% Esfuerzo
  2. Interfaces gráficas 13% Esfuerzo
  3. Diseño de la BBDD y backend 20% Esfuerzo
  4. Implementación de las actividades 20% Esfuerzo
  5. Pruebas de funcionamiento, incidencias y versión final 25% Esfuerzo
  6. Documentación 20% Esfuerzo

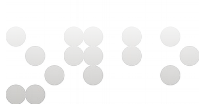
Derivas

En la práctica el punto 4 de la segunda entrega se ha demostrado sub estimado, consumiendo una gran cantidad de tiempo no prevista > 60 o 70% del total.

1.5 Breve sumario de productos obtenidos

Este proyecto da como resultado diferentes productos:

- Documento de análisis funcional, técnico detallado de la aplicación que se quiere obtener y el diseño arquitectónico del mismo (este documento)
- Memoria formal del desarrollo del proyecto (este documento)
- Dos aplicaciones (cliente y servidor) que componen la aplicación



1.6 Breve descripción de los otros capítulos de la memoria

- Funcionalidad: Descripción de las funcionalidades, actores implicados y interacciones a implementar

- Datos: Relación de las entidades de datos para dar respuesta a la necesidad de permanencia de datos derivadas de la funcionalidad a implementar.

- UML: Descripción de la clases a implementar durante el desarrollo para administrar los datos de la aplicación y implementar les funcionalidades deseadas.

- Diseño y especificación de pantallas: Especificación de las diferentes pantallas a desarrollar

- Desarrollo: Descripción de las herramientas, detalles de implementación y decisiones concretas de la arquitectura técnica de la aplicación y muestras de la aplicación desarrollada.

Después vienen los capítulos habituales en las memorias y auto descriptivos: Conclusiones, Glosario, Bibliografía y anexos

2. Funcionalidad

2.1 Introducción

Para poder definir la funcionalidad de la aplicación definiremos primero los «actores» o roles principales de los usuarios que interaccionan. Después identificaremos las acciones principales, debidamente priorizadas, y finalmente lo relacionaremos en un diagrama de secuencia. Par aquellas acciones que contengan implícitas otras acciones más detalladas.
De este modo se puede no sólo definir la funcionalidad y flujos de los usuarios en la aplicación, sino que se puede priorizar los casos de uso habituales.

2.2 Actores

- *Usuario jugador*: Se trata del usuario principal de la aplicación, aquel usuario casual que se ha descargado la aplicación con la intención de jugar.

- *Usuario administrado*r: Se trata del usuario administrador del *backend. No tiene ningún caso de uso especial con el *app del dispositivo móvil, sólo a l interfaces web de administración no disponible a ningún usuario final.

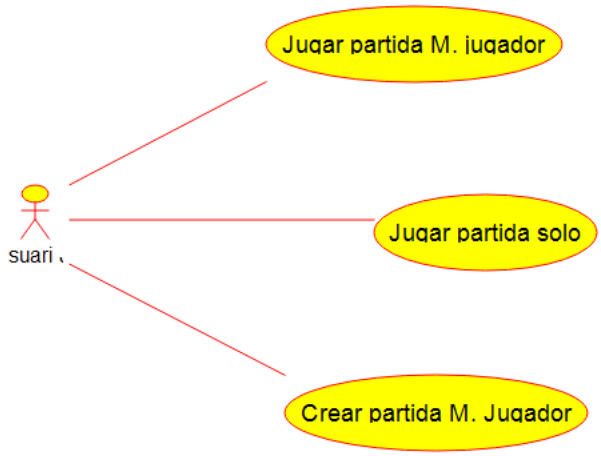
2.2.1 Casos de uso.

Acciones generales:

- Registrarse y iniciar sesión.
- Jugar partida individual o multijugador.
- Alta y votaciones de preguntas
- Consulta del perfil y clasificaciones

La votación de preguntas puede descomponerse en:

1. Busca de preguntas por categoría/usuario/data entrada/puntuación acumulada
2. Votación
3. Revisión del resultado



Drawing 1: Casos de usos generales 1

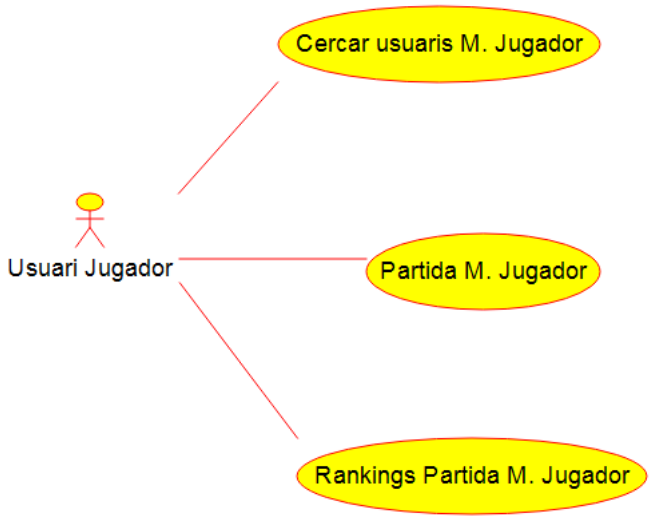
Drawing 2: Casos de usos generales 2

Detalle acciones multijugador:

- Invitar jugadores a una partida multijugador:

  1. Buscar de usuarios y invitarlos a una partida
  2. Aceptar unirse a una partida (usuarios invitados a la partida)

3. La partida con sus preguntas y mecánica de juego
4. Resultados (de la partida y rankings globales)

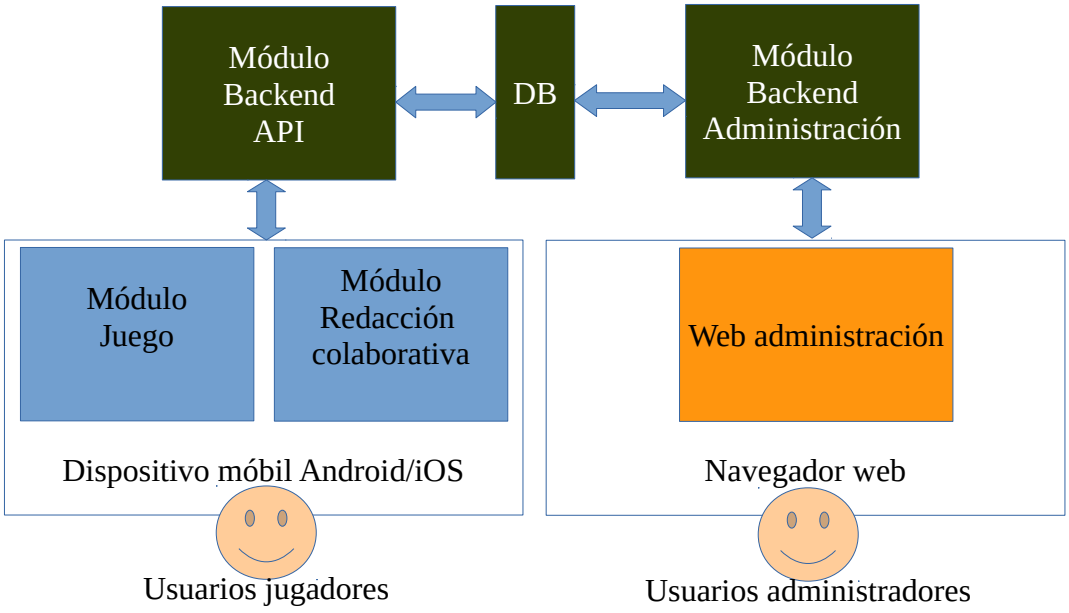


Drawing 3: Casos de usos generales 3

2.3 Módulos

Los diferentes módulos, o componentes funcionales que conforman el proyecto son:

- **Módulo backend y DB**: Comprende la parte del proyecto que se encarga de la persistencia de los datos, validar usuarios y coordinar partidas.
- **Módulo backend de administración y web de administración**: Módulo para gestionar los datos de usuarios y estadísticas en modo global. No se implementa como parte del proyecto pero se prevé como una posible mejora futura.
- **Módulo frontend aplicación**: La parte visible por lo usuarios, o sea la aplicación o aplicaciones para los dispositivos móviles que permite jugar partidas y editar preguntas de forma colaborativa.



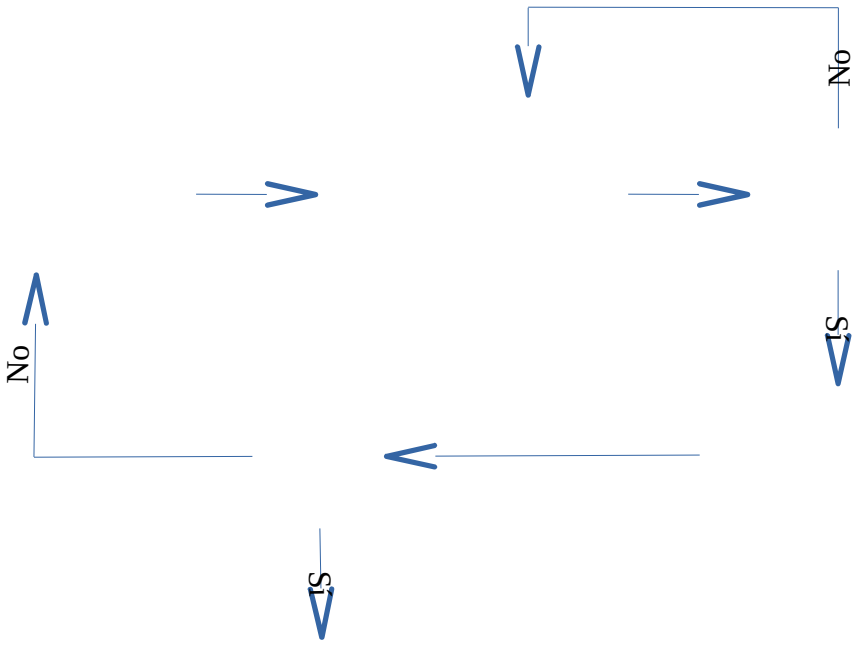
Drawing 4: Módulos del proyecto

2.3.1 Mecánica de juego

La app se plantea como un juego de preguntas y respuestas para varios jugadores simultáneos. El juego tiene que permitir crear partidas individuales y con otros jugadores al mejor de un número determinado de rondas. Se obtendrán puntos por cada respuesta correcta y también por ser el primero en responder. Ganará el jugador con más puntos al finalizar todas las rondas.

En el modo individual, la mecánica de juego se plantea sin mucha complicación:

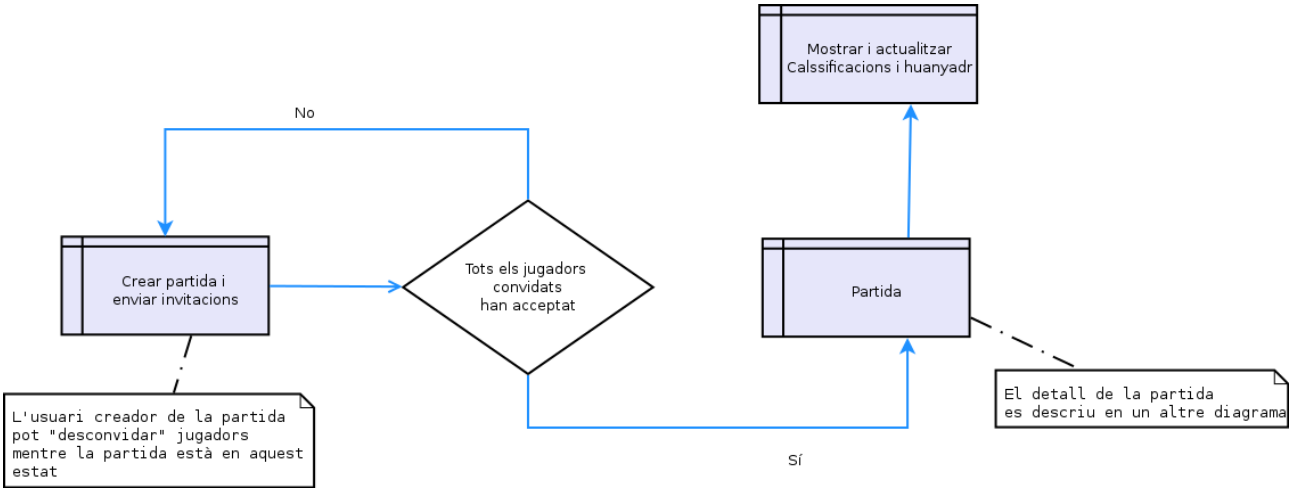
1. Un jugador crea la partida con un número concreto de rondas.
2. El jugador tiene un tiempo límite para responder a la pregunta de la ronda.
3. Al final del turno se dan los resultados del turno y se actualiza el panel de resultados de la partida en curso. En el último turno se muestra el resultado de la partida.



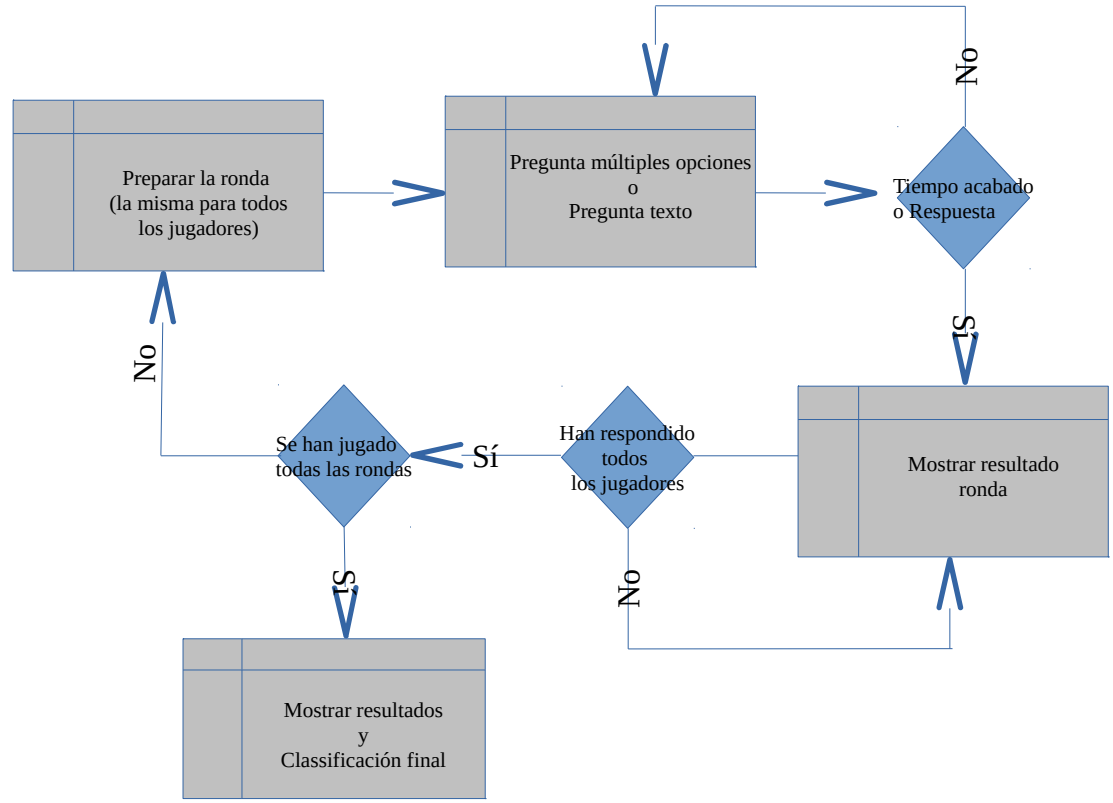
Drawing 5: Flujo de juego individual

En el modo multijugador, la mecánica de juego se plantea de forma semi síncrona:

1. Un jugador crea la partida con un número concreto de rondas.
2. Invita a otros jugadores que podrán aceptar o no unirse a la partida
3. Hay 2 tipos de rondas, las que las gana el primero a responder y las que las ganan todos los que acierten sin tiempos. En los dos casos se envía la misma pregunta a todos los participantes que tienen el mismo tiempo para responderla. Hay un de-calaje máximo de hasta un minuto de sincronización entre todos los jugadores para que no se rompa la ilusión de una partida síncrona entre todos los jugadores.
4. Al final del turno se dan los resultados del turno y se actualiza el panel de resultados de la partida en curso. En el último turno se muestra el resultado de la partida.
5. Gana el que haya obtenido más puntos. Sumará una victoria por la clasificación general de todos los usuarios del sistema.
    - Adicionalmente los puntos de todos los jugadores de suman a una clasificación general.



Drawing 6: Flujo de creación de una partida multijugador



Drawing 7: Flujo de juego multijugador

Mecánica edición de preguntas

1. Cualquier jugador puede añadir preguntas. Se clasifican por tipo de pregunta (múltiple con o sin foto o tipo acertar palabra) y tendrán un

nombre de pregunta identificativo.
Las preguntas no son publicadas automáticamente, .e hay que obtener los votos equivalentes en dos votaciones de jugadores en el cuartil 75 de experiencia.

2. Si una pregunta de un jugador es aceptada, el jugador recibe puntos N de experiencia.
3. Cualquier jugador puede votar preguntas de la lista, por cada N votaciones emitidas se ganan puntos de experiencia.
   Hay una clasificación general de experiencia

Clasificaciones

Hay tres listados generales de clasificaciones por todos los usuarios de la app relacionados con las partidas:

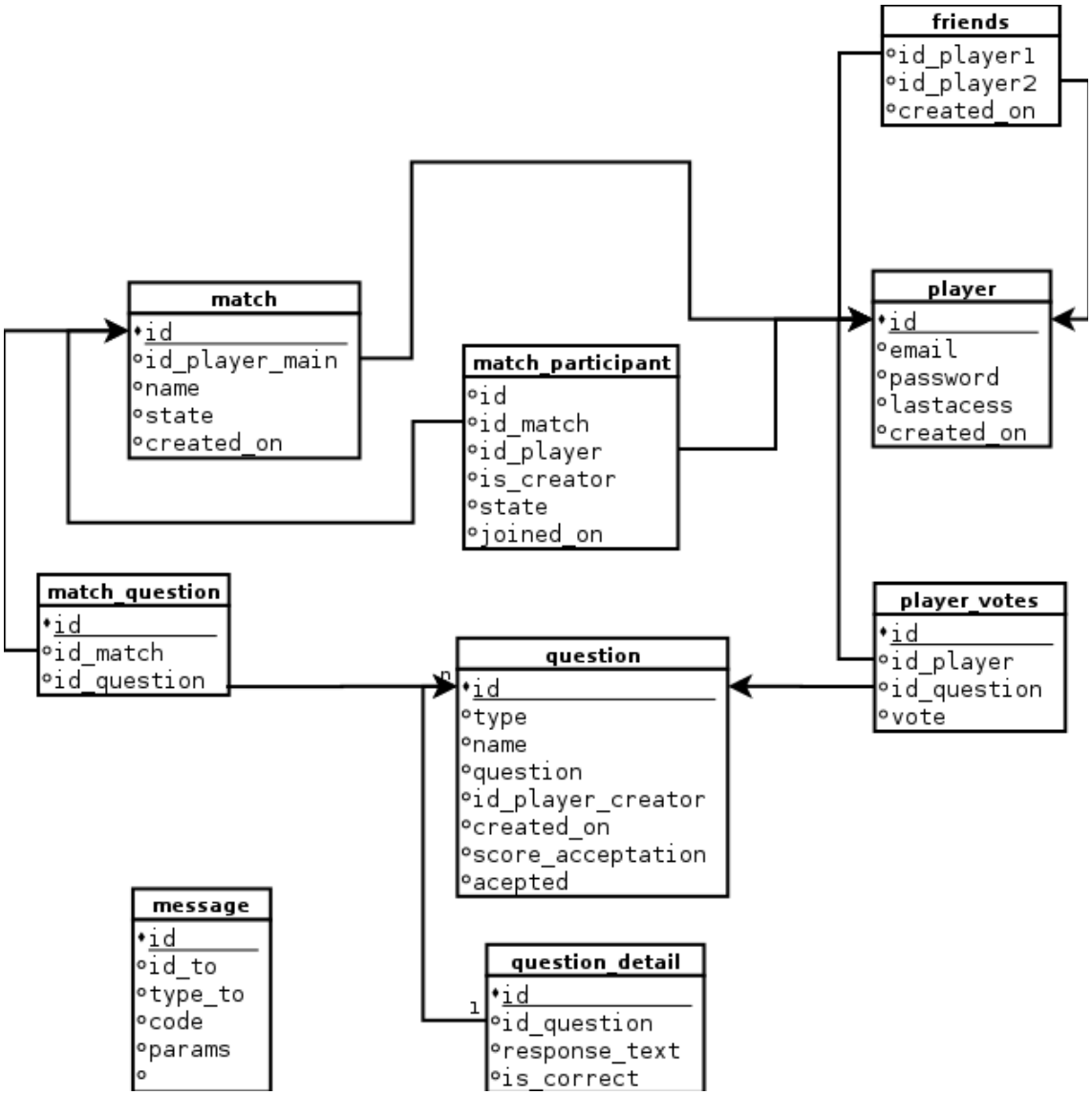
1. Los puntos obtenidos en las partidas
2. Un listado de la clasificación en la parte de generación de contenidos
3. Clasificación general. La media (45% + 55%) de los anteriores

3. Datos

Para dar respuesta a la especificaciones, mecánica y modos de juego de la se crearán las siguientes entidades (tablas en un sistema gestor de bases de datos robusto):

Player: Entidad guarda la información de cada usuario, credenciales para iniciar sesión y estadísticas de juego y acceso.
Match: Entidad que guarda información de las partidas multijugador (las que deben ser persistentes y compartidas).
Question: Entidad para las preguntas a usar en las partidas
Friends: Entidad para que cada usuario pueda tener una lista de usuarios “amigos”.
Message: Entidad para guardar los mensajes entre usuarios (invitaciones a partidas) y mensajes genéricos de coordinación entre participantes a una misma partida.
Player votes: Entidad para guardar las votaciones a las cuestiones de los usuarios.

Adicionalmente para poder representar las relaciones entre las entidades se creaaran estas tablas:
Match participant: Guarda que usuarios están en que partidas.
Match question: Guarda las preguntas que conforman una partida.

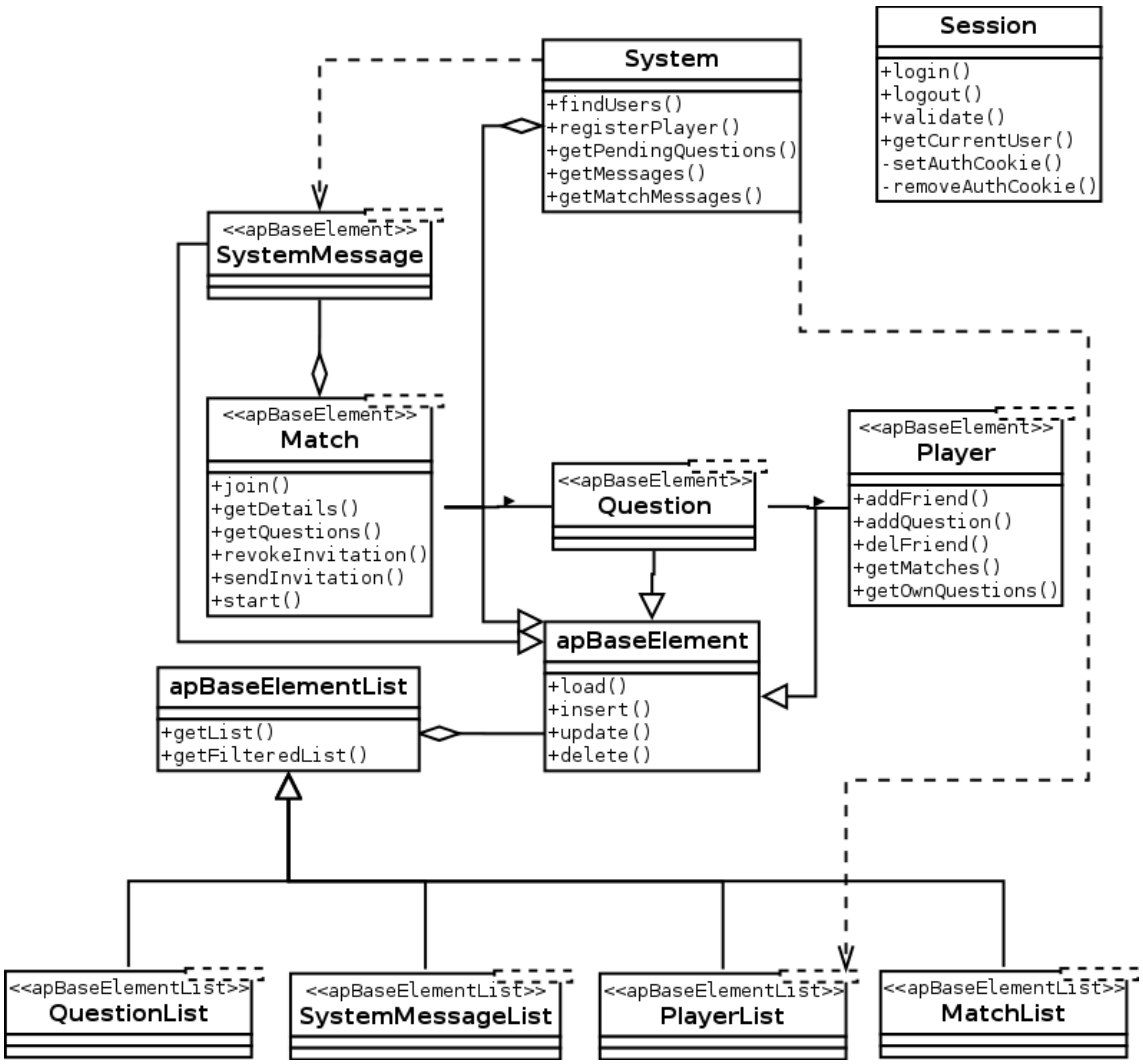


Drawing 8: Diagrama entidad relación de los datos

4. UML

4.1 Parte servidora

A continuación hay un diagrama UML de las clases implementadas en el servidor. Como se verá guarda una relación bastante directa con el modelo de datos. En el diagrama se muestran lo principales métodos i visibilidad de los mismos.



Drawing 9: Diagrama UML de clases implementadas en la parte servidor

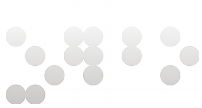
4.2 Parte cliente

En la parte cliente se han implementado las siguientes clases Javacript:

Session

Métodos principales:

- login()
- logout()
- register()



Player

Métodos principales:

- createMatch()
- getMatches()
- joinMatch()
- addQuestion()
- voteQuestion()

Match

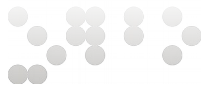
Métodos principales:

- load()
- start()
- nextRound()
- answer()

5. Diseño y especificación de pantallas

5.1 Introducción

La primera pantalla de la aplicación es la puerta de entrada a la funcionalidad y este impacto inicial hace que su organización y aspecto sea importante.
A pesar de que las aplicaciones para Android pueden seguir especificaciones y patrones muy claros definidos en los *Material design* de Google, al tratarse el proyecto de un juego, hay más de libertad a la hora de definir el aspecto de la interfaz. De hecho, de un juego se espera que lo haga un poco, y huya del aspecto formal de otro tipo de aplicaciones.
Esto no quiere decir que no haya que estudiar los diferentes patrones posibles de cara en el diseño de la aplicación.



5.2 Disposición

La aplicación dispone de un número relativamente pequeño de apartados. Este hecho lleva a establecer que la disposición de los elementos se efectúe mediante una lista plana de botones directos a la pantalla principal. Esto conlleva una navegación en árbol de un solo tronco inicial (la pantalla inicial)

Esta disposición tiene puntos fuertes y puntos débiles:

- A favor:
  - Simple
  - Opciones principales visibles

- En contra
  - A más opciones más esfuerzo navigacional para el usuario para moverse entre ellas y menor facilidad
  - No es muy sofisticada

También se estima usar un menú superior a la derecha (con la disposición e iconografía habitual a la plataforma) para acceder al perfil de usuario y clasificaciones, de forma parecida a otras aplicaciones.

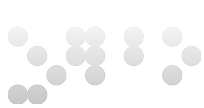
6. Desarrollo

6.1 Herramientas

El primer paso en el desarrollo es conseguir todas aquellas herramientas que se necesitarán para llevar a cabo el proyecto, bien ya sean herramientas con un rol activo en la arquitectura final de la solución desarrollada (p.e. Un sistema gestor de Base de datos) o coma parte necesaria del desarrollo estructurado (p.e. Un gestor de versiones de código fuente)

Listado de las herramientas para el desarrollo:

- Ubuntu 12.04. Sistema operativo base que se escoge este sistema para desarrollar por la familiaridad del desarrollador con el entorno y para poder tener en la misma máquina de desarrollo todas las herramientas y servicios que se usarían en una eventual publicación del resultado, específicamente del *backend*
- *Dia:* Para la creación de Diagramas
- *GIMP*: Para la manipulación de la imagenes
- *Balsamiq:* Para desarrollar los mockups de las pantallas



- *git: C*ontrol de versiones de código (VCS) para poder gestionar los cambios, versiones y futuras ampliaciones y ramificaciones del código.
- *Apache + PHP + PostgreSQL*: Aplicaciones servidor web (apache y php) y de base de datos (postgresql) necesarias para el desarrollo de la plante servidor del proyecto.

Por lo que respecta el uso de un VCS, siguiendo recomendaciones estándares [2] se optaría por usar un modelo con dos ramas principales (ramas, o más comúnmente referido en su termino inglés, *branches*) “develop” y “master”. Una vez la aplicación estuviera en producción podrían aparecer más ramas “releases” y “hotfixes”.

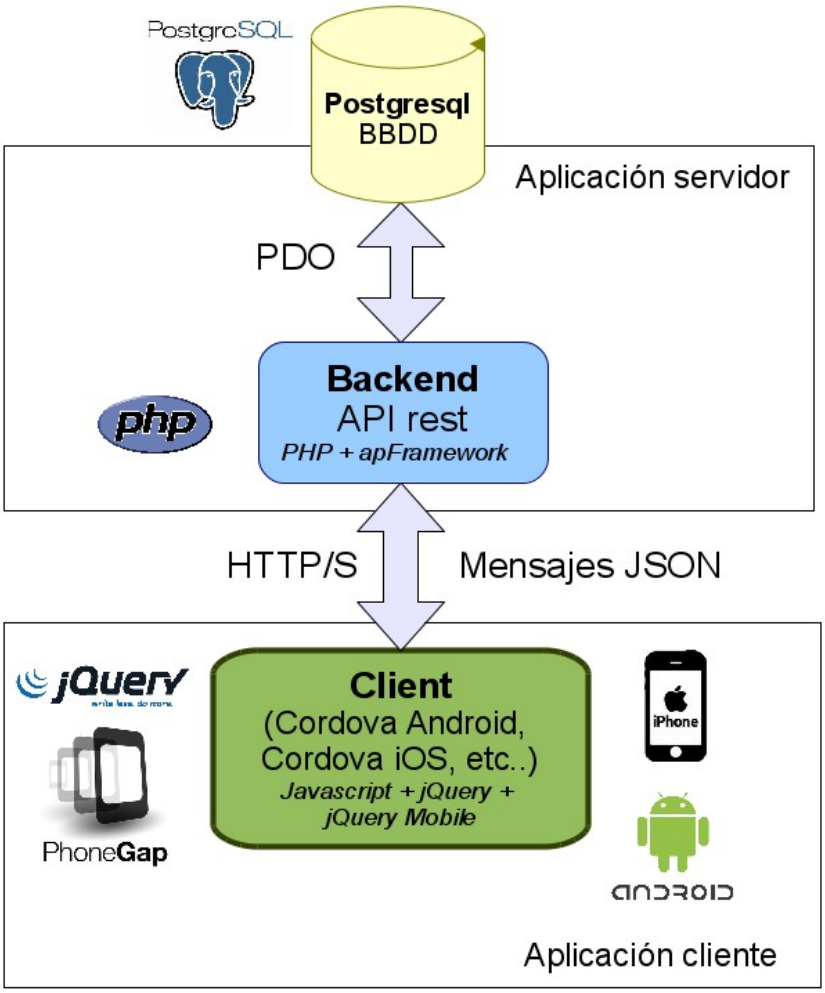
6.2 Arquitectura

La aplicación des de la perspectiva de la arquitectura consiste en 2 grandes bloques:

- La **aplicación cliente o *frontend:*** de los Clientes: Consiste la para los dispositivos móviles que permite al usuario final llevar a cabo las funcionalidades descritas (jugar y aportar preguntas de la forma descrita) en este documento.
- La **aplicación servidor o *backend* API**: Consiste en la aplicación encargada de toda la lógica de negocio, autenticación y autorización de acceso a datos y funcionalidades necesarias para que las dos aplicaciones nativas lleven a cabo sus funcionalidades
- La **aplicación *backend* de administracion**: Consiste en la aplicación encargada de gestionar a modo de administrador todos los datos de la aplicación. No se ha desarrollado en el presente proyecto, pero usando la API rest debería ser factible implementarla sin mucho esfuerzo-

Una vez publicada la aplicación, la comunicación entre las dos partes debería usar una conexión segura SSL, pero durante el desarrollo no es un requisito.

2 A successful Git branching model, blog personal de «Vincent Driessen» http://nvie.com/posts/a-successful-git-branching-model/



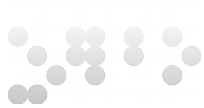
Drawing 10: Arquitectura técnica de la aplicación cliente/servidor

Aplicación *backend* o servidor

Se implementará mediante en PHP como una capa de webservices. La funcionalidad será accesible mediante una API rest. La idea es que se comporte como una API agnóstica tecnológicamente de los clientes que la usen de forma que pueda usarse con diferentes aplicaciones frontend si esto fuera necesario.

Librerías

- PHP 5.4
- PHP PDO Postgresql
- apFramework



End points servidor:

La carpeta “controllers” de la aplicación servidor contiene las clases que sirven la funcionalidad de la parte servidor de la aplicación.

Toda la información entrande o saliente va en formato JSON mediante esta estructura:

- Parámetro “result” → Boleano que establece si la llamda acabo sin errores
- Parámetro “messages” → Array para retornar mensajes, normalmente si hubo errores
- Parámtro “data” → Campo de respuesta con datos en función de la lógica de la operación

Ejemplo de salida de una operación que haya producido un error:

{ result: false, messages: [“ERR_NO_PLAYER_ID”]. “” }

que actuan como punt de entrada de la funcionalidad proporcionada para la aplicación cliente:

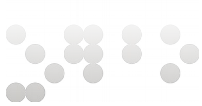
_credentials

Para ninguno de estos método es necesario tener sesión activa (o sea una *cookie* de sesión válida) en el momento de su llamada

Métodos:

- _login: Comprueba las credenciuals y retorna una cookie encriptada llamada “auth” con la informació del usuario, mediante la clase Session. La duración por defecto de esta cookie es de 10 años, de forma que se permanente.
- _logout: Destruye la cookie borrando su contenido y haciéndola caducar
- _validate: responde si la session actual establecida por la cookie es válida.
- _currentUser: responde el usuario de la cookie actual, si es que hay una
- _register: crea un nuevo usuario en el sistema

_match



Para todos estos método es necesario tener sesión activa (o sea una cookie de sesión válida) en el momento de su llamada o retornan un error de credenciales sin procesar la llamada.

Métodos:

- _sendInvitation:Envía un mensaje al usuario especificado para que se una a la partida especificada.
- _join: El usuario que llama este método acepta unirse a la partida especificada.

_player

Para todos estos método es necesario tener sesión activa (o sea una cookie de sesión válida) en el momento de su llamada o retornan un error de credenciales sin procesar la llamada.

Métodos:

- _getMatches:Obtiene una lista de los juegos en los que participi el usuaio.
- _createMatch:Crea una partida y la guarda en la base de datos.
- _addFriend: Añade un usuario a la lista de amigos
- _delFriend: Borra un usuario a la lista de amigos

_question

Para todos estos método es necesario tener sesión activa (o sea una cookie de sesión válida) en el momento de su llamada o retornan un error de credenciales sin procesar la llamada.

Métodos:

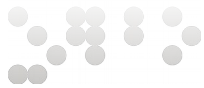
- _add: Añade una pregunta al sistema
- _vote: Procesa la votación de la pregunta

_system

Para todos estos método es necesario tener sesión activa (o sea una cookie de sesión válida) en el momento de su llamada o retornan un error de credenciales sin procesar la llamada.

Métodos:

- _findUsers: Busca y devuelve usuarios con el text de búsdqueda en el nombre o email



- _getMessages: Devuelve una lista de mensajes para el usuario (eventos de la aplicación. p.e: invitación para unirse a una partida)
- _getMatchMessages: Devuelve una lista de mensajes para la partida (eventos de sincronizaciṕmn de las partida. p.e: ronda finalizada)

Aplicación *frontend* de los Clientes

El *frontend* de la aplicación consistirá en las pantallas especificadas en este documento y será desarrollada como una aplicación web y compilada con la herramienta Cordova de Phonegap.

Librerías y herramientas:

- **Phonegap** (Cordova v.4.2): Se ha usado para obtener una versión instalable en dispositivos móviles y aprovechar la funcionaldad de cámara para las fotos del perfil.
- **jquery 1.11 + jquery Mobile 1.4:** Se ha usado para simplificar el desarrollo en javascrip y dar un look & feel más cercano a las plataformas móviles.

6.3 Código servidor

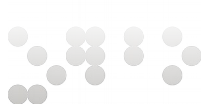
Se ha optado por aplicar el patrón MVC (del acrónimo inglés Model/View/Controller) para separar el modelo de datos, la lógica de control y el resultado que en este caso más que una vista es una respuesta en formato JSON. Para reducir el esfuerzo de programación se ha usado la librería apFramework[3] que da un marco MVC con el que se es tructuran los directórios de la siguiente manera:

- controllers/ carpeta para los archivos de los controladores del pryecto. Estos se corresponden a los **End points** comenados anteriormente.
- Models/ carpeta para los archivos con las clases desarrolladas para el proyecto:

  Match.php, MatchParticipant.php, Player.php, Question.php, Session.php, System.php, SystemMessage.php y OperationResult.php
- sql/ contiene el archivo create.sql para la creación de la base de dades desde cero.

Se ha decidido usar el estándar “*lowerCamelCase” para desarrollar el código del proyecto que consiste en los nombres de las variables compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra, excepto la primera letra de todas, que está en minúscula. En las

3 https://github.com/apfutura/apframework



clases se usa mayúsculas también en la primera letra.

Ejemplo nombre de variable: nombreVariableEjemplo

Ejemplo nombre de clase: NombreClaseEjemplo

6.4 Código cliente

Dos pantallas zonas html principales separadas para las pantallas que no necesitan sesión (privada) y las que si **public.html** y **app.html** respectivamente.

En los todos los casos hay separación entre la presentación HTML, CSS y la lógica implemendata en Javascript.

Estructura código JS

utils.js: funciones genéricas sin distinción parte pública y de aplicación. Por ejemplo «isValidEmail» retorna un boleano si la cadena de entrada es un correo electrónico bien formado. Otro ejemplo es el método postJSON que es el usado para la comunicación con el servidor API.
También tiene, el literal de la IP servidor. Idealmente se debería usar un nombre dominio, pero para versión actual esto no es así.

session.js: Es la clase para gestionar el usuario actual (login, registro, update) y que presenta de forma ordenada la funcionalidad de inicio i mantenimiento de sesión.

match,js, player.js: Clases con la lógica de juego.

public.js y **app.js**: funciones genéricas de la parte pública y privada de la aplicación que no son de ninguna pantalla en concreto

main.js, in-game.js, new-question.js: Métodos específicos de cada pantalla.

Hay 2 partes separadas:

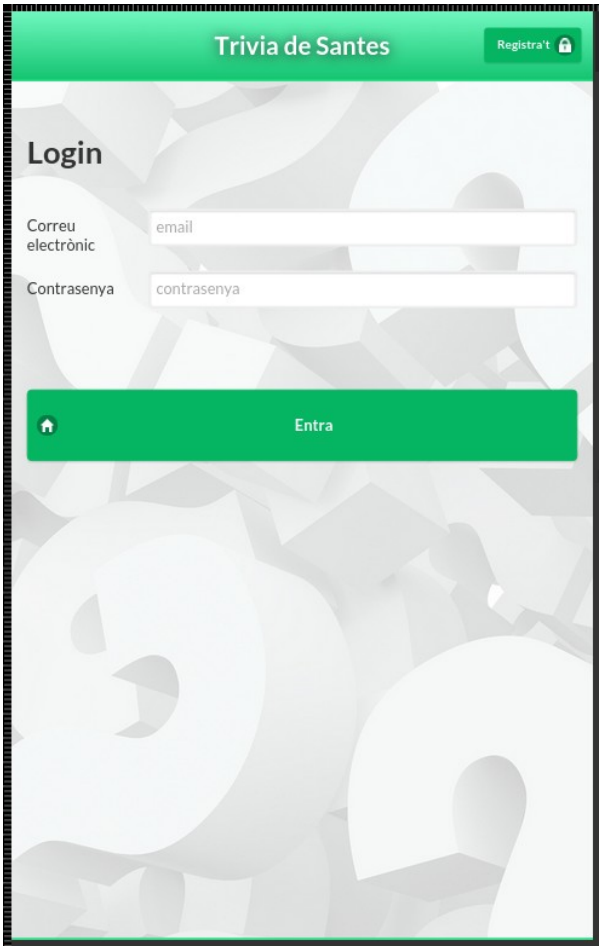
- Una dentro del evento *page-show* de jquery mobile (que esta dentro del document ready de jquery) y que inicializa los eventos visuales de los controles y pone valores por defecto que corresponda a los controles.
- Otra que son funciones, en el namespace global, pero que realizan algo para la pantalla (llamadas desde los eventos o como parte de la inicialización de valores)

Para que cordova pueda generar la aplicación correctamente, al no usar la estructura descrita el archivo index.html es necesario adapar la directiva <content src="index.html"> a <content src="public.html" dentro del archivo config.xml del proyecto.

6.5 Recorrido por la aplicación desarrollada

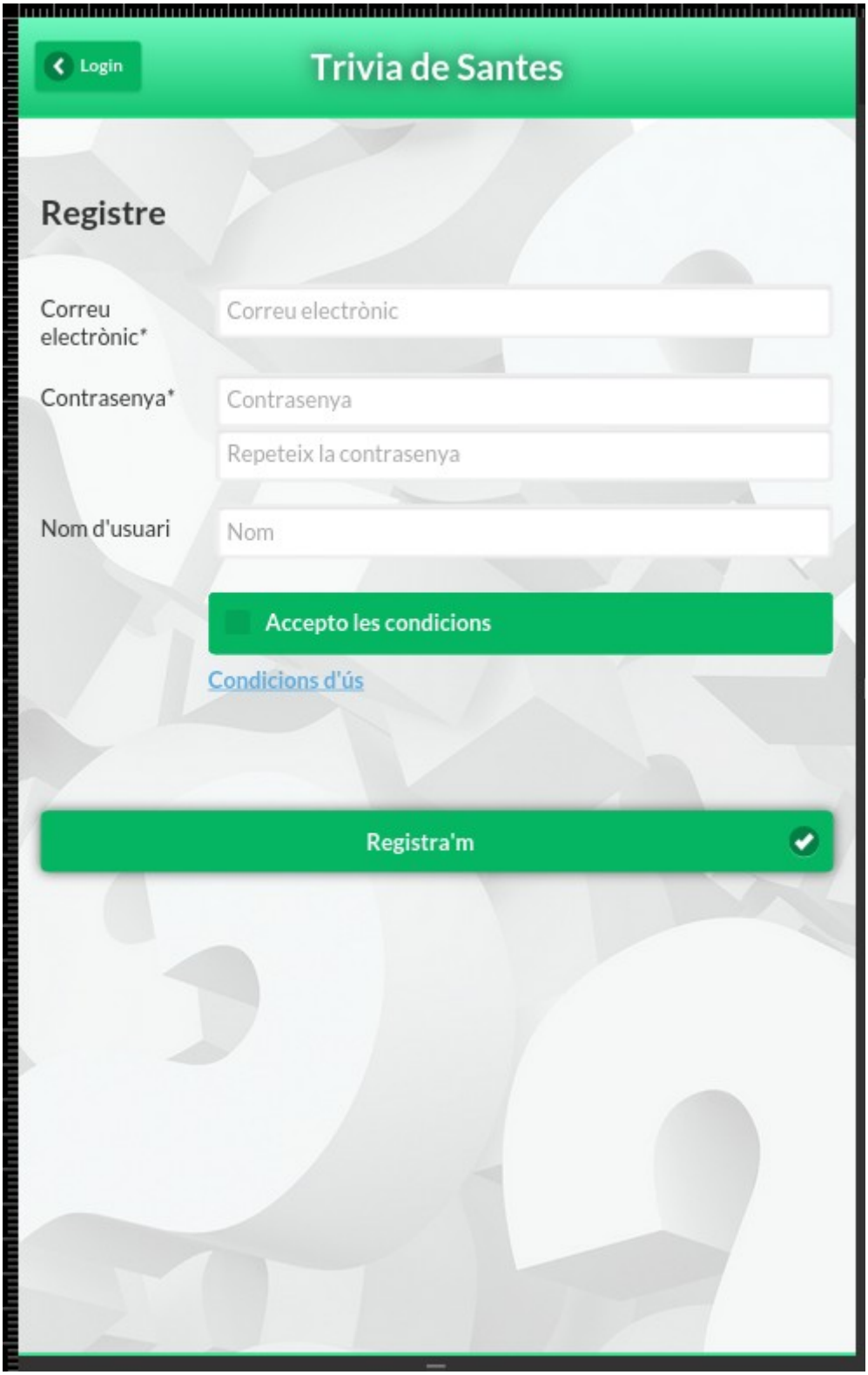
Primero se muestran la pantalla de inicio de sesión -A- y la de registro -B- en las cuales se accede cuando aun no se tiene ninguna cuenta o se ha cerrado la sesión. A continuación se muestra la pantalla principal de la aplicación -C- que permite acceder a la funcionalidad de juego y de edición de preguntas, dónde aun no hay ninguna partida pendiente para el jugador en cuestión. Finalmente se muestran las pantalla de juego individual -D-, -E- y de alta de pregunta -F-.

A - Pantalla de login



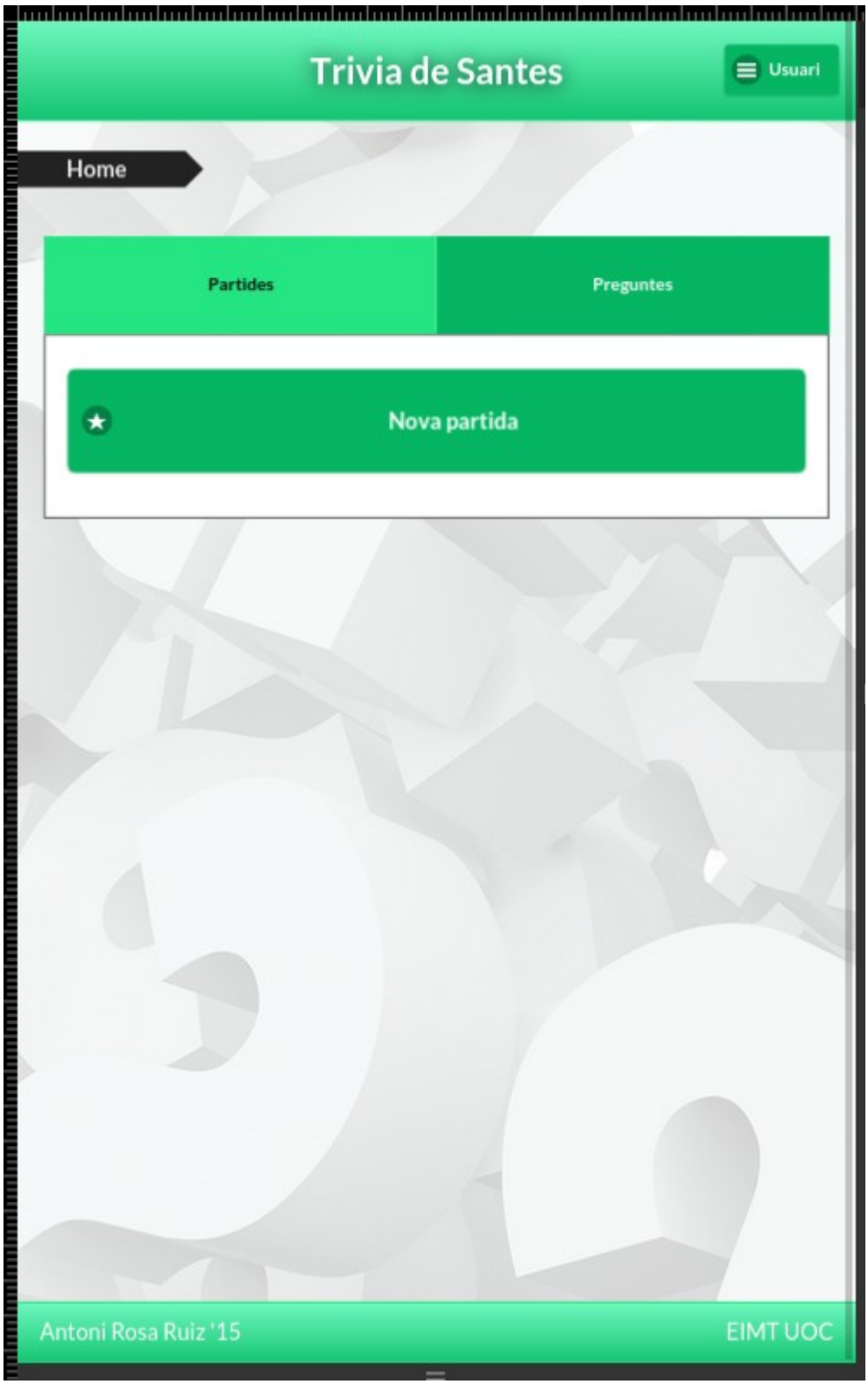
Drawing 11: Captura login

B- Pantalla de registro



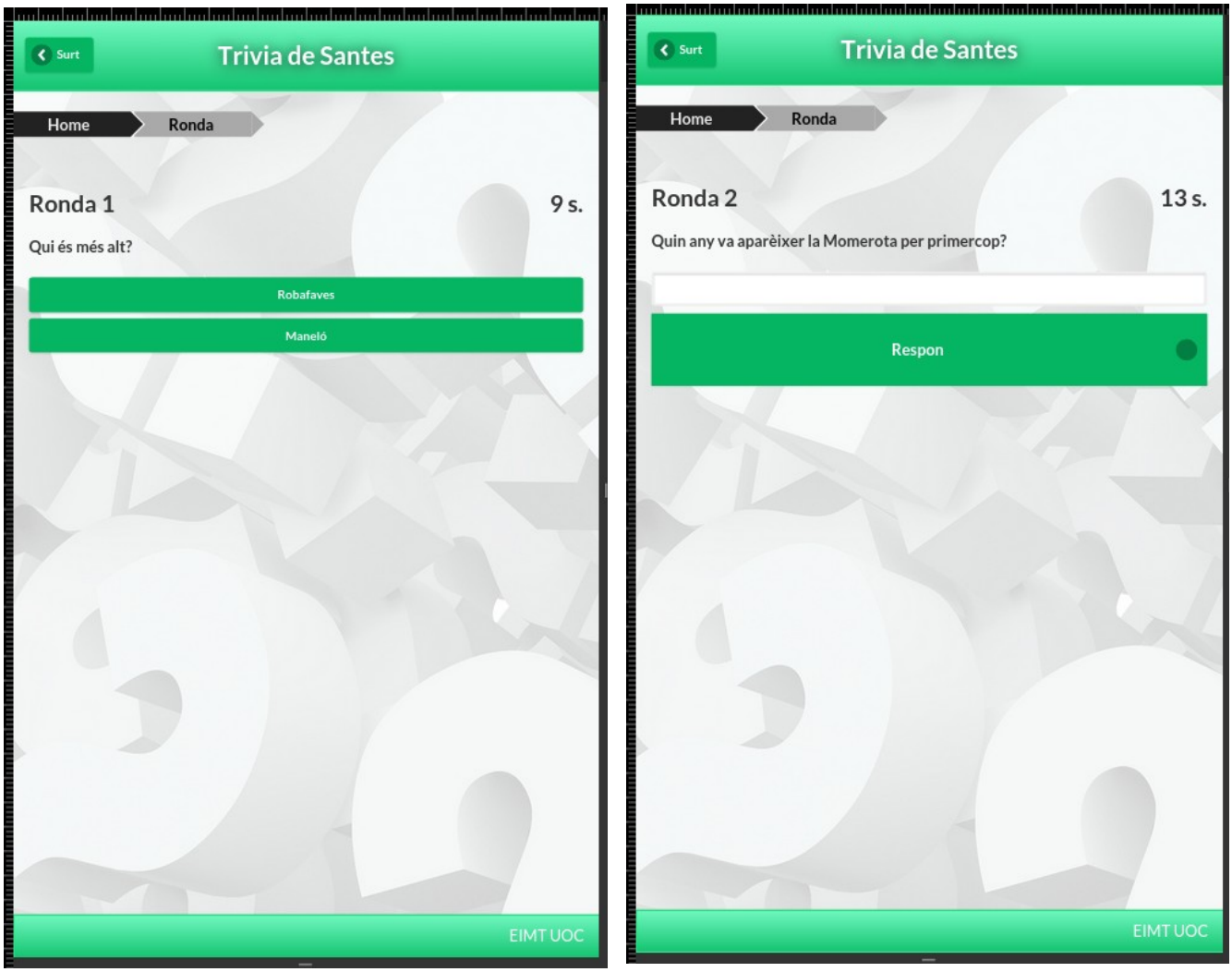
Drawing 12: Captura registro

C - Pantalla principal



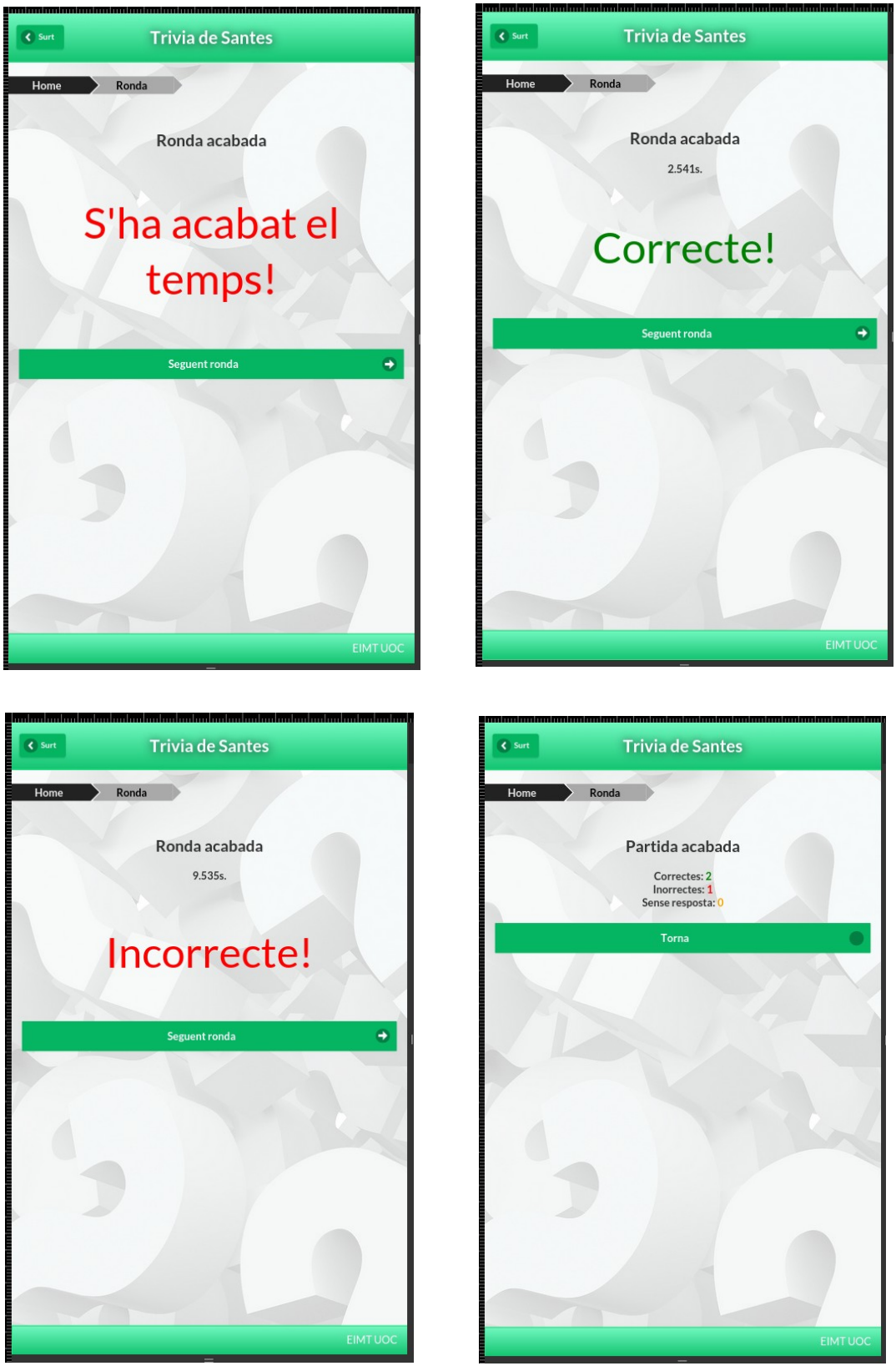
Drawing 13: Captura principal

D - Pantalla de juego: Múltiples opciones y Respuesta directa



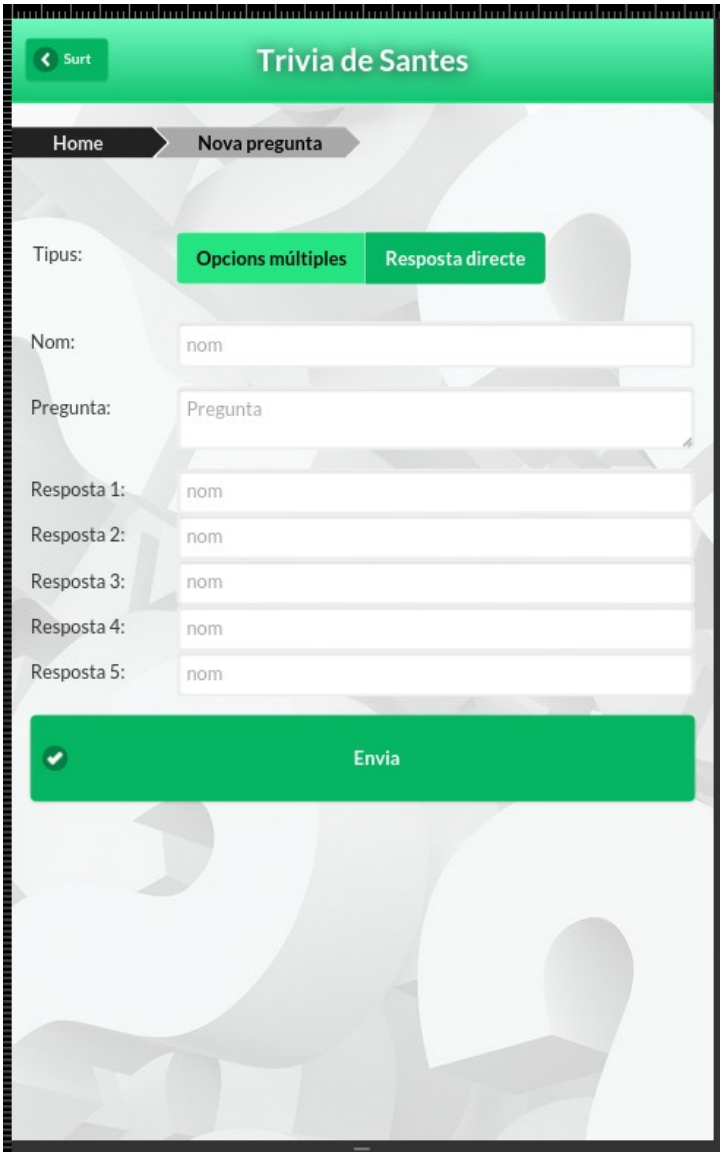
Drawing 14: Capturas juego individual

E - Pantalla de juego: resultados de la ronda y resultado final

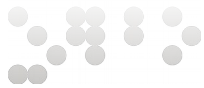


Drawing 15: Capturas resultados rondas

F - Pantalla de alta de una pregunta



Drawing 16: Capturas nueva pregunta



7. Conclusiones

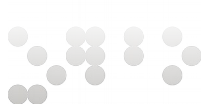
Este capítulo presenta las conclusiones del proyecto de final de postgrado.

La intención inicial de desarrollar un juego individual y multijugador, con sus dos partes aplicación cliente móvil y *backend* servidor se ha cumplido parcialmente. La carga de trabajo de había subestimado en el apartado de desarrollo y sólo se ha podido implementar parte de la funcionalidad planificada. Se han obtenido dos productos funcionales y, en este sentido, se ha validado la aproximación técnica al desarrollo, pero el juego multijugador no es aun funcional.
Se ha visto que gracias a una capa API rest y el intercambio de mensajes JSON es posible implementar un mecanismo directo y no muy complicado de intercambio de datos entre la capa servidora y la cliente. La parte cliente, gracias al uso de Phonegap, puede considerarse una aplicación multi dispositivo

A pesar de las derivas y la limitación temporal del proyecto que no han permitido dotar la aplicación de todas las funcionalidades previstas, se ha focalizado en obtener una solución genérica y ampliable que cumple con su objetivo básico: implementar un juego, o la base del mismo, de preguntas y respuestas usando preguntas entradas por parte de los propios usuarios jugadores.

Así mismo, con los productos obtenidos y la arquitectura desarrollada se ha puesto la base para una posterior finalización de la aplicación según el plan inicial y posteriores ampliaciones. Esto permitirá que la parte cliente pueda convertirse en un producto potencialmente publicable que ayude a difundir las festividades locales.

Una dirección futura que se podría estudiar es el uso de una librería JavaScript para la creación de efectos visuales y sonoros, como CreateJS, para dotar a la aplicación de más atractivo visual. Otra dirección futura interesante sería implementar la aplicación backend sobre algún sistema de *Cloud*: Google Apps, Heroku, etc...



8. Glosario

App: Versión reducida de *application* (aplicación en inglés) comúnmente usada haciendo referencia a aplicaciones para dispositivos móviles.

REST: Del inglés *Representational State Transfer* hace referencia a una arquitectura de software para sistemas distribuidos basados en hipermèdia, como por ejemplo el web. Este término fue introducido en 2000 en la tesis doctoral de Roy Thomas Fielding, uno de los autores principales de las especificaciones del protocolo HTTP.. A pesar de que en un principio RISTRA se refería tan sólo a un conjunto de principios de arquitectura de red y la definición y direccionamiento de los recursos, actualmente este concepto se utiliza para referirse a una interfaz web que utiliza XML o JSON y HTTP sin ningún conjunto de cabeceras como podría ser en el caso de SOAP o XML-RPC.

MVC: Es el acrónimo de Modelo-Vista-Controlador y hace referenia a un patrón de arquitectura desarrollo. Divide una aplicación en tres partes interconectadas, a fin de separar (1) las representaciones internas de información de (2) las formas en que la información se presenta o (3) es usada

API: Del inglés *Application Programming Interface*, hace referencia a un conjunto de funciones y procedimientos que permiten acceder a las características o los datos de un sistema operativo, aplicaciones, u otro servicio.

CSS: Del inglés *Cascading Style Sheets*, es un lenguaje de hojas de estilo usado para describir el aspecto y formato de cualquier documento escrito en lenguaje de marcas, normalmente páginas o aplicaciones web.

UML: Del inglés *Unified Modeling Language*, es un lenguaje unificado de modelado de propósito general en el campo de la ingeniería de software. Está diseñado para proporcionar una forma estándar para visualizar el diseño técnico de un sistema. Inicialmente fue creado y desarrollado por Grady Booch, Ivar Jacobson y James Rumbaugh en la empresa Rational Software durante los años 1994, 1995 y 1996,

9. Bibliografía

Aparte de la lectura de los materiales recomendados en el enunciado de esta primera entrega y disponibles en el apartado “Materiales y fuentes” del aula, para centrar la propuesta de este TFP se han consultado los siguientes recursos:

Teoria sobre estrategias *gamificación*:

Nicholson, Scott. "A user-centered theoretical framework for meaningful gamification." *Games+ Learning+ Society* 8.1 (2012).

Djaouti, Damien, Julian Alvarez, and Jean-Pierre Jessel. "Can Gaming 2.0 help design Serious Games?: a comparative study." *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*. ACM, 2010.

Información sobre algoritmos de métricas de reputación

Galli, R. “Cálculo comparativo de la diversidad de votos mediante densidad de grafos”, *Ricardo Galli, de software,* 2012. https://gallir.wordpress.com/2012/11/04/calculo-comparativo-de-la-diversidad-de-votos-mediante-densidad-de-grafos/

Andreas Kaltenbrunner et al., “Comparative analysis of articulated and behavioural social networks in a social news sharing website” *New Review of Hypermedia and Multimedia,* 2011. http://www.tandfonline.com/doi/pdf/10.1080/13614568.2011.598192.

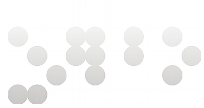
Información sobre las fiestas de Mataró

LesSantes.cat, Página oficial del ayuntamiento de la fiesta. Mataró [consultada por última vez el 9 de marzo de 2015]. Disponible en: http:www.lessantes.cat

LesSantes.net, Página con información de la fiesta. Mataró [consultada por última vez el 9 de marzo de 2015]. Disponible en: http://www.lessantes.net

Publicar un apk en Google Play

GooglePlay publish help page, Página de ayuda al desarrollador de Google Play. [consultada por última vez el 20 de mayo de 2015]. Disponible en: https://play.google.com/apps/publish



Anexos

1 Preparación del entorno

Conectar un dispositivo para desarrollar

Una vez conectado el dispositivo al puerto USB, para mirar los dispositivos reconocidos usaremos el comando adb del sdk de android.

```
#adb devices
List of devices attached
```

Si no reconoce ningún dispositivo, como es el caso por defecto de una instalación Ubuntu 12.04 LTS, los pasos para reconocer el dispositivo real son:

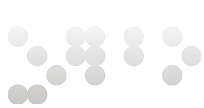
1. Añadir las reglas a udev

#sudo nano /etc/udev/rules.d/51-android.rules

Añadiremos la línea siguiente poniendo en XXXXX el código del fabricante (o "Vendor" en inglés) del dispositivo:

SUBSYSTEM==”usb”, SYSFS{idVendor}==”XXXXX", MODE=”0666", GROUP="plugdev"

Company	USB Vendor ID
1. Acer	2. 0502
3. ASUS	4. 0b05
5. Dell	6. 413c
7. Foxconn	8. 0489
9. Fujitsu	10. 04c5
11. Fujitsu Toshiba	12. 04c5
13. Garmin-Asus	14. 091e
15. Google	16. 18d1
17. Haier	18. 201E
19. Hisense	20. 109b
21. HTC	22. 0bb4
23. Huawei	24. 12d1
25. Intel	26. 8087
27. K-Touch	28. 24e3
29. KT Tech	30. 2116
31. Kyocera	32. 0482
33. Lenovo	34. 17ef
35. LG	36. 1004
37. Motorola	38. 22b8
39. MTK	40. 0e8d
41. NEC	42. 0409
43. Nook	44. 2080
45. Nvidia	46. 0955
47. OTGV	48. 2257
49. Pantech	50. 10a9
51. Pegatron	52. 1d4d
53. Philips	54. 0471
55. PMC-Sierra	56. 04da
57. Qualcomm	58. 05c6
59. SK Telesys	60. 1f53



61. Samsung	62. 04e8	Communications	
63. Sharp	64. 04dd		
65. Sony	66. 054c	71. Teleepoch	72. 2340
67. Sony Ericsson	68. 0fce	73. Toshiba	74. 0930
69. Sony Mobile	70. 0fce	75. ZTE	76. 19d2

Base de datos de "Vendors"
Fuente: http://developer.android.com/tools/device.html

2. Reiniciar el servicio "udev"

#sudo udevadm control –reload-rules

#sudo service udev restart

3. El servicio adb debe ejecutarse como usuario root ya que si no, no se reconoce bien el dispositivo. Para conseguirlo:

Paramos el servicio:
#adb kill-server

Lo ejecutamos con el comando sudo:
#sudo ./adb start-server

Resumen de Comandos habituales adb y cordova

Crear un nuevo proyecto Phonegap:

```
# cordova create directori app.companyia.com App
```

o más corto:

```
# cordova create xxxxx
```

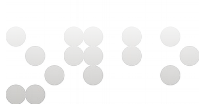
Añadirle una plataforma *target, por ejemplo Android:*

```
# cordova platform add android
```

Añadirle un plugin (p.e. el de Google analytics):

```
# cordova plugin add https://github.com/phonegap-build/GAPlugin.git
```

o la camera:



```
# cordova plugin add org.apache.cordova.camera
```

Ejecutar el proyecto en el dispoisitivo:

```
# cordova run android
```

Generar un apk:

```
# cordova build android
```

Generar un apk firmado (publicable en Google Play):

```
# cordova build android -release
```

En este caso, dentro 'platforms/android/ant.properties' deberían estar informados estos datos:

```
key.store=/Path/to/KeyStore/myapp-release-key.keystore
key.alias=myapp
```

Instalar un apk en un dispositivo:

```
# adb install -r app.apk
```

Ver en directo el log de un dispositivo:

```
# adb logcat
```

Acceder a la consola de un dispositivo:

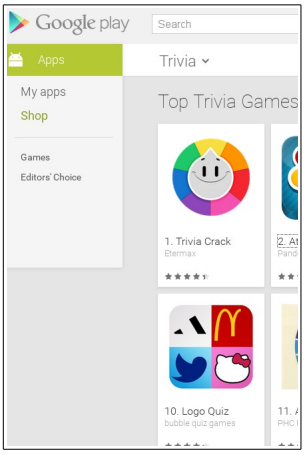
```
# adb logcat
```

2 Revisión de las categorías en los *markets* para las categorías de la app

iTunes

En el App store iTunes de Apple la app estaría en la categoría de «Juegos» y la subclasificación «Trivia» junto a juegos populares como: Preguntados, Atriviate, ¿Qué sabes de...Coches?

https://itunes.apple.com/es/genre/ios-juegos-trivia/id7018?mt=8



Google Play

La app entraría en la categoría de «Juegos» y la subclasificación «Trivia» de la tienda Play de Google.

Parece que hay juegos comparables (p.e. Atriviate) clasificados en diferentes subclassificaciones: «Puzles» y «Cultura general» en las que la app propuesta también podría encajar

https://play.google.com/store/apps/category/GAME_TRIVIA/collection/topselling_free

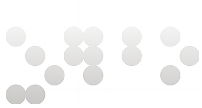
Windows phone

En el *store* para la plataforma de Microsoft la categoría para la app sería «Juegos» y la subcategoria «Puzles y curiosidades»

http://www.windowsphone.com/es-es/store/xbox-games/puzles-y-curiosidades/puzzleandtrivia

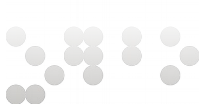
3 Scripts de creación de las tablas

```
CREATE TABLE IF NOT EXISTS player
(
  id SERIAL NOT NULL,
  email VARCHAR(250),
  name VARCHAR(200),
  password VARCHAR(100),
  lastaccess timestamp with time zone,
  created_on timestamp with time zone DEFAULT now(),
  active boolean DEFAULT true,
  CONSTRAINT pk_player PRIMARY KEY (id),
  CONSTRAINT unique_email UNIQUE (email)
);


CREATE TABLE IF NOT EXISTS question
(
  id serial NOT NULL,
  type VARCHAR(3),
  name VARCHAR(200),
  text TEXT,
  times_used integer,
  best_time integer,
  best_time_id_player integer,
  avg_time integer,
  percent_correct integer,
  accepted_on date,
```



```
  accepted boolean DEFAULT false,
  id_player_creator integer,
  created_on timestamp with time zone DEFAULT now(),
  CONSTRAINT pk_question PRIMARY KEY (id),
  foreign key (id_player_creator) references player(id) ON DELETE SET null
);

CREATE TABLE IF NOT EXISTS question_detail
(
  id serial NOT NULL,
  id_question INTEGER,
  response_text VARCHAR(512),
  is_correct boolean DEFAULT false,
  CONSTRAINT pk_question_detail PRIMARY KEY (id),
    CONSTRAINT fk_question_detail_question FOREIGN KEY (id_question) REFERENCES question(id)
);


CREATE TABLE IF NOT EXISTS friends
(
  id_player1 integer,
  id_player2 integer,
  created_on timestamp with time zone DEFAULT now(),
  CONSTRAINT pk_friends PRIMARY KEY (id_player1, id_player2),
  foreign key (id_player1) references player(id) ON DELETE CASCADE,
  foreign key (id_player2) references player(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS match (
```

```
  id SERIAL NOT NULL,
  id_player_main integer,
  name VARCHAR(200),
  state VARCHAR(100),
  created_on timestamp with time zone DEFAULT now(),
  CONSTRAINT pk_match PRIMARY KEY (id),
  foreign key (id_player_main) references player(id) ON DELETE SET NULL
);

CREATE TABLE IF NOT EXISTS match_question (
  id SERIAL NOT NULL,
  id_match VARCHAR(200),
  id_question VARCHAR(200),
  round index,
  CONSTRAINT pk_match_question PRIMARY KEY (id),
  FOREIGN KEY  (id_match) references player(id) ON DELETE CASCADE,
  FOREIGN KEY (id_question) REFERENCES question(id)
);

CREATE TABLE IF NOT EXISTS match_participant (
  id SERIAL NOT NULL,
  id_match integer,
  id_player integer,
  is_creator BOOLEAN DEFAULT false,
  state VARCHAR(100),
  joined_on timestamp with time zone DEFAULT now(),
  CONSTRAINT pk_match_participant PRIMARY KEY (id),
  foreign key (id_player) references player(id) ON DELETE CASCADE,
  foreign key (id_match) references player(id) ON DELETE CASCADE
);
```



```
CREATE TABLE IF NOT EXISTS message (
  id SERIAL NOT NULL,
  id_player_from integer,
  id_player_to integer,
  code VARCHAR(100),
  param VARCHAR(100),
  message VARCHAR(1024),
  created_on timestamp with time zone DEFAULT now(),
  CONSTRAINT pk_message PRIMARY KEY (id),
  foreign key (id_player_from) references player(id) ON DELETE CASCADE,
  foreign key (id_player_to) references player(id) ON DELETE CASCADE
);
```

4 Archivo de configuración de apFramwork

En el archivo de configuración, situado en la carpeta “config/” hay que personalizar los datos de la base de datos cargada. Un ejemplo si la base de datos se encuentre en el mismo servidor de Apache y PHP y la Base de datos la hemos llamado “trivia” y el usuario de acceso queremos que sea “postgres” con password “passpostgres”:

```
public static $server = 'localhost';
public static $db = 'trivia';
public static $user = 'postgres';
public static $pass = 'passpostgres';
```

5 Para la generación de unl APK

Se debería cambiar el del end-point a un nombre de mominio y no a una IP privada tal y con está actualmente en el archivo utils.js:

```
var config = {
        urlBase:"http://192.168.1.10/trivia_santes/"
};
```

6 La aplicación cliente en una tablet Nexus 10