DS

Desarrollo de Software

Práctica 3

Autor: Javier Lorenzo García
Autor: Pedro Serrano Pérez

Profesor: María del Mad Abad Grau

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Curso 2019 - 2020

Índice general

Capítulo 1

Descripción - La Ruleta de las Palabras

Vamos a implementar un juego para dispositivos Android, basado en el clásico *Pasapalabra*. El juego consiste en acertar las veinticinco palabras, cada una de las cuales se corresponde con una letra (comienza por dicha letra) del rosco para la que se ofrece una definición. Los aciertos se reflejarán en las letras mediante el color verde, mientras que los fallos se mostrarán en color rojo. Las cuestiones no respondidas o "pasapalabra"se mostrarán en color azul.
El usuario tiene la opción de contestar a la definición proporcionada o de pasar palabra. Además se guarda el estado del rosco actual por si el usuario se desconecta, de esta forma puede retomar el juego actual en un futuro. Cuando el usuario decida pasar palabra, la siguiente palabra que deberá responder será una aleatoria entre la restantes.

Capítulo 2

Análisis de Requisitos

2.1. Requisitos Funcionales

Estos requisitos funcionales se realizan mediate la aplicación móvil, excepto el último de ellos (Consultar estadísticas usuario, que se realiza mediante un navegador web.

- Alta de un usuario.
- Iniciar sesión.
- Generar nueva ruleta de palabras.
- Continuar ruleta guardada.
- Introducir palabra.
- Pasar palabra.
- Rendirse.
- Ver resultados.
- Volver a menú principal.
- Cerrar sesión.
- Consultar estadísticas usuario (navegador web).

2.2. Requisitos no Funcionales

- Seguro: Se garantiza la protección contra ataques externos y la confidencialidad de datos personales.

- Fácil de usar: GUI amigable e intuitiva.
- Portabilidad: Diferentes funcionalidades para aplicación móvil y navegador web.
- Transparente al usuario: El cliente se comunicará con el servidor de forma transparente al usuario.

Capítulo 3

Partes interesadas e inquietudes

3.1. Partes interesadas

Listado de partes interesadas:

- Arquitecto: Javier Lorenzo García será el arquitecto software que se encargará de elaborar la D.A. y el supervisor del proyecto.
- Cliente.
- Desarrolladores: Javier Lorenzo García y Pedro Serrano Pérez estudiantes de ingeniería informática.
- Ingeniero de producción y mantenimiento: Pedro Serrano Pérez se encarga de dar soporte hardware a la aplicación y supervisa su mantenimiento.
- Técnico de pruebas: Los propios desarolladores se encargarán de las pruebas en la aplicación.
- Usuarios: Personas que compran la aplicación en Google Play. Los siguientes tres usuarios son conocidos que tendrán el papel de usuarios de la aplicación:
  - Paco Hérnandez (U1).
  - Ángel Muñoz (U2).
  - Silvia Hurtado (U3).

3.2. Inquietudes

3.2.1. Inquietudes del cliente

- Me gustaría poder ampliar el sistema con nuevas funcionalidades como modificar datos de un usuario.
- El juego debe ser lo más fluido posible, en cuanto al envío y recepción de datos del servidor. Los usuario no deben esperar respuestas del servidor de forma continua mientras juegan, para tener una buena experiencia utilizando la aplicación.

3.2.2. Inquietudes de los usuarios

- Paco Hérnandez (U1): Temo que la aplicación sea difícil de usar.
- Ángel Muñoz (U2): Espero que no sea necesaria una buena conexión a Internet.
- Silvia Hurtado (U3): Me gustaria poder guardar el progreso de mi partida.

Capítulo 4

Arquitectura del sistema

4.1. Elementos

- Servidor: Es servidor se encargará de recibir y responder a las peticiones de los clientes. Realizará consultas a la base de datos e insertará datos de los usuarios en esta.

- Base de datos: Base de datos MySQL que almacenará la información necesaria para la aplicación multiplataforma.

- Aplicación web. Aplicación web (móvil) que dispondrá de toda la funcionalidad de la aplicación multiplataforma menos la consulta de estadísiticas.

- Cliente web (navegador web). Cliente web que dispondrá de la funcionalidad para consultar las estadísticas de un usuario. No dispondrá del juego en sí.

4.2. Diagrama de componentes

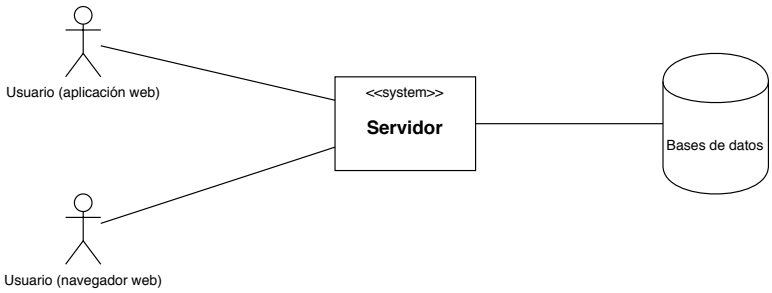


Figura 4.1: Diagrama UML de componentes

Capítulo 5

Listado de criterios de calidad (perspectivas)

5.1. Criterios de calidad

- Fácil de usar. La aplicación se ha diseñado para que sea intuitiva para cualquier usuario, haciendo uso de botones de texto que indican las acciones posibles que se pueden realizar y del botón de un nóvil para volver al menú anterior.
- Seguridad. Las contraseñas de los usuarios se guardan encriptadas en la base de datos y, además, el servidor está protegido ante antaques de inyección de código en la base de datos.
- Transparencia. El usuario desconoce en todo momento las comunicaciones que se están realizando entre el cliente (aplicación móvil) y el servidor.
- Evolución. El sistema admite cambios y la implementación de nuevas funcionalidades a un coste razonable.

Capítulo 6

Tecnología utilizadas

6.1. Servidor

El servidor será desarrollado con **php** (conexión con la aplicación web) y con **node.js** (conexión con un cliente web - navegador).

6.2. Aplicación web

La aplicación web se va a desarrollar haciendo uso de **Android Studio** y de **peticiones http** para comunicarse con el servidor php.

6.3. Cliente web

Página web desarrollada haciendo uso de **html**, **css** y **Javascript**.

Capítulo 7

Diseño

En este punto se van a desarrollar los distintos aspectos relacionados con la fase de diseño del sistema (métodos utilizados y diagramas de clase). El sistema posee una arquitectura de cliente-servidor en el que hay dos clientes (una aplicación móvil y un navegador web) que se comúnican, cada uno, con una parte del servidor, tal y como se han mencionado en el capítulo anterior de Tecnologías utilizadas.

7.1. Aplicación móvil

Como ya se ha mencionado anteriormente, para el desarrollo de la aplicación móvil se hará uso de Android Studio, por lo que esta tendrá un arquitectura que seguirá el patrón **Modelo Vista Controlador**.

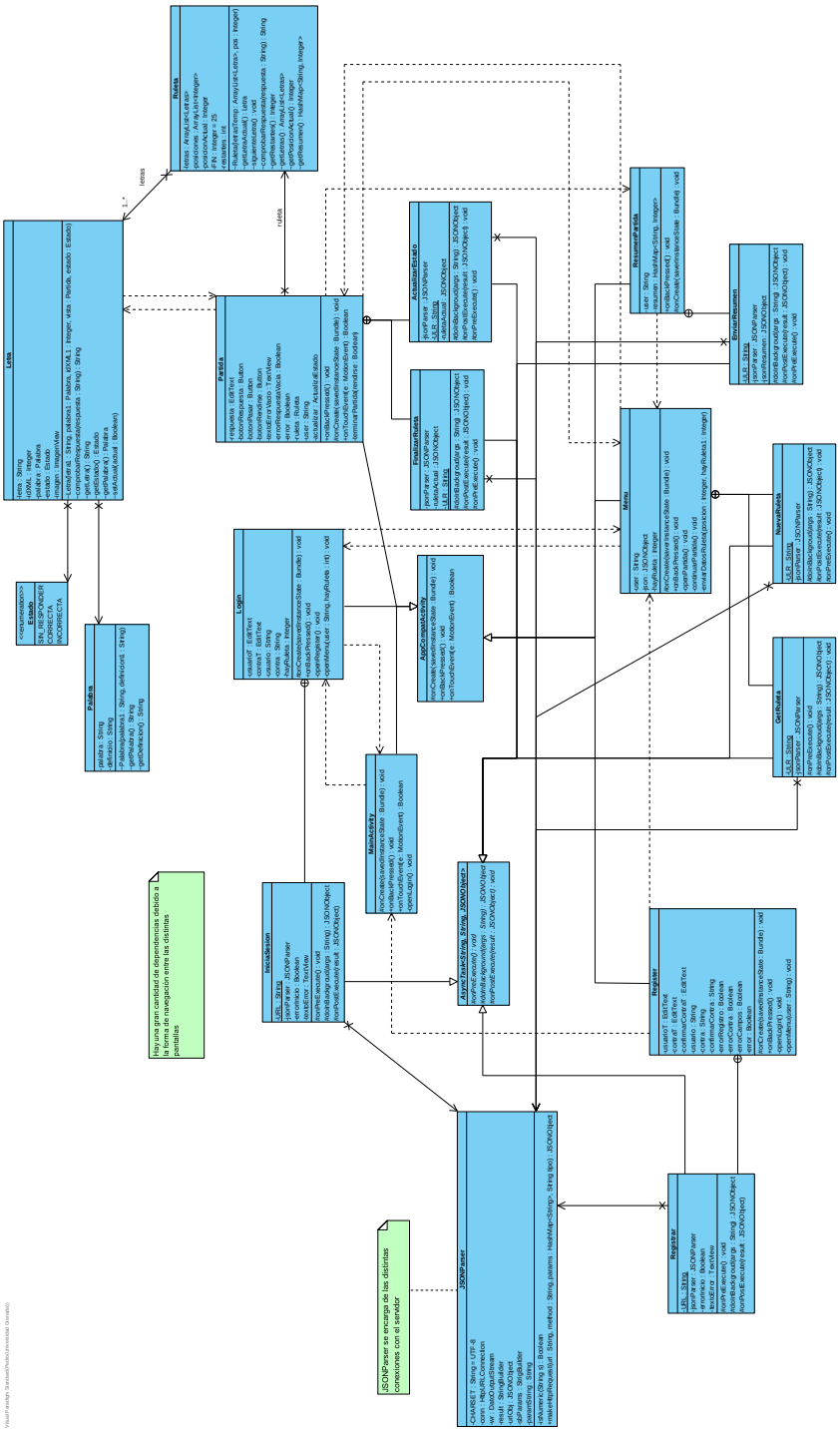
Para representar el diseño de la aplicación se utilizará un diagrama de clases UML que muestra sus operaciones.

7.1.1. Diagrama de clases de Aplicación móvil

En este diagrama de clases podemos ver la estructura de la aplicación y sus distintas funcionalidades.

Gracias al uso de Android Studio se puede implementar una arquitectura MVC que separa el modelo (ruleta) de la vista (diferentes pantallas que tiene la aplicación). Estas pantallas (vistas) son controladas mediante las clases que heredan de AppCompatActivity.

La clase JSONParses es la encargada de codificar la información que se va a enviar (en formato JSON), por tanto esta es la única clase que se comunica con el servidor. El resto de clases usan esta (de forma asíncrona, heredando de la clase abstracta AsyncTask).


Reglas
Letra
Estado
Palabra
Partida
Login
AsyncTaskString, String, JSONObject
JSONParser
Register
Menu
Ranking
RecuperarPartida
EnviarResultado
AnadirUsuario
PedirPalabra
GetPalabra
NuevaPalabra
DecideAccion
MainActivity
AppCompatActivity
letras
rutas
Hay una gran cantidad de dependencias debido a la forma de navegación entre las distintas pantallas
JSONParser se encarga de las distintas conexiones con el servidor

7.2. Cliente web

Se ha desarrollado un cliente web (accesible mediante un navegador web) que permite consultar las estadísticas de cualquier usuario registrado en el sistema.

Se trata de una página HTML que se comunicará mediante socket.io con el servidor, para obtener las estadísticas de un jugador

7.3. Servidor

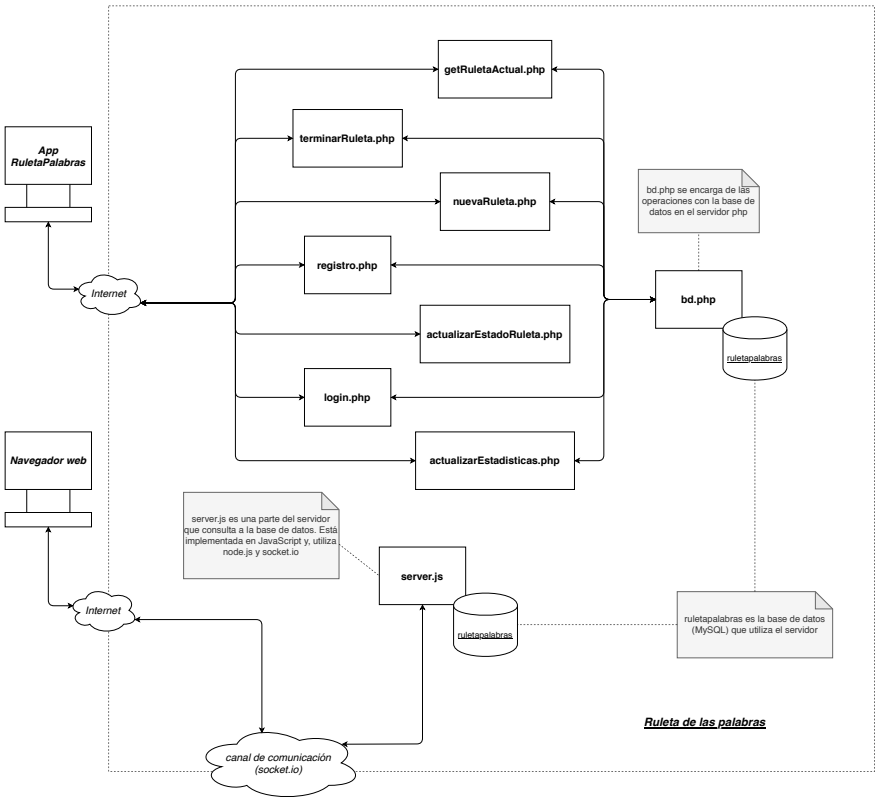
El servidor se divide en dos partes como se ha mencionado en el capítulo de tecnologías usadas. Por una parte, el servidor php, será utilizado por la aplicación móvil y, por otra, el servidor implementado mediante node.js, será utilizado por el cliente web.

Para representar su estructura y diseño se ha utilizado un diagrama de Cuadros y Líneas.

7.3.1. Diagrama de cuadros y líneas del servidor

En este diagrama se han representado los distintos clientes (web y móvil) y su comunicación con el servidor (ficheros php y Javascript (mediante node.js)).

La comunicación del navegador web con el servidor node.js, debido a que se realiza mediante socket.io, es asíncrona, mientras que la comunicación de la aplicación es síncrona (teniendo en cuenta que se hace uso de la clase abstracta AsyncTask para realizar el proceso en segundo plano).


getRuletaActual.php
terminarRuleta.php
App RuletaPalabras
nuevaRuleta.php
bd.php se encarga de las operaciones con la base de datos en el servidor php
registro.php
Internet
bd.php
actualizarEstadoRuleta.php
ruletapalabras
login.php
Navegador web
actualizarEstadisticas.php
server.js es una parte del servidor que consulta a la base de datos. Está implementada en JavaScript y, utiliza node.js y socket.io
server.js
Internet
ruletapalabras
ruletapalabras es la base de datos (MySQL) que utiliza el servidor
Ruleta de las palabras
canal de comunicación (socket.io)

Capítulo 8

Desarrollo de la práctica

Durante el desarrollo de esta práctica hemos usado Android Studio, PHP, javascript, Node.js, html y css.

8.1. Servidor

El servidor cuenta con una base de datos (MySQL) en la que se guarda toda la información correspondiente a usuarios y ruleta (tanto palabras como ruletas guardadas por los distintos usuarios que podrán reanudar en un futuro). Además funciona como servidor web que se ha desarrollado haciendo uso de Node.js.
Para poder servir la información a la aplicación móvil cuenta con diferentes archivos PHP que gestionan las distintas peticiones HTTP que puede recibir.

Cuando se recibe una petición de la aplicación móvil, un fichero PHP es el encargado de realizar la tarea correspondiente. Hay distintos ficheros PHP para cada petición que pueda solicitar la apliación. Estos ficheros hacen uso del fichero bd.php (el encargado de comunicarse con la base de datos. Además se usan las consultas preparadas para evitar inyección de código.

Para el caso del servidor nodejs, además del paquete de socket.io se hace uso de los paquetes **mysql** y **mysql-events**. El paquete mysql permitirá crear una conexión a una base de datos mysql y realizar consultas, modificar datos, etc. El paquete mysql-events se utiliza para crear un disparador, una vez un cliente se conecta al servidor (realza una búsqueda de un usuario), que se activará cuando se detecte un cambio en la tabla estadísticas de la base de datos ruletapalabras (leyendo los archivos binarios de la base de datos) y, este volverá a mandar la información a los clientes si el nombre de usuario de la fila modificada coincide con el que buscaron los clientes, de forma que la información que observan los cliente se actualiza de forma automática con los cambios que se producen en la tabla de la base de datos.

8.2. Cliente web

Los usuarios web pueden hacer uso de un cliente web en el que pueden consultar las estadísticas de los usuarios. El envío y recepción de datos se realiza mediante socket.io.

8.3. Aplicación móvil

La aplicación móvil se ha desarrollado en Android Studio. Todas las comunicaciones con el servidor se realizan mediante peticiones HTTP síncrondas (dejándolas en segundo plando mediante la clase abstracta AsyncTask). Toda la información que se usa en las comunicaciones tiene formato JSON. El estado actual de la ruleta se envía al servidor de forma automática cada vez que se responde una letra o pasa palabra, pero el usuario no necesita esperar por la petición para continuar jugando. Una vez se finaliza una ruleta se actualiza la base de datos para eliminar la ruleta guardada y actualizar las estadísticas del usuario.

Guía de uso de la aplicación:

- La navegación al menú anterior en la jerarquía es posible haciendo uso del botón *Atrás* del móvil.

- Una vez se pulsa en la pantalla inicial se puede iniciar sesión o registrarse en el sistema. Algunos errores, como campos vacíos, no confirmación de la contraseña, se comprueban en el cliente antes de lanzar la petición al servidor. Si ocurre algún error como contraseña o usuario incorrectos el servidor lo notificará al cliente y este avisará el usuario.

- Una vez el usuario ha iniciado sesión podrá generar una nueva ruleta (en este caso se guardaría la nueva ruleta) o continuar una guardada (si la tiene).

- Una vez en el juego podrá contestar una palabra, pasarla (en este caso la siguiente palabra a contestar será aleatoria entre las restantes) o rendirse.

- Una vez el usuario completa la ruleta o se rinde es dirigido a una página con las estadísticas de la partida. Se borra la actual ruleta guardada y se actualizan las estadísticas. Si se vuelve atrás en este punto el usuario es dirigido al menú para generar una nueva ruleta.

8.4. Errores y dificultades

- La principal dificultad que se ha encontrado durante el desarrollo ha sido manejar correctamente las peticiones cliente-servidor en la aplicación móvil. El envío y recepción de datos se realiza utilizando el formato JSON, el cual es muy estricto en cuanto al contenido. Además, no hay un conversor a JSON definido para objetos complejos en Android o Java, por lo que ha sido necesario implementar el código que pase un objeto (en este caso HashMap) a formato JSON y enviarlo, por lo que la mayoría de errores se han debido al contenido de los distintos JSON que se enviaban.

- Durante la realización de pruebas en la aplicación de Android observamos que en algunos casos la hebra principal finalizaba debido a algunas esperas del resto de hebras (que heredan de la clase abstracta AsyncTask). Este error lo solucionamos haciendo uso del método **get()** de AsyncTask, que permite obtener el resultado del proceso que se ha ejecutado y, por tanto se espera de forma adecuada por este.

- Al montar el servidor web en node.js. Cuando se realizaba una petición a dicho servidor, había un problema ya que trataba de buscar el favicon y no lo encontraba. Para solucionarlo tuvimos que descargar un favicon y colocarlo en el servidor.

- Ya que queríamos hacer uso de nodejs para la realización de un cliente web, queríamos aprovechar una de sus principales ventajas, el envío de información al un cliente de forma asíncrona, para actualizar la información del cliente si se producía algún cambio en esta. Para esto decidimos utilizar el paquete mysql-events de nodejs que, como hemos descrito antes, permite implementar disparadores para eventos que ocurran en las tablas (actualizaciones en nuestro caso). Los problemas que han surgido con este paquete vienen del escaso mantenimiento que tiene este, por lo que ha sido complicado encontrar las funciones, y su correcta aplicación, para que el disparador funcione de forma adecuada. Además, tras varias pruebas fallidas durante la implementación del disparador, finalmente encontramos que los archivos binarios de la base de datos deben ser accesibles.