UNIVERSIDAD DE GRANADA

Práctica 3

Desarrollo de Software
Grado en Ingeniería Informática
Curso 2019/2020

Francisco José Molina Sánchez
franmolsan@correo.ugr.es

Miguel Ángel Molina Jordán
miguel99@correo.ugr.es

Escuela Técnica Superior de
Ingenierías Informática y de Telecomunicación

Índice

1. Análisis

En este apartado procederemos a detallar la fase de análisis del sistema mediante una descripción arquitectónica (DA).

1.1. Requisitos funcionales

1. El juego debe tener un sistema para que el usuario pueda iniciar sesión o registrarse.
2. Se debe poder almacenar la puntuación de cada partida, de forma que se puedan consultar las puntuaciones de otros jugadores (ranking).
3. El personaje se podrá mover hacia arriba, abajo, izquierda y derecha.
4. Los proyectiles podrán aparecer desde todas las direcciones y aleatoriamente.
5. Si un proyectil golpea al personaje, se acabará la partida y se volverá a la pantalla de inicio.
6. El personaje dispone de una habilidad (que se carga periódicamente) para ayudarle a sobrevivir.

1.2. Requisitos no funcionales

1. La aplicación tendrá unos controles sencillos, para que pueda jugarse desde dispositivos móviles.
2. La parte del cliente se desarrollará usando Phaser.
3. La parte del servidor se encargará de la autenticación y gestión de los usuarios, y los datos de puntuaciones de los usuarios (ranking).
4. La parte del servidor usará Node.js, Express y MongoDB.

1.3. Listado y descripción de las partes interesadas en el sistema

- Arquitectos: Miguel Ángel Molina Jordán y Francisco José Molina Sánchez (graduados en Ingeniería informática) son los arquitectos software encargados de elaborar la D.A.
- Cliente (adquiriente)
- Desarrolladores: María Pérez (graduada en Ingeniería informática) es la encargada de desarrollar el proyecto.
- Ingeniero de producción y mantenimiento y webmaster: Pedro Martínez (graduado en ingeniería informática) será el encargado de dar soporte al sistema y supervisar su mantenimiento.
- Administrador del sistema: Antonio Rodríguez es el encargado de la administración del sistema, debe estar a cargo del mantenimiento
- Técnico de pruebas: Federico López es el encargado de realizar las pruebas en la aplicación que se está desarrollando.

- Usuarios:
  - Clientes del juego: Son los jugadores que utilizan nuestra aplicación

1.4. Inquietudes de los interesados

Una de las inquietudes que más preocupan a los interesados de este sistema es la seguridad, ya que no quieren que la información que guardamos en nuestra base de datos se filtre o pueda ser robada. Para solucionar este problema, tenemos que asegurarnos de que utilizaremos un cifrado seguro para encriptar dichos datos de forma que si nuestro sistema sufre un ataque los datos del usuario no estén expuestos.

2. Software utilizado

- Hemos usado Node.js para la creación de la API del servidor, con diferentes módulos:
  - *Express*, el framework principal de Node para la creación de APIs.
  - *JWT* (*json web tokens*): para la autenticación del usuarios. En la versión del navegador, los JWT se guardan en las cookies. En la versión de *cordova*, se guardan en el almacenamiento local del navegador.
  - *Passport.js* : para autenticar a los usuarios, junto con los JWT.
  - *Nodemailer*: para enviar correos a los usuarios para recuperar la contraseña.
  - *Bcrypt*: para encriptar las contraseñas que introduzcan los usuarios.
  - *Dotenv*: para cargar variables de entorno desde un archivo *.env*.
  - *Body Parser* y *Cookie Parser*: para procesar las peticiones que los usuarios realizen a la api y las cookies.
- Para la creación de la base de datos hemos usado MongoDB Atlas, que es una base de datos que se encuentra en la nube.
  - *Mongoose*: para ayudar en el manejo de la base de datos (MongoDB) con Node.js
- Para la creación del cliente, hemos utilizado varias herramientas:
  - Las páginas de iniciar sesión, registro y contraseña olvidada las hemos creado usando html y *Bootstrap* para el estilo.
  - Para el diseño del videojuego hemos usado *Phaser* (framework de JavaScript).
  - *Apache Cordova* para transformar el código en JavaScript en una aplicación Android, y hacer que funcione en dispositivos móviles.
  - *jQuery*, la biblioteca que utilizamos para realizar consultas al servidor,

3. Pasos seguidos

- En primer lugar, creamos la base de datos, en mongoDB Atlas. En ella se guardan las puntuaciones de los usuarios y sus credenciales, y la gestionaremos con mongoose, una librería que facilita la interacción entre mongoDB y Node.

- Después creamos el servidor usando Node.js. Definimos todas las rutas de la API (*end-points*): /login, /signup, /scores, etc. Manejan la información que reciben de la petición realizada por el cliente y la insertan/comparan con la información de la base de datos.
- Luego creamos unas páginas html sencillas (usando Bootstrap para el estilo). Mediante estas páginas, proporcionamos una interfaz para que los clientes se conecten a la API y puedan registrarse, inciar sesión, etc.
- Tras esto, empezamos a crear el juego. Lo primero que hicimos fue crear el menú principal donde se puede acceder al ranking (donde aparecen los 10 mejores) o jugar.
- Después creamos el ranking en el que se consultan las 10 mejores puntuaciones de las almacenadas en la base de datos (mediante una petición a la API) y se imprime por pantalla la puntuación y el nombre de la persona que hizo dicha puntuación.
- Para crear el juego, primero creamos el personaje que puedes manejar y le pusimos físicas, y le añadimos una habilidad que le hacía invulnerable. Luego creamos un método para generar asteroides que aparecen desde un borde aleatorio y en un punto también aleatorio. Si un asteroide colisiona con el jugador saldrá una pantalla de game over y se guardará la puntuación si es la máxima del usuario.
- El último paso ha sido adaptar el código que habíamos creado previamente para que funcione como aplicación Android, usando Apache Cordova. Modificamos los controles del programa original por botones para poder usarlos usando la pantalla táctil.

4. Diagramas

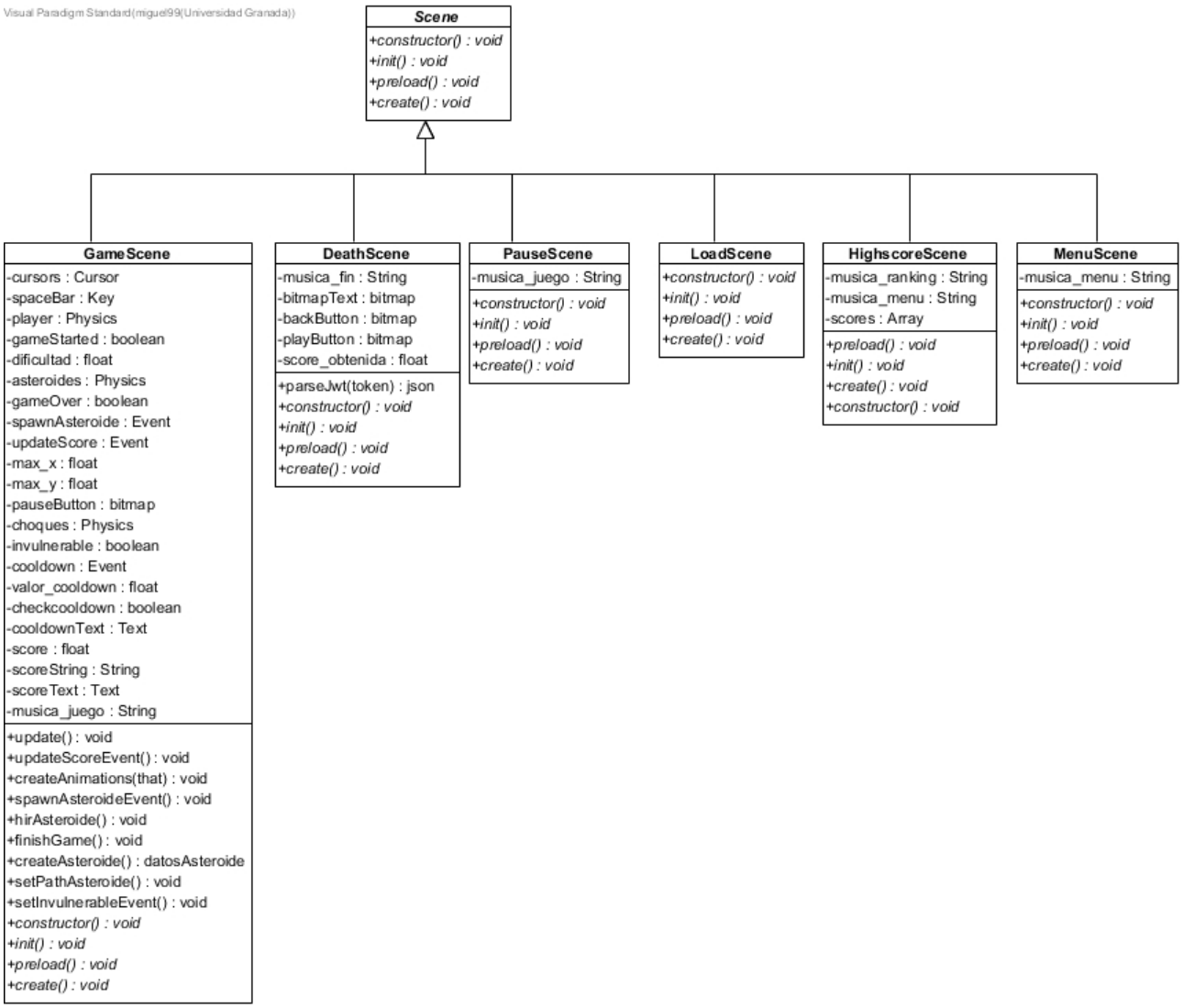

Figura 1: Diagrama de clases del juego

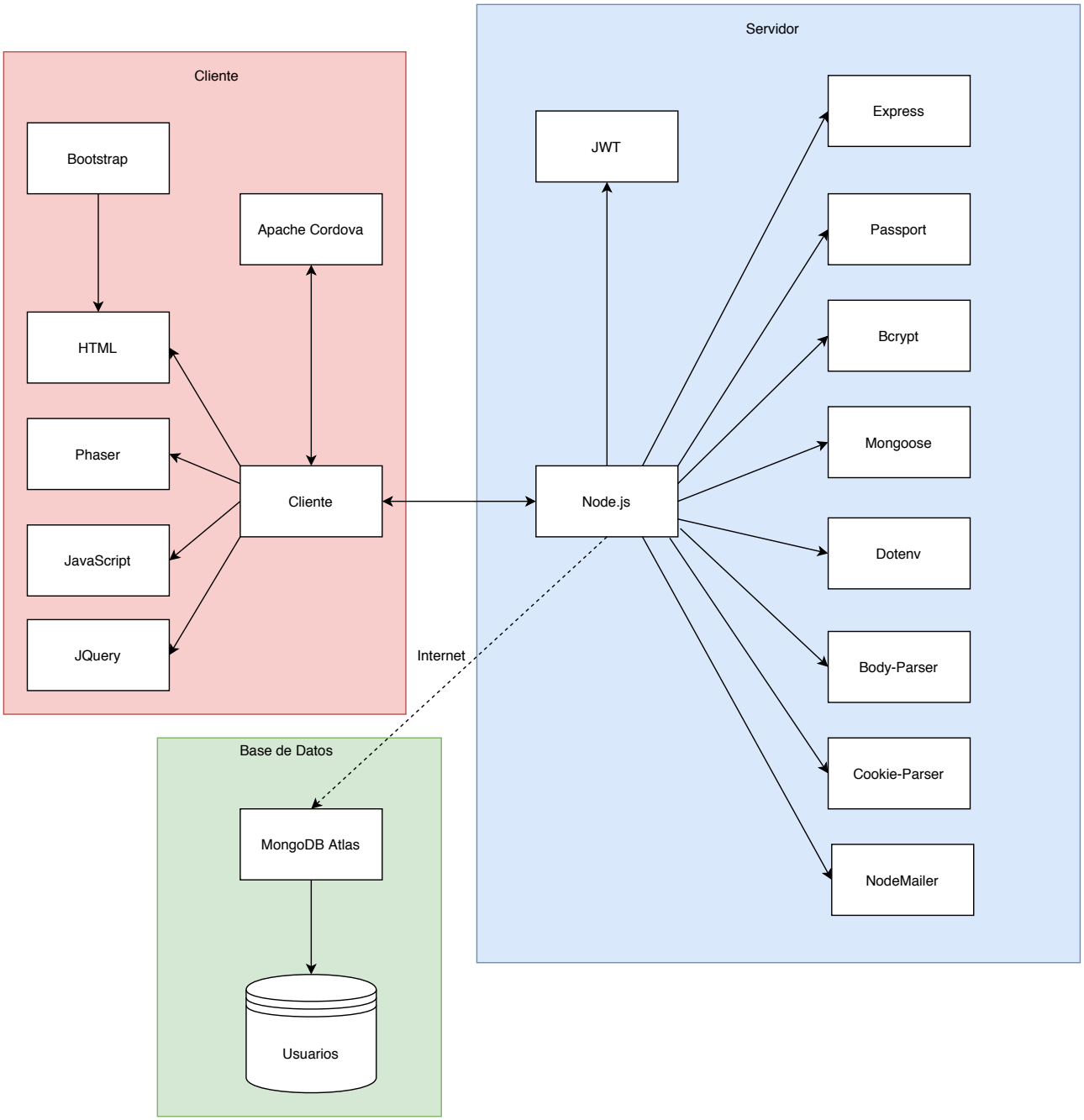

Figura 2: Arquitectura del sistema

5. Criterios de calidad

- La información almacenada sobre los jugadores es confidencial y no se podrá distribuir. Se ha de almacener con un cifrado lo suficientemente seguro.
- La apicación debe asegurar que la información almacenada de los jugadores no se pierda.
- La aplicación se irá manteniendo a lo largo del tiempo para que los jugadores igan teniendo una buena experiencia.
- Debe soportar cargas de usuarios hasta 10 veces más grandes de lo habitual.

6. Comentarios

La dificultad más grande que nos hemos encontrado en esta práctica ha sido conseguir que todas las tecnologías que hemos utilizado funcionen correctamente y puedan coordinarse entre sí. Por ejemplo, para la parte del servidor tuvimos algunos problemas con la autenticación del usuario (módulos passport y JWT). También encontramos errores relacionados con el módulo Nodemailer e incluso tuvimos que cambiar la versión de dicho módulo porque no era del todo compatible con nuestro proyecto.

La otra gran fuente de problemas ha sido la transformación del código para ejecutarlo en Android. Es una herramienta muy cómoda y que agiliza mucho el desarrolo (ya que permite generar el código para varias plataformas, incluyendo Android, iOS y navegadores web). Debido a esto, tiene muchas dependencias con Android, por lo que hemos tenido muchos errores relacionados con problemas de versiones de Android, políticas de seguridad, problemas de emulación e incluso problemas para procesar correctamente el código que ya teníamos (y funcionaba perfectamente) para el cliente web.

Como conclusión, creemos que esta práctica ha sido muy útil, ya que hemos aprendido a usar varias tecnologias nuevas (realmente hemos aprendido todas las tecnologías que hemos usado, no habíamos usado ninguna anteriormente). También hemos aprendido la importancia de gestionar las dependencias en un sistema heterogéneo como el nuestro, ya que un cambio en uno de los sistemas externos que usamos puede dejar nuestra aplicación inservible.

Asimismo hemos entendido la facilidad que proporcionan los frameworks, librerías y otras herramientas como *Apache Cordova* a la hora de desarrollar un sistema, ya que permiten agilizar y simplificar el proceso de desarrollo. Además si las herramientas son de calidad alta y se han creado con buenas prácticas de ingeniería de software, también es más fácil que el sistema que desarrollemos tenga la calidad y eficiencia que deseamos. Por ejemplo, el framework que hemos utilizado para desarrollar el juego (*Phaser*), utiliza muchos patrones de diseño internamente para gestionar y crear los objetos (factoría, factoría abstracta, *composite*, etc).