Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Aplicación Web para la visualización de las Olimpiadas (LiveTheOlympics)

Web Application for the visualization of the Olympics (LiveTheOlympics)

Ana de Lorenzo-Cáceres Luis

La Laguna, 10 de junio de 2019

D. **Vicente José Blanco Pérez**, con N.I.F. 42.171.808-C profesor Titular de Universidad adscrito al Departamento de Estadística, Investigación Operativa y Computación de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"*Aplicación Web para la visualización de las Olimpiadas (LiveTheOlympics)*"

ha sido realizada bajo su dirección por D. **Ana de Lorenzo-Cáceres Luis**, con N.I.F. 79.097.526-C.

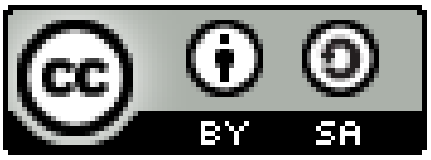
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de junio de 2019

Agradecimientos

En primer lugar agradecer a Vicente José Blanco Pérez por su ayuda, orientación y apoyo a la hora de realizar este proyecto ya que siempre estuvo pendiente de que todo fuera bien y avanzando y me ayudaba en cualquier duda que pudiera tener.

También agradecer a mis amigos, familia y compañeros que me han ayudado y apoyado durante estos meses de trabajo.

Licencia



Resumen

Este trabajo se ha basado en crear una innovadora herramienta que consiste en una aplicación web que facilite la visualización de los juegos olímpicos desde cualquier lugar, ya que esta plataforma hoy en día no existe teniendo en cuenta que es un evento multitudinario a nivel mundial.

La idea de este proyecto es permitir que el usuario disfrute de la visualización de las pruebas sin tener que depender de las televisiones públicas o plataformas de pago que las ofrecen. Es decir, que toda la información sobre estas este disponible en cualquier momento de forma gratuita y de primera mano de los organismos oficiales, desde cualquier terminal. Para poder ofrecer esta alternativa sin costo adicional al usuario, se ha planteado un modelo de negocio basado en añadir publicidad y conseguir financiación de la organización de las olimpiadas y sus patrocinadores.

Para llevar a cabo esta aplicación se han empleado diversas tecnologías, para el desarrollo de esta se ha utilizado el framework Vue CLI que trabaja con Webpack para realizar la parte de cliente y el despliegue se ha realizado de forma continua con Heroku añadiendo un servidor Node.js/Express, realizándose este despliegue mediante la integración continua con Travis CI.

Para el almacenamiento de los datos que ha de mostrar la aplicación se ha optado por el uso de Firestore de Firebase ya que permite delegar el desarrollo completo del backend a las plataformas de Google, lo que a la hora de realizar la aplicación, se consideró beneficioso ya que la cantidad de datos a almacenar no es demasiada. Además se han hecho uso de los recursos de Twitter y YouTube para mostrar la información de los juegos a los usuarios y los vídeos o directos que se retransmitan desde la cuenta oficial de las olimpiadas de YouTube.

Para el maquetado de la aplicación se ha hecho uso del framework Vuetify que combina los componentes de Vue.js con el aspecto de Material Design, lo que permite un desarrollo más fluido y un maquetado más rápido y eficiente.

Palabras clave: Juegos Olímpicos, Vue CLI, Firestore, Heroku, Travis CI, Vuetify, Twitter, Youtube

Abstract

This project has been based on creating an innovative tool. It consists of a web application that facilitates the visualization of the Olympic Games from anywhere. Since this platform does not exist today considering that it is a massive event worldwide.

The idea of this project is to allow the user to visualize the Olympics without having to depend on the public televisions or payment platforms. All the information about these is available at any time, free of charge and from any terminal. In order to offer this alternative without any cost, the business model is based on advertising and being financed by the organization of the Olympics and its sponsors.

To carry out this application, various technologies have been use. For the development, it has been used the Vue CLI framework that works with Webpack to perform the client. The deployment has been done continuously with Heroku adding a Node.js/Express server.It is carried out through continuous integration with Travis CI.

For the storage of the data of the application, it is using Firestore from Firebase. Since it allows delegating the full development of the back-end to the Google platforms, it was considered beneficial since the amount of data to be stored is not much. In addition, Twitter and YouTube resources have been used to show the information and the videos or directs that are broadcast from the official YouTube account of the Olympics.

For the layout of the application, the Vuetify framework has been used. It combines the components of Vue.js with the aspect of Material Design. This allows a more fluid development and a faster and more efficient layout.

Keywords: Olympic Games, Vue CLI, Firestore, Heroku, Travis CI, Vuetify, Twitter, Youtube

Índice general

Índice de Figuras

Índice de Tablas

Listings

Capítulo 1

Introducción

Este proyecto se ha desarrollado con el objetivo de crear una aplicación web para la visualización de los juego olímpicos. Esta mostrará la información que publiquen las cuentas oficiales de los juegos además de tener la información de los atletas que compiten en cada prueba.

1.1. Antecedentes y estado actual del tema

Los Juegos Olímpicos se han celebrado desde 1896 hasta la actualidad reuniendo a grandes masas en los estadios olímpicos por no mencionar todos los seguidores que no se pueden permitir desplazarse para verlos. Estos últimos recurriendo a la televisión, y hoy en día a Internet, para ver las pruebas.

Las próximas olimpiadas (2020) están empezando a tomar forma restando un año para su celebración, prevista su celebración entre el 24 de julio y el 9 de agosto. La retransmisión de estas asciende a nivel mundial siendo vista en todos los continentes. Tras una investigación se llegó a la conclusión de que estas aún hoy en día no poseen una plataforma oficial para su visualización. Estas se transmiten de forma online mediante plataformas como Youtube, las páginas web de las televisiones públicas o mediante las plataformas ofrecidas por las teleoperadoras como, por ejemplo, Movistar+, antiguamente Yomvi, perteneciente a la operadora Movistar.

Por ello, este proyecto desarrolla una plataforma novedosa que estaría al alcance de todo el mundo de forma gratuita, teniendo en su interior toda la información oficial de los juegos y la retransmisión de las pruebas agrupada en un solo lugar, de fácil acceso desde cualquier terminal para la comodidad del usuario.

La interfaz de la plataforma ha sido inspirada en la aplicación móvil creada tanto para IOS como para Android del programa de televisión Operación Triunfo. El uso que se le da a esta aplicación es para poder ver las emisiones de los programas a través de ellas y poder ver las descripciones de los concursantes y poder votarles. La idea es generar una aplicación con una interfaz similar a la creada por el programa pero utilizando tecnologías de desarrollo web.

La idea principal es generar una interfaz similar que se adapte al ámbito de las olimpiadas, para que se puedan ver las retransmisiones de las pruebas, las noticias relacionadas con estas y poder ver los perfiles de los atletas competidores en las diferentes pruebas. El proyecto se llevaría a cabo con la realización de una aplicación web que difiere del modelo de aplicación que creó el programa.

1.2. Características de la Aplicación

La aplicación creada permite la visualización de las pruebas olímpicas, ver los perfiles de los atletas y ver las noticias que cuelgan los organismos oficiales en Twitter. En próximos capítulos se describirán las funcionalidades en su totalidad.

Para conseguir este desarrollo se han empleado diferentes tecnologías, estas están listadas a continuación:

- Framework Vue CLI [4]
- Webpack [17]
- Framework Vuetify [16]
- Firestore by Firebase [7]
- Despliegue en Heroku [10]
- Integración continua con Travis CI [3]
- Youtube API [9]
- Componente Timeline de vue-tweet-embed [11]
- Express [6]

Todas estas tecnologías serán tratadas en profundidad en el capítulo 2 y más adelante se explicará cómo fueron empleadas en la aplicación.

1.3. Actividades a realizar

Este proyecto consistirá en la creación de una aplicación web mediante el uso de las tecnologías Firestore, Express, Vue CLI y Node.js con un acercamiento a la creación de *Progressive Web Apps* (PWAS) y uso de la estrategia de desarrollo *Mobile First* para la visualización de las próximas olimpiadas (2020). Se realizará con los datos de las olimpiadas pasadas (2016) y con la información que vayan subiendo los organismos oficiales a su Twitter y cuenta de Youtube para poder mostrar la simulación de cómo funciona.

Para la conclusión satisfactoria del proyecto se realizarán diferentes actividades, tales descritas a continuación:

Tarea	Descripción
Tarea 1	Desarrollo de prototipado con la herramienta Justinmind
Tarea 2	Crear un repositorio de Github para proporcionar control de versiones
Tarea 3	Añadir integración continua mediante Travis CI
Tarea 4	Añadir el despliegue continuo con Heroku a través de Travis CI
Tarea 5	Desarrollo de las vistas con el Framework Vue CLI
Tarea 6	Añadir la integración de los tweets
Tarea 7	Añadir la integración de la AP de Youtube
Tarea 8	Añadir la integración con Firestore a las vistas
Tarea 9	Adaptar la versión Desktop

Cuadro 1.1: Tareas descritas

Capítulo 2

Entorno de Desarrollo

En el capítulo pasado se han introducido los antecedentes y estado actual del tema, las características de la aplicación y las tareas a realizar para llevarla acabo.

Este capítulo se centrará en describir y analizar las tecnologías y el entorno de desarrollo de la aplicación. Al ser un poryecto greenfield, el programador tuvo la oportunidad de escoger su propio entorno de desarrollo.

2.1. Bases de la aplicación

LiveTheOlympics se desarrolló bajo en framework JavaScript Vue CLI, además de la incorporación del framework Vuetify para explotar los componentes que permite generar Vue con un estilo de Material Design. Para la parte de back-end se decidió migrar de la idea principal, que era generarlo con MondoDB, a Firestore de Firebase ya que la cantidad de información a almacenar era poca y se decidió delegar esto a la nube.

El proyecto se inició con PivotalTracker [13] como medio de seguimiento del desarrollo de este, aunque se hizo uso de este, resultó poco útil ya que al ser un solo programador no siempre se marcaban como finalizadas las tareas o se indicaba que se habían iniciado tareas nuevas. Si embargo, se hizo uso de Git para tener un control de versiones, usando un repositorio GitHub para almacenar estos cambios generados en cada realización de tareas.

El proyecto siguió la metodología XP o eXtreme Programming. Se realizó una planificación previa del proyecto, se continuó con una fase de diseño en la que se generó el Mock-Up de la aplicación y, por último, se comenzó con la programación.

2.2. Arquitectura Software

2.2.1. Patrón MVVM (Modelo - Vista, Vista - Modelo)

Este patrón [19] actualmente es muy utilizado en JavaScript, frameworks como Vue.js, React o Angular lo siguen. El gran atractivo de este se resume en la sencillez con la que se pueden pasar datos desde el código en JavaScript (Modelo) al código HTML (Vista).

- **Relación Modelo-Vista**

  Permite utilizar una cualidad interesante llamada data-bindig. Esta permite que cada vez que cambie el valor del modelo, Vue ordena que se vuelva a renderizar la vista con los valores nuevos y las variables que estén tomando ese valor cambien.

- **Relación Vista-Modelo**

  En esta relación se pasa la información desde la Vista al Modelo, un ejemplo puede ser un evento de click, en Vue se incluye la directiva v−on:click="function()" que es similar a la propiedad onclick de las etiquetas html.

2.2.2. Vue CLI

Vue CLI es un sistema para el desarrollo con el framework Vue.js. Como su nombre indica se trata de la interfaz de línea de comandos o command line interface. La principal misión de esta es ayudar a un rápido desarrollo de aplicaciones Vue.js.

Vue.js es un framework progresivo open-source que se puede integrar con otras librerías o proyectos pero que, además, permite desarrollar Aplicaciones Single-Page e interfaces de usuario si se combina con otras librerías que le dan soporte.

Este fue creado por Evan You, trabajador de Google que usaba AngularJS. Su idea para crearlo fue, extraer las partes de Angular que le gustaban y hacer un framework muy ligero. La primera versión de este salió en Febrero de 2014.

2.2.3. Webpack

El framework Vue CLI trabaja con Webpack, por ello se va a explicar esta tecnología. Webpack es un bundler que usan diferentes frameworks así como Vue.js, Angular y React también. Este es usado en aplicaciones tan famosas como Instagram o Twitter.

Además de ser un bundler puede convertir formatos, gestionar dependencias y ser un servidor de desarrollo entre otras características. Todo esto genera que sea una herramienta muy útil a la hora de desarrollar aplicaciones web modulares.

2.3. Herramientas de Desarrollo

Esta sección abarcará las herramienta de despliegue, funcionalidad, control de versiones, navegación en la interfaz y apariencia.

2.3.1. Despliegue: Heroku

El despliegue de la aplicación se ha decidido hacer mediante la herramienta Heroku a través de la integración continua de Travis CI. Esta herramienta inicialmente fue construida para soportar solo Ruby pero a lo largo del tiempo se ha extendido ha soportar otros como Node.js, el que usará para esta aplicación.

Esta herramienta permite obtener un nombre que esté disponible en el dominio herokuapp.com. No obstante, no solo se centra en el despliegue, algunos de sus productos ofrecen servicios de Cloud database, control de rendimiento y monitorización y métricas entre otros.

2.3.2. Funcionalidad

YouTube API

Para que los usuarios puedan ver la retransmisión se decidió incluir los vídeos de su canal oficial de YouTube en el que suben estas durante la celebración de los juegos. Esta API permite extraer los vídeos de los diferentes canales lo que nos permite añadir esta funcionalidad.

Twitter API - vue-tweet-embed

También se extraen los tweets de la cuenta oficial de Twitter para que los usuarios estén informados desde las fuentes oficiales al momento. Primero se usó la API de Twitter para añadir esta funcionalidad pero se migró a vue-tweet-embed una dependencia que esta soportada por la API widget de Twitter [14] que permite este contenido lo mismo pero de una manera más sencilla y ligera en el código.

Firestore

Por último, los usuarios podrán ver la información de los atletas de cada deporte de los juegos. Para ello, se ha decidido hacer uso de Firestore que proporciona una velocidad en las consultas mayor e incluye lo mejor de Realtime Database de Firebase siendo una base de datos NoSQL.

2.3.3. Control de versiones: GitHub

Para llevar el control de versiones de este proyecto se ha elegido crear un repositorio público en el software GitHub[8] ya que este almacena los proyectos utilizando la herramienta Git. Esta es la url al repositorio `https://github.com/alu0100972016/livetheolympics`.

2.3.4. Navegación por la interfaz: Vue Router [15]

Para generar el dinamismo entre las vistas de la aplicación y que el usuario pueda utilizar la interfaz de manera intuitiva y sencilla se ha utilizado el router oficial de Vue.js que hace que generar una Aplicación Single-Page sea rápido y sencillo.

2.3.5. Apariencia: Vuetify

Para darle forma a la aplicación se ha utilizado el framework Vuetify que permite utilizar componentes de Vue que mediante props se puede modificar su estética como desee el cliente o el desarrollador pero siempre con un estilo de Material Design de Google.

2.4. Herramientas de Calidad de Código

EsLint - Prettier

El proyecto hace uso del linter EsLint [5] que es de código abierto. Su misión es analizar el código y encontrar patrones o código que no concuerde con ciertas pautas. Este linter permite al programador añadir sus propias reglas al igual que modificar las preestablecidas.

Prettier [12] es un formateador de código o code formatter que soporta Vue y JavaScript entre otros. Su función es modificar el formato del código original y lo remodela a un determinado estilo para que sea consistente. Lo que se ha hecho en este proyecto es integrarlo junto a EsLint.

Babel

Babel [1] es un compilador JavaScript gratuito y de código abierto que se puede usar para el desarrollo web. Lo que hace este compilador es permitir al desarrollador escribir el código fuente en HTML u otro lenguaje de programación y para que sea entendido por los navegadores traduce este código a JavaScript.

Capítulo 3

Descripción de la Aplicación

Este capítulo abarcará la explicación de todas las funcionalidades que posee la aplicación LiveTheOlympics.

3.1. Usuarios

Esta aplicación está destinada a todo el mundo de forma gratuita, por ello se tomó la decisión de que no existiría un método de autenticación ya que así facilita el uso y la navegación de usuario en la aplicación sin tener pasos previos para acceder al contenido, siendo su acceso a este el objetivo principal.

3.2. Sección de noticias

Como se a comentado en capítulos anteriores la aplicación constará de una sección de noticias, representada en las imágenes 3.1 y 3.2, la cual proporcionará a los usuarios interesados en usar la aplicación toda la información que los organismos oficiales cuelguen en su cuenta de la red social Twitter. Mediante esta funcionalidad, los usuarios pueden estar informados en todo momento a tiempo real de lo que ocurre en la arena.

Figura 3.1: Sección de noticias móvil

Figura 3.2: Sección de noticias pc

3.3. Sección de pruebas olímpicas

En esta sección el usuario podrá navegar por los distintos vídeos de las pruebas olímpicas o de este contenido relacionado que vayan subiendo los organismos oficiales a su canal de YouTube. Se ha decidido incorporar la retransmisión de las pruebas de esta manera ya que, la plataforma YouTube da soporte a la retransmisión de vídeos en directo además de dejar estos almacenados en el canal para posterior reproducción. La vista móvil y en ordenador están representadas en las imágenes 3.3 y 3.4.

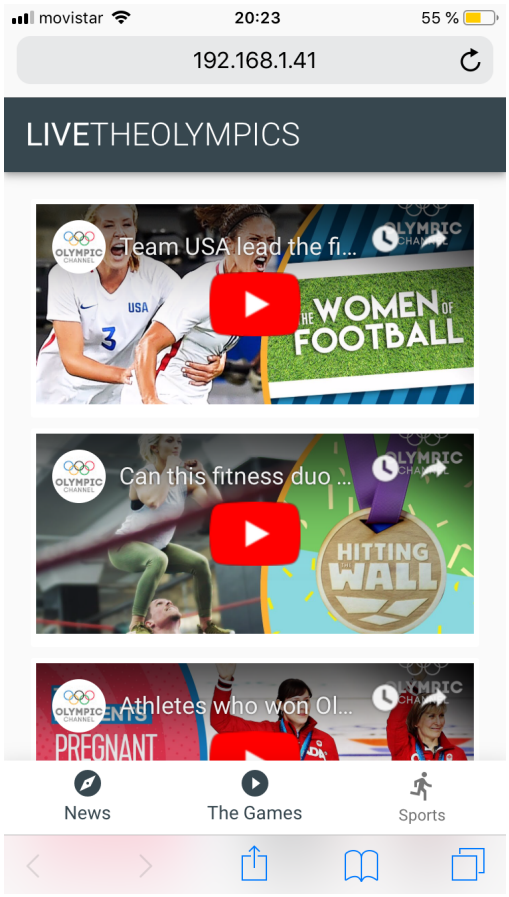


Figura 3.3: Sección de pruebas olímpicas móvil

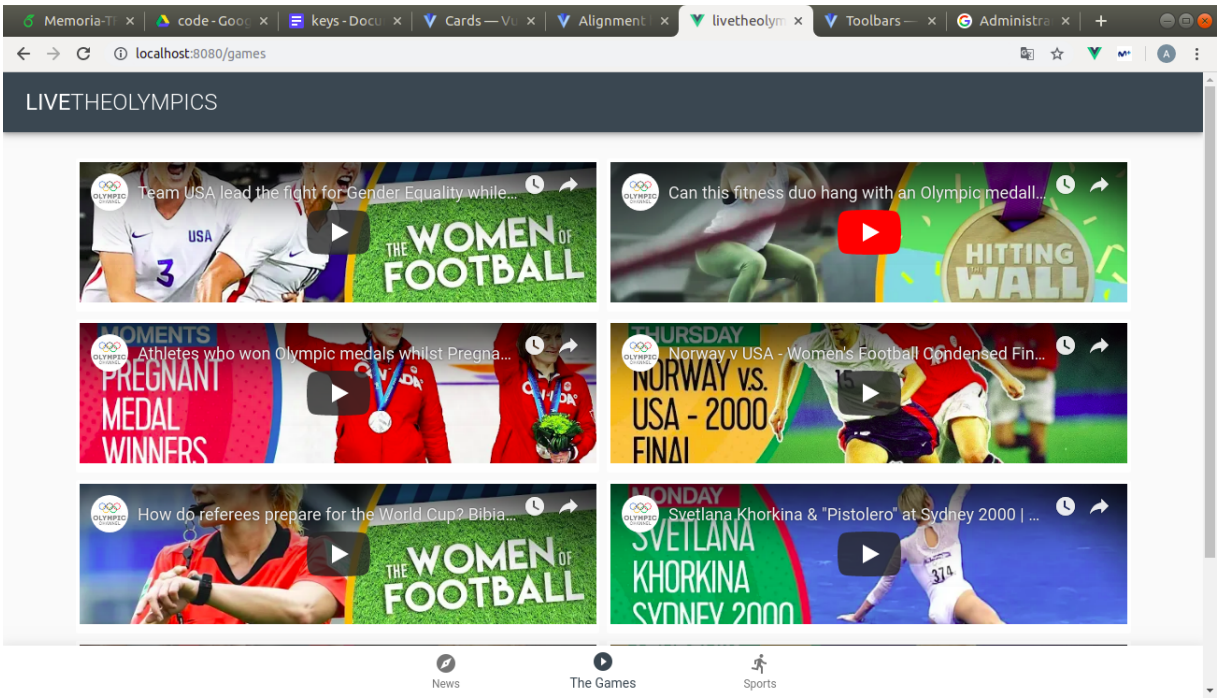


Figura 3.4: Sección de pruebas olímpicas pc

3.4. Sección de deportes

En esta vista, representada en las imágenes 3.5 y 3.6, el usuario podrá navegar por los diferentes deportes que acontecerán en los Juegos Olímpicos. Cada deporte será clicable lo que llevará al usuario a una vista donde aparezcan los nombres y fotografías de los atletas que competirán en este y una breve descripción de cada uno para aportar más información al usuario.

La información de muestra de los atletas ha sido extraída de las páginas dedicadas a estos en Wikipedia [18].

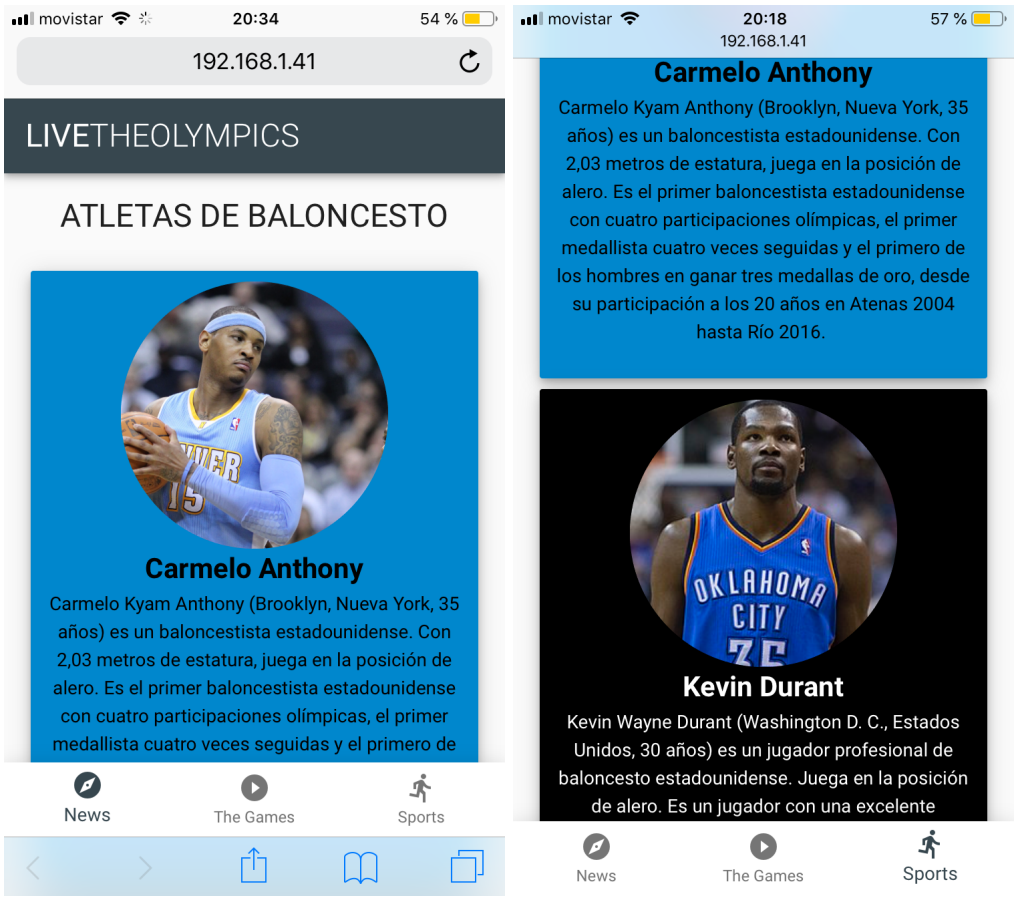


Figura 3.5: Sección de deportes móvil

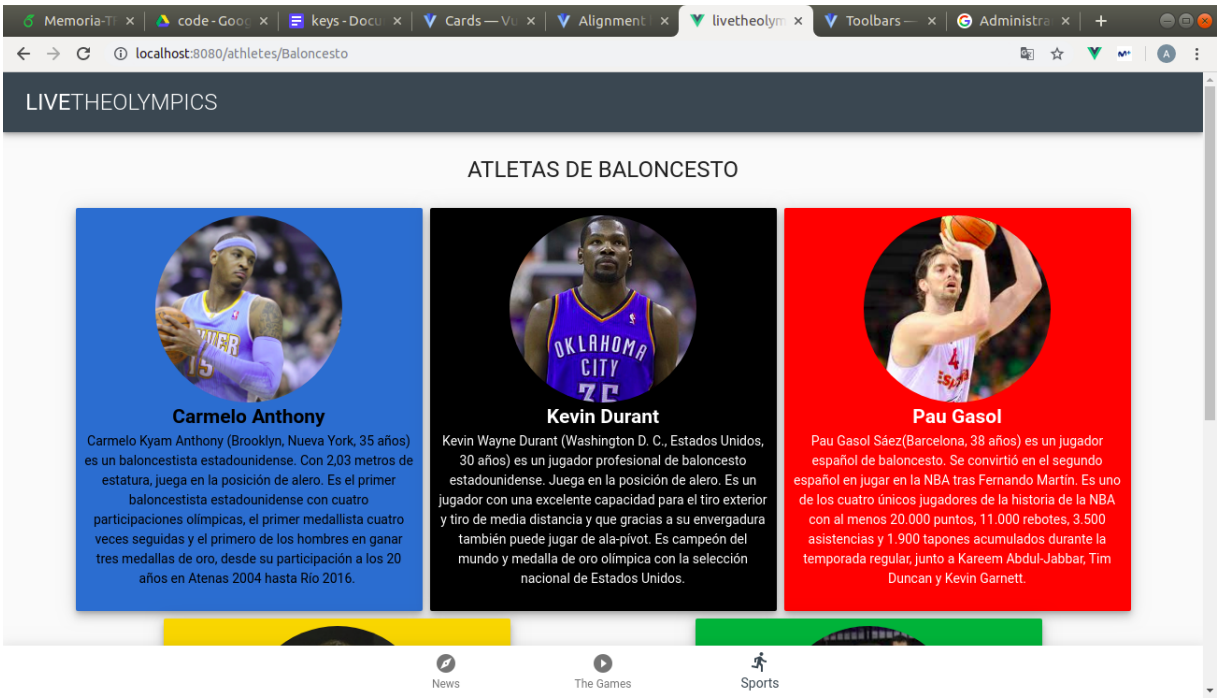


Figura 3.6: Sección de deportes pc

Capítulo 4

Desarrollo de la Aplicación

En el capítulo 3 se desarrolló la explicación de las diferentes funcionalidades de la aplicación. A continuación, en este capítulo, se explicarán todos los pasos para su desarrollo.

4.1. Primeros Pasos

Al ser un proyecto greenfield se tuvieron que instalar todas las herramientas necesarias para poder iniciar el desarrollo. Este se ha llevado acabo en un sistema linux por lo que se especificaran los comandos de este sistema operativo.

Antes de comenzar con el proyecto, se ha de crear el repositorio GitHub dónde se van a mantener las versiones y ramas de este. También se ha de crear una cuenta en todas las herramientas de las que se van a hacer uso, Heroku, Travis CI, Firebase y Google Cloud Platform para hacer uso de la YouTube Data API, y crear el proyecto en estas. Además, hay que instalar la versión de Node.js y Express que se va a emplear, siguiendo los pasos del código 4.1

```
1 curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
2 sudo apt install nodejs
3 npm install express
```

Listing 4.1: Instalar Vue CLI

Una vez preparadas las herramientas externas, el primer paso para comenzar el proyecto es instalar Vue CLI y crear un proyecto como se muestra en el código 4.2

```
1 npm install -g @vue/cli
2 vue create hello-world
```

Listing 4.2: Instalar Vue CLI

Al crear el proyecto se deberán elegir las dependencias necesarias de las que se van a hacer uso, aunque Vue CLI soporta que estas se añadan también una vez creado el proyecto.

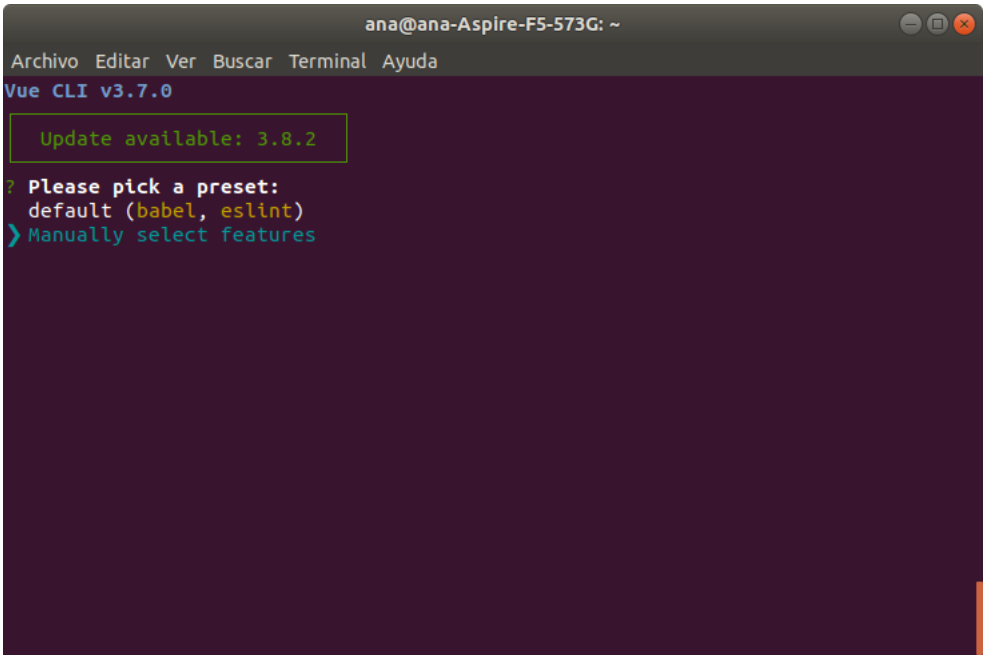


Figura 4.1: Selección de preset

Se elegirá la configuración de Eslint y Babel como herramientas de calidad de código, las cuales se definen en el capítulo 2, Router, Vuex, PWA Support y E2E Testing cómo dependencias necesarias para el correcto desarrollo del proyecto.

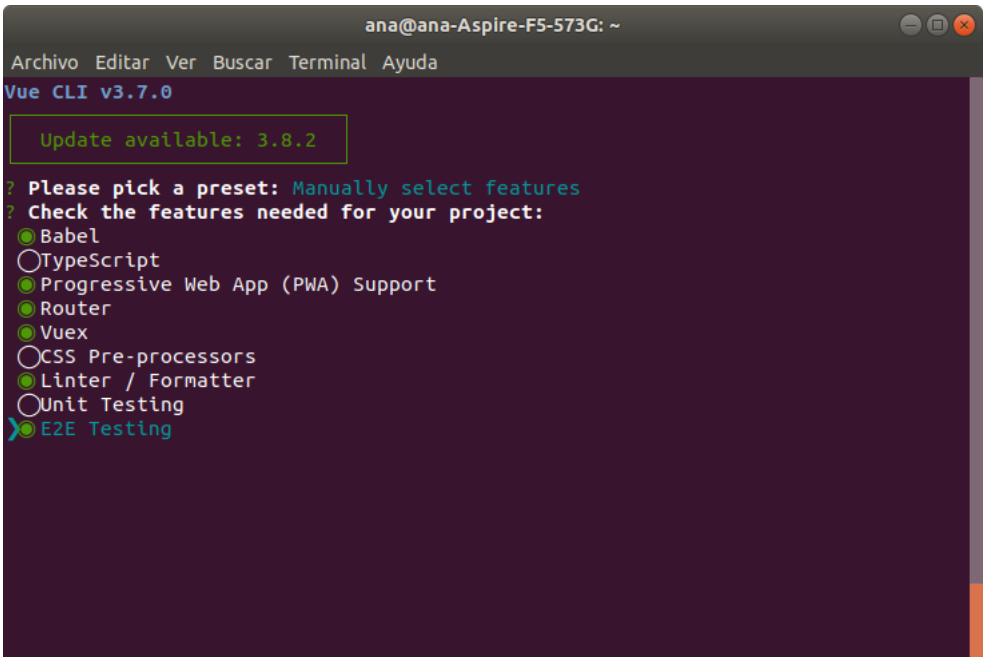


Figura 4.2: Selección de dependencias

A continuación se añadirá el formateador Prettier al linter como preferencia personal del desarrollador, todas las siguientes configuraciones dependen de las preferencias personales de desarrollo.

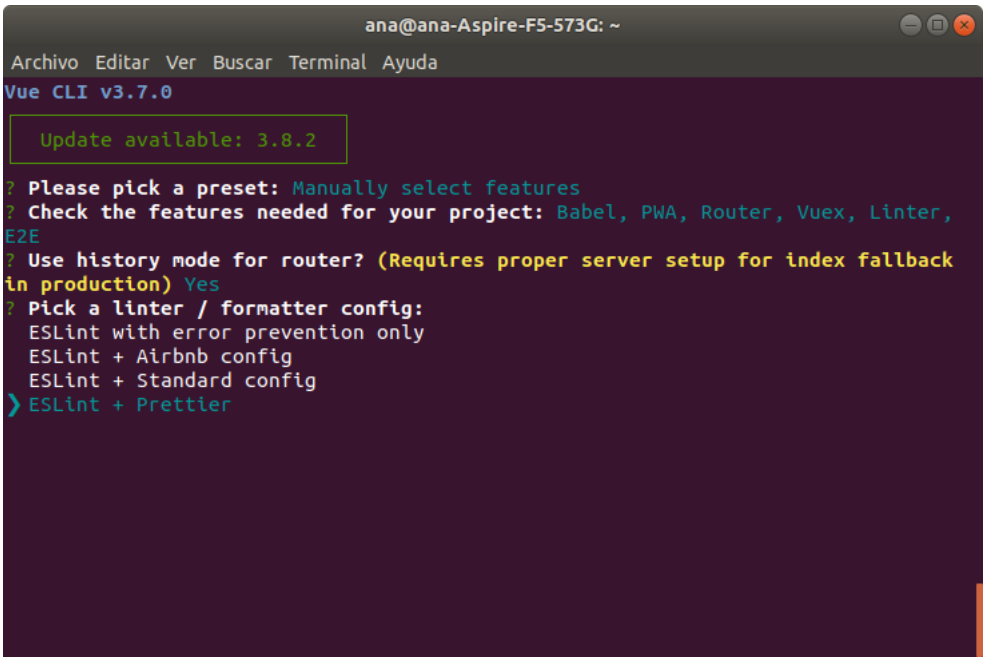


Figura 4.3: Añadir Prettier al linter

En la figura 4.4 se muestran el resto de configuraciones que decidió el programador.

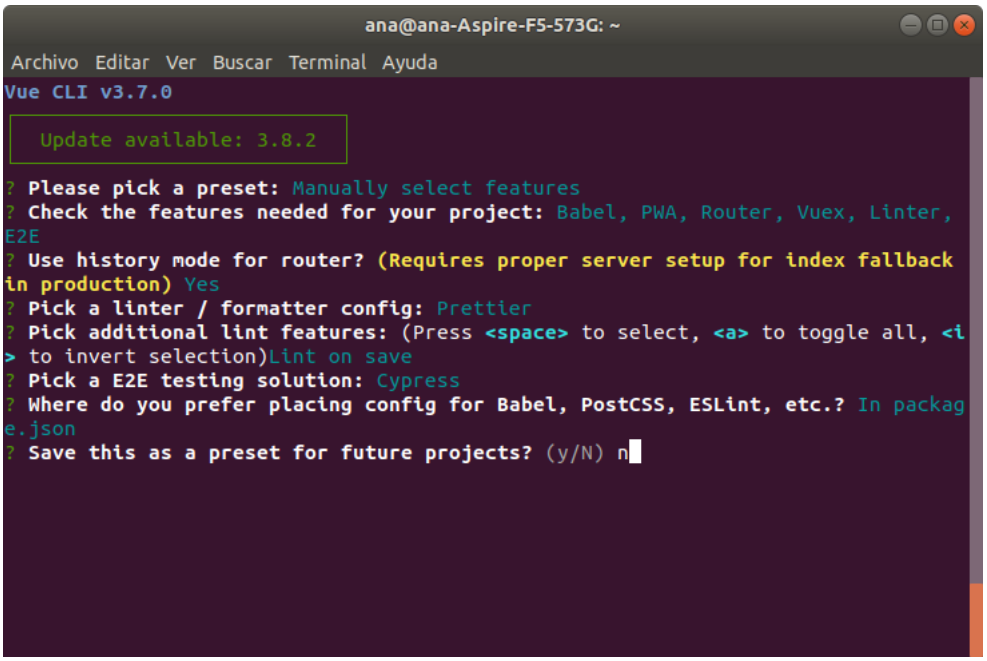


Figura 4.4: Elección del resto de configuración

Después de este proceso ya se tiene la base del proyecto sobre la que trabajar, como se empleará Vuetify junto con Vue CLI es necesario añadir esta dependencia.

```
vue add vuetify
```

Listing 4.3: Añadir Vuetify

4.1.1. Travis CI

Como se introdujo en el capítulo 2, la aplicación hace uso de Integración Continua mediante la herramienta Travis CI.

Para que se haga uso de esta, además de crear la cuenta en la plataforma como se señaló anteriormente, se debe crear un fichero llamado .travis.yml en el que se especifica el entorno de desarrollo y los comandos a ejecutar durante la integración. Para este caso será:

```
language: node_js
node_js:
- node
script:
  - npm install
```

Listing 4.4: .travis.yml Versión 1

4.1.2. Heroku

Como se decidió realizar el despliegue con Heroku mediante Travis CI, se ha de añadir al fichero .travis.yml, especificando el proveedor, la aplicación Heroku, la key de esta y el repositorio GitHub en el que se encuentra.

```
language: node_js
node_js:
- node
script:
  - npm install
deploy:
  provider: heroku
  api_key:
    secure: 97a4fe25-4443-4aa7-99e4-1a3647f70ba7
  app: livetheolympics
  on:
    repo: alu0100972016/livetheolympics
```

Listing 4.5: .travis.yml Versión 2

Añadiendo estas líneas se consigue que cuando se realiza la Integración Continua, también se despliegue la aplicación.

4.2. Diseño de Vistas

Una vez creado el proyecto se puede comenzar a definir las vistas y componentes que se van a usar. Dichos componentes se explican en las siguientes secciones.

El proyecto posee cuatro vistas principales, la de noticias, explicados sus componentes en la sección 4.3, la de pruebas olímpicas, en la sección 4.5, y la de deportes y la de atletas, explicados conjuntamente en la sección 4.4

Todas las vistas tienen un aspecto similar al código 4.6

```
<template>
  <TwitterTimeline />
</template>

<script>
import TwitterTimeline from "../components/TwitterTimeline.vue";

export default {
  components: {
    TwitterTimeline
  }
};
</script>
```

Listing 4.6: News.vue

En estas se importan el código generado en los componentes para renderizarse. Lo único que se ve modificado en las diferentes vistas son los componentes a los que se llama.

Para que se pueda navegar entre las diferentes vistas, ya que es una Single Page Application, se hace uso de Vue-Router como se muestra en el código 4.7

```
<template>
  <v-app>
    <Toolbar />

    <v-content>
      <router-view />
    </v-content>

    <v-content>
      <BottomNavigation />
    </v-content>
  </v-app>
</template>

<script>
import BottomNavigation from "./components/BottomNavigation.vue";
import Toolbar from "./components/Toolbar.vue";

export default {
  components: {
    BottomNavigation,
    Toolbar
  }
};
</script>
```

Listing 4.7: App.vue

Para que esto funcione se ha de modificar el fichero router.js y añadir las diferentes rutas. En este fichero se ha añadido una ruta por defecto para que si el usuario manipula la url siempre vaya a la vista de noticias.

```
import Vue from "vue";
import Router from "vue-router";
import News from "./views/News.vue";
import Games from "./views/Games.vue";
import Sports from "./views/Sports.vue";
import Athletes from "./views/Athletes.vue";

```

```
Vue.use(Router);

export default new Router({
  mode: "history",
  base: process.env.BASE_URL,
  routes: [
    {
      path: "/",
      name: "news",
      component: News
    },
    {
      path: "/games",
      name: "games",
      component: Games
    },
    {
      path: "/sports",
      name: "sports",
      component: Sports
    },
    {
      path: "/athletes/:sport",
      name: "athletes",
      component: Athletes
    },
    {
      path: "/*",
      name: "news",
      component: News
    }
  ]
});
```

Listing 4.8: router.js

Una vez implementado esto, para que el usuario pueda navegar de manera intuitiva, se ha añadido un componente BottomNavigation que dependiendo del botón que pulse el usuario carga una vista u otra, siempre iniciándose. Este componente esta creado como se indica en el código 4.9

```
<template>
  <v-container>
    <v-bottom-nav :active.sync="bottomNav" :value="true" fixed color="white">
      <v-btn to="/" color="blue-grey darken-3" flat value="news">
        <span>News</span>
        <v-icon>explore</v-icon>
      </v-btn>

      <v-btn to="/games" color="blue-grey darken-3" flat value="games">
        <span>The Games</span>
        <v-icon>play_circle_filled</v-icon>
      </v-btn>

      <v-btn to="/sports" color="blue-grey darken-3" flat value="sports">
        <span>Sports</span>
        <v-icon>directions_run</v-icon>
      </v-btn>
    </v-bottom-nav>
  </v-container>
</template>
```

```
21
22 <script>
23 export default {
24   data() {
25     return {
26       bottomNav: "news"
27     };
28   }
29 };
30 </script>
```

Listing 4.9: BottomNavigation.vue

4.3. Añadir Twitter Widget API

Esta ha sido la primera vista sobre la que se ha trabajado, se tuvieron dos acercamientos, el segundo siendo considerado el más adecuado y sencillo.

4.3.1. Primer acercamiento

Se analizó la Twitter Widget API, esta proporciona una Javascript API para añadir Twitter a los sitios web, que lo que hace es añadir el script widget.js. Esta es un poco engorrosa de usar y no sigue estándares de código limpio por lo que difícil de entender lo que hacer y los parámetros que utiliza. El código proporcionado por esta API es el que se muestra la etiqueta script en el código 4.10

```
1  <template>
2    <v-container align-content-center>
3      <v-layout justify-space-around row wrap>
4        <v-flex lg8>
5          <div id = "id">
6            <a class="twitter-timeline" href="https://twitter.com/juegosolimpicos?
                 ref_src=twsrc%5Etfw">Tweets by juegosolimpicos</a>
7          </div>
8        </v-flex>
9      </v-layout>
10   </v-container>
11 </template>
12
13 <script>
14 window.twttr = (function(d, s, id) {
15   var js, fjs = d.getElementsByTagName(s)[0],
16     t = window.twttr || {};
17   if (d.getElementById(id)) return t;
18   js = d.createElement(s);
19   js.id = id;
20   js.src = "https://platform.twitter.com/widgets.js";
21   fjs.parentNode.insertBefore(js, fjs);
22
23   t._e = [];
24   t.ready = function(f) {
25     t._e.push(f);
26   };
27
28   return t;
29 }(document, "script", "twitter-wjs"));
30 </script>
```

Listing 4.10: TwitterTimeline.vue Versión 1

4.3.2. Segundo acercamiento

El acercamiento anterior dejaba un problema, al cambiar de vista no se renderizaba bien el componente al volver a esta y no se volvían a cargar los tweets en la vista.

Este era un problema serio por lo que se siguió investigando para encontrar una solución hasta que se encontró el paquete vue-tweet-embed que permite que el script widget.js quede abstraído de la lógica del código aún siendo utilizado por el paquete.

Para ello se ha de instalar el paquete de la siguiente manera:

```
npm install vue-tweet-embed
```

Listing 4.11: Añadir vue-tweet-embed

Además, al usarlo, se arregla el problema de la recarga de la vista y queda un código más sencillo y limpio como el mostrado en el código 4.12.

```
<template>
  <v-container align-content-center>
    <v-layout justify-space-around row wrap>
      <v-flex lg8>
        <div id="id">
          <Timeline id="juegosolimpicos" sourceType="profile" />
        </div>
      </v-flex>
    </v-layout>
  </v-container>
</template>

<script>
import { Timeline } from "vue-tweet-embed";
export default {
  components: {
    Timeline
  }
};
</script>
```

Listing 4.12: TwitterTimeline.vue Versión 2

Esta es la versión final por la que se ha optado para el componente.

4.4. Listado de Deportes y Atletas

Para poder realizar listar los deportes y atletas que compiten en estos es necesario integrar Cloud Firestore en los componentes. Para ello se ha creado un archivo firebaseDB.js en el que se inicializa la conexión a Firebase y se va a importar al componente SportsMenu y al AthletesMenu.

```
import firebase from "firebase";

var firebaseConfig = {
  apiKey: "AIzaSyDdzEEqw3efYkfg0OapZPVTAbEVtEsly98",
  authDomain: "livetheolympics.firebaseapp.com",
  databaseURL: "https://livetheolympics.firebaseio.com",
  projectId: "livetheolympics",
  storageBucket: "livetheolympics.appspot.com",
  messagingSenderId: "157730914443",
  appId: "1:157730914443:web:41457f6994ddf7a1"
};
firebase.initializeApp(firebaseConfig);
```

```
13
14 export const firebaseDB = firebase.firestore();
```

Listing 4.13: firebaseDB.js

En los propios componentes se ha de realizar el trabajo de extraer los datos almacenados en Firestore. En el caso de SportsMenu, los datos a coger son la foto característica del deporte y su nombre, esto se almacena en un array que luego en el template se itera y se extraen los datos en los cards como se puede apreciar en el código 4.14

```
1  <template>
2    <v-container align-content-center>
3      <v-layout justify-space-around row wrap>
4        <v-flex pa-1 xs6 sm6 md6
5          v-for="(sport, index) in sports"
6          :key="sport.name"
7        >
8          <v-card
9            :color="ringsColor[index % ringsColor.length]"
10           :class="'pa-1 text-xs-center ' + textColor[index % textColor.length]"
11           elevation="6"
12           hover
13           :to="'athletes/' + sport.name"
14         >
15           <h2>{{ sport.name }}</h2>
16           <v-img aspect-ratio="1.3" :src="sport.img" />
17         </v-card>
18       </v-flex>
19     </v-layout>
20     <br />
21   </v-container>
22 </template>
23
24 <script>
25 import { firebaseDB } from "@/firebaseDB";
26 export default {
27   data() {
28     return {
29       sports: [],
30       ringsColor: ["#0087CD", "#000000", "#FB0222", "#FCC405", "#019F31"],
31       textColor: [
32         "black--text",
33         "white--text",
34         "white--text",
35         "black--text",
36         "black--text"
37       ]
38     };
39   },
40   mounted() {
41     firebaseDB
42       .collection("sports")
43       .orderBy("name")
44       .get()
45       .then(querySnapshot => {
46         this.sports = [];
47         querySnapshot.forEach(doc => {
48           this.sports.push(doc.data());
49         });
50       })
51       .catch(() => {
```

```
        alert("Error getting documents");
      });
  }
};
</script>
```

Listing 4.14: SportsMenu.vue

Tanto en este componente como en el de atletas, se van iterando los colores de los diferentes cards dependiendo de la posición del array, para que se vayan mostrando en los diferentes colores de los aros olímpicos.

Como se puede apreciar en el código 4.14 los cards tienen un prop :to, lo que permite que se establecezca la ruta para que cuando se pulse el card se dirija a la vista AthletesMenu y que muestre los datos de los atletas de ese deporte al que pertenece el card.

El código JavaScript de AthletesMenu es muy similar ya que sólo se trata de extraer información de la base de datos. Lo único que cambia son los cards ya que estos ahora poseen un avatar con la foto de los atletas, su nombre y una breve descripción, siguiendo el esquema de colores mencionado anteriormente.

```
<template>
  <v-container align-content-center>
    <h1 class="headline text-uppercase" align="center">
      Atletas de {{ $route.params.sport }}
    </h1>
    <br />
    <v-layout justify-space-around row wrap>
      <v-flex pa-1 xs12 sm6 md4
        v-for="(athlete, index) in athletes"
        :key="athlete.name"
      >
        <v-card
          height="100%"
          :color="ringsColor[index % ringsColor.length]"
          :class="'pa-2 text-xs-center ' + textColor[index % textColor.length]"
          elevation="6"
        >
          <v-avatar size="200px">
            <v-img :alt="athlete.name" :src="athlete.img" />
          </v-avatar>
          <h2 class="namep">{{ athlete.name }}</h2>
          <p class="descriptionp">{{ athlete.description }}</p>
        </v-card>
      </v-flex>
    </v-layout>
    <br />
  </v-container>
</template>

<script>
import { firebaseDB } from "@/firebaseDB";
export default {
  data() {
    return {
      athletes: [],
      ringsColor: ["#0087CD", "#000000", "#FB0222", "#FCC405", "#019F31"],
      textColor: [
        "black--text",
```

```
        "white--text",
        "white--text",
        "black--text",
        "black--text"
      ]
    };
  },

  beforeMount() {
    firebaseDB
      .collection("athletes")
      .where("sport", "==", this.$route.params.sport)
      .get()
      .then(querySnapshot => {
        this.athletes = [];
        querySnapshot.docs.forEach(doc => {
          this.athletes.push(doc.data());
        });
      })
      .catch(err => {
        alert("Error getting documents: " + err);
      });
  }
};
</script>
```

Listing 4.15: AthletesMenu.vue

Con esto estarían las vistas terminada, cualquier deporte o atleta que se añadiera a la base de datos, sería añadido dinámicamente a la aplicación web en tiempo real.

4.5. Añadir YouTube Data API

Para realizar la funcionalidad de la sección de las pruebas olímpicas se hizo uso de la YouTube Data API. Para ello hubo que crear una API KEY en el proyecto que creamos al inicio en Google Cloud Platform para poder hacer uso de esta en la aplicación web.

Como esta funciona de forma asíncrona en la aplicación, fue necesario el uso de axios para controlar las promesas y poder extraer la información necesaria de los datos que proporciona la API.

La API no obtiene directamente las url de los vídeos de un canal, sino que devuelve el ID de los diferentes vídeos con el que hay que generar la url de dicho vídeo.

El siguiente código 4.16 hace uso de la API, en la url se le especifica el canal del que se quieren coger los vídeos y un pageToken que se inicializa a vacío en la primera iteración, permite cargar los vídeos a medida que el usuario vaya haciendo scroll, sin esto, los vídeos se tendrían que cargar de golpe por lo que el tiempo de espera para que se montara el componente sería muy largo. También se marca en esta que se vayan adquiriendo los vídeos de ocho en ocho, que se ordenen por la fecha de subida y la API KEY del proyecto.

Para adquirir la información del canal se hace uso de axios que, una vez de respuesta la promesa, recoge el nextPageToken para que cuando se vuelvan a tener que cargar ocho vídeos más sepa cual es el último que se cargó y los ocho siguientes que tiene que cargar. La información necesaria para poder mostrar los vídeos es la que devuelve el objeto data. De este se extrae un array que por cada posición contiene el id el vídeo de esa posición más la url de este.

```
1   getVideos() {
2     const axios = require("axios");
3     const APIKEY = "AIzaSyC9gTmYA4ies0KSkMhx9KQrmQTA7wVVMCM";
4     const url = `https://www.googleapis.com/youtube/v3/search?part=snippet&
          channelId=UCT13QQTvqHFjurroKxexy2Q&pageToken=${
5       this.pageToken
6     }&maxResults=8&order=date&type=video&key=${APIKEY}`;
7     return axios.get(url).then(response => {
8       if (response.status === 200) {
9         this.pageToken = response.data.nextPageToken;
10        return response.data.items.map(item => ({
11          id: item.id.videoId,
12          src: "https://www.youtube.com/embed/" + item.id.videoId
13        }));
14      }
15    });
16  }
```

Listing 4.16: Implementación Youtube API

Para que esto funcione correctamente, hay que generar otras funciones que se muestran en el código 4.17 que comprueben en que punto de la pantalla está el usuario y si se tienen que cargar ocho vídeos más.

La función bottomVisible() solo calcula en si se encuentra al final de la pantalla, lo que devuelve un true o si no a llegado a este. Esta función es utilizada por el eventListener en la función created() que actualiza el valor de la variable bottom dependiendo de lo devuelto por la función y en addVideos() que comprueba se se encuentra al final de la pantalla y si es así llama a getVideos() y almacena esa información en el array videos.

Para que todo esto sirva, se implementa un observer, que lo que hace es comprobar el estado de la variable bottom, si esta es verdadera llama a addVideos() para que se añadan más vídeos.

```
1   bottomVisible() {
2     const scrollY = window.scrollY;
3     const visible = document.documentElement.clientHeight;
4     const pageHeight = document.documentElement.scrollHeight;
5     const bottomOfPage = visible + scrollY >= pageHeight;
6     return bottomOfPage || pageHeight < visible;
7   },
8   addVideos() {
9     if (this.bottomVisible()) {
10      this.getVideos().then(videos => {
11        this.videos.push(...videos);
12      });
13    }
14  }
15 },
16 watch: {
17   bottom(bottom) {
18     if (bottom) {
19       this.addVideos();
20     }
21   }
22 },
23 mounted() {
24   this.getVideos().then(videos => {
25     this.videos.push(...videos);
26   });
```

```
  },
  created() {
    window.addEventListener("scroll", () => {
      this.bottom = this.bottomVisible();
    });
    this.addVideos();
  }
```

Listing 4.17: Implementación cargar al hacer scroll

El código 4.18 representa el componente para mostrar estos vídeos.

```
    <template>
  <v-container align-content-center>
    <v-layout justify-space-around row wrap>
      <v-flex pa-1 xs12 sm6 v-for="video in videos" :key="video.id">
        <v-card flat class="pa-1 text-xs-center">
          <iframe
            :aspect-ratio="16 / 9"
            id="player"
            type="text/html"
            :src="video.src"
            frameborder="0"
            allowfullscreen
          ></iframe>
        </v-card>
      </v-flex>
    </v-layout>
  </v-container>
</template>

<script>
export default {
  data() {
    return {
      videos: [],
      bottom: false,
      pageToken: ""
    };
  },
  methods: {
    getVideos() {
      const axios = require("axios");
      const APIKEY = "AIzaSyB-RVbhrV7FxLqwtSHF6VaBgW-U-cma5hk";
      const url = `https://www.googleapis.com/youtube/v3/search?part=snippet&
          channelId=UCTl3QQTvqHFjurroKxexy2Q&pageToken=${
        this.pageToken
      }&maxResults=8&order=date&type=video&key=${APIKEY}`;
      return axios.get(url).then(response => {
        if (response.status === 200) {
          this.pageToken = response.data.nextPageToken;
          return response.data.items.map(item => ({
            id: item.id.videoId,
            src: "https://www.youtube.com/embed/" + item.id.videoId
          }));
        }
      });
    },

    bottomVisible() {
      const scrollY = window.scrollY;
      const visible = document.documentElement.clientHeight;
```

```
50        const pageHeight = document.documentElement.scrollHeight;
51        const bottomOfPage = visible + scrollY >= pageHeight;
52        return bottomOfPage || pageHeight < visible;
53      },
54      addVideos() {
55        if (this.bottomVisible()) {
56          this.getVideos().then(videos => {
57            this.videos.push(...videos);
58          });
59        }
60      }
61    },
62    watch: {
63      bottom(bottom) {
64        if (bottom) {
65          this.addVideos();
66        }
67      }
68    },
69    mounted() {
70      this.getVideos().then(videos => {
71        this.videos.push(...videos);
72      });
73    },
74    created() {
75      window.addEventListener("scroll", () => {
76        this.bottom = this.bottomVisible();
77      });
78      this.addVideos();
79    }
80  };
81  </script>
82
83  <style>
84  iframe {
85    width: 100%;
86    height: 100%;
87  }
88  </style>
```

Listing 4.18: YoutubeFeed.vue

Capítulo 5

Conclusiones y líneas futuras

Para concluir este trabajo recalcar que su realización ha sido un camino de aprendizaje adicional a lo que se cursa en el grado. Al ser un Trabajo de Fin de Grado con tantas posibilidades de ejecución fue un reto elegir la temática sobre la aplicación a realizar y las tecnologías empleadas. Fue una trayectoria de adquirir conocimiento nuevo, probar y fallar, de enfrentarse a herramientas que no conocía antes. Gracias a la ayuda del tutor y la búsqueda de información este camino de desarrollo se allanó bastante.

Se decidió emplear Vue para este proyecto ya que su popularidad iba cobrando fuerza y después de utilizarlo por primera vez he de decir que entiendo porqué. Es un framework que permite hacer lo mismo que otros frameworks de JavaScript pero de una manera mucho más sencilla y rápida para el programador. Además aporta la posibilidad de utilizarlo junto con Vuetify que da más potencia al desarrollo ya que permite que al escribir el componente se pueda añadir estilos directamente como props lo que genera un maquetado mucho más fácil y rápido.

También se han adquirido conocimientos nuevos sobre como integrar Firebase en un proyecto que utiliza frameworks o las APIs tanto de Twitter y YouTube que, con esta última sobretodo, se aprendió como funcionan las APIs de Google que puede ser de gran utilidad en el futuro.

Para finalizar las conclusiones, decir que este proyecto ha sido una idea propia. No se tuvieron que contar con los requisitos de cliente por lo que si estos cambiaban era por decisión propia, se sabía exactamente lo que se quería en cada momento y la decisión del maquetado estaba completamente a manos del desarrollador.

En cuanto a las líneas futuras, se podrían añadir más funcionalidades a esta aplicación como podría ser añadir un registro de usuarios. Así podríamos interactuar y comentar directamente en la propia aplicación, de más interactiva y social. También se podría modificar la pestaña de los vídeos de las pruebas y generar un caroussel en las demás vistas para que en cualquier página de navegación donde esté, el usuario se tenga fácil acceso a estos. También se podría generar un buscador para que el usuario pudiera acceder a la información que quisiera directamente, sin tener que navegar por la aplicación.

Si el proyecto llegara a tener éxito y se quisiera utilizar como medio oficial de transmisión de los Juegos Olímpicos habría que conseguir el permiso de reutilización de las imágenes oficiales proporcionadas por los organismos oficiales. Además también se debería conseguir el permiso de reproducción de los vídeos de YouTube fuera de su plataforma ya que actualmente el canal de Youtube de las Olimpiadas tiene deshabilitada esta funcionalidad.

Por todo lo anterior, se puede decir que la aplicación está abierta a mejoras y adiciones de funcionalidades que se le puedan ocurrir a diferente programadores y usuarios.

Capítulo 6

Summary and Conclusions

To conclude this project, emphasize that its realization has been a learning path in addition to what has been done during the degree. Being an End-of-Degree Project with so many possibilities of execution, it was a challenge to choose the topic on the application to be carried out and the technologies to be used. It was a trajectory of acquiring new knowledge, trying and failing, of facing tools that I did not know before. Thanks to the help of the tutor and the research of information, this development path was easier.

It was decided to use Vue for this project since its popularity was gaining strength and after using it for the first time I have to say that I understand why. It is a framework that allows to do the same as other JavaScript frameworks but in a much simpler and faster way for the programmer. It also provides the possibility to use it together with Vuetify that gives more power to the development since it allows that when writing a component you can add styles directly as props which generates a layout much easier and faster.

New knowledge has also been acquired on how to integrate Firebase in a project that uses frameworks or the APIs of both Twitter and YouTube that, with this last one, was learned how the Google APIs work and that can be very useful in the future.

To end the conclusions, to say that the development has been easier since it was an own idea. The developer does not have to take into account the requirements of a client, if those changed was by own decision, it was known exactly what was wanted in every moment and the decision of the layout was completely at the developer's hands.

As regards future lines, more functionalities could be added to this application, such as adding a user registry. This would enable them to interact and comment directly on the application, that is to say, it would be more interactive and social. The videos' tabs of the Olympic Games could be also modified so that instead of the tab, a carousel could be always present at the top of the app.

If the project were successful and wanted to be used as the official platform of broadcasting the Olympic Games, it would be necessary to obtain permission to reuse the official images. It would be also necessary to obtain permission to play YouTube videos outside the YouTube platform since currently the Youtube channel of the Olympics has disabled this functionality.

For all the above, it can be said that the application is open to improvements and additions of functionalities that different programmers and users can contribute.

Capítulo 7

Presupuesto

Debido a que es una aplicación para que su uso sea a escala mundial es complejo estimar la cantidad de usuarios que la van a poseer. Es por ello que hacer un plan con cualquier servicio de hosting es difícil. Sin embargo, la mayor parte de peticiones se delega a la parte de front-end ya que es esta la que hace las peticiones directamente a las API's de YouTube y Twitter.

Al tenerse que guardar y brindar datos sobre los deportes y atletas,se debe contratar un servicio que sea capaz de manejar gran cantidad de peticiones y que además pueda servir la página.

Ya que se ha usado Firebase para el desarrollo del proyecto se decidió hacer un presupuesto con uno de sus planes, el plan Blaze[2]. De hecho es el más flexible por si hace falta aumentar en momentos pico el ancho de banda permitido.

Como toda la información que se va a almacenar es texto, se estimó que con la cantidad de deportes y atletas por deporte haría falta un almacenamiento de 2GB.

La cantidad de lecturas a base de datos es uno de los parámetros más difícil de calcular ya que dependerá de la cantidad de usuarios que usen la aplicación y que requieran información del deporte o el atleta. Como medida de prevención se quiso estimar a lo alto.

Por último, al tener que servir la página también haría falta contratar un servicio de hosting. El peso de la web no es muy elevado (aproximadamente 2Mb) pero las peticiones de la misma sí serían elevadas.

Como la aplicación esta desplegada en Heroku (`https://livetheolympics.herokuapp.com/`), se ha decidido coger el servicio de hosting Standard 2x que ofrece 1 GB de RAM y permite el escalado horizontal, entre otras características

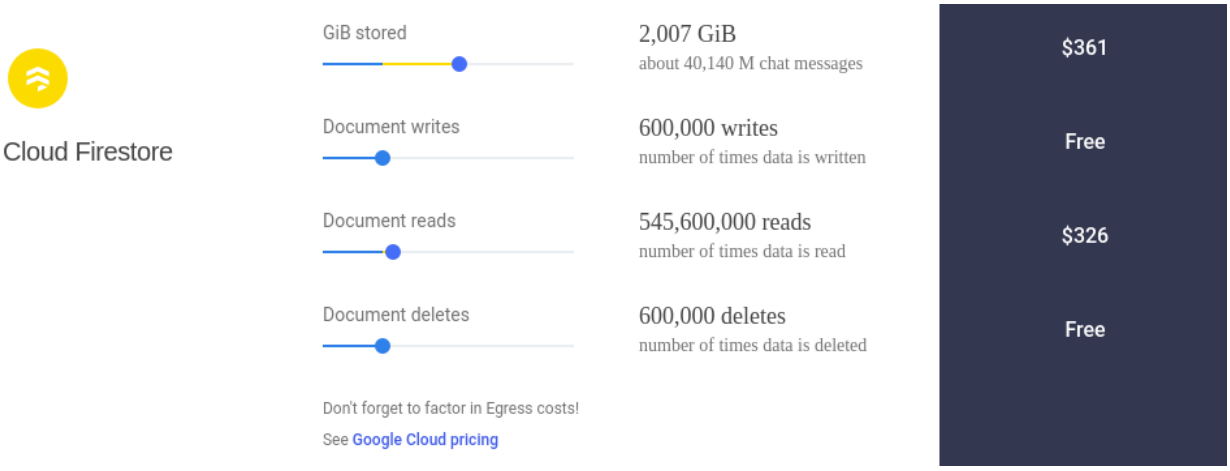


Figura 7.1: Presupuesto de Cloud Firestore por mes

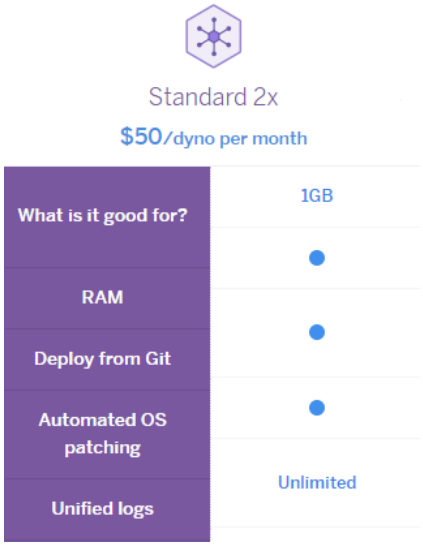


Figura 7.2: Presupuesto de Hosting con Heroku por mes

Concepto	Tiempo de contrato	Coste	Coste total
Firestore	3 meses	604,56€/mes	1813,68€
Hosting	3 meses	50€/mes	150€
Programador	180 Horas	8,33€/hora	1500€*
Coste total			3463,68€

Cuadro 7.1: Presupuesto para los tres meses de duración del evento

* El presupuesto de 1500€ de gastos de programador se ha calculado tras visionar diversas webs y ofertas de trabajo real, y concluyendo un salario bruto anual de 21000€ en 14 pagas. Como el tiempo dedicado al desarrollo del proyecto fue de aproximadamente 20 horas semanales se supuso un sueldo de media jornada, por tanto de 10500€ brutos anuales en 14 pagas. Ya que la dedicación fue de dos meses se llegó al presupuesto de 1500€. En las horas de trabajo sólo están contempladas aquellas dedicadas al desarrollo del software, sin contar la elaboración de la presente memoria.

Bibliografía

[1] Babel.js. Babel.js. https://babeljs.io/docs/en/. Accessed: 2019-06-09.

[2] Plan Blaze. Blaze: pay for usage plan of firebase. https://firebase.google.com/pricing#blaze-calculator. Accessed: 2019-06-08.

[3] Travis CI. Travis continous integration. https://docs.travis-ci.com/. Accessed: 2019-06-09.

[4] Vue CLI. Comand line interface for vue.js. https://cli.vuejs.org/. Accessed: 2019-06-09.

[5] Eslint. Eslint. https://eslint.org/docs/user-guide/getting-started. Accessed: 2019-06-09.

[6] Express.js. Express. https://expressjs.com/es/. Accessed: 2019-06-09.

[7] Firebase. Cloud firestore by firebase. https://firebase.google.com/docs/firestore?hl=es-419. Accessed: 2019-06-09.

[8] GitHub. Control de versiones git. https://github.com/torvalds/linux. Accessed: 2019-06-9.

[9] Google. Youtube data api v3. https://developers.google.com/youtube/v3/getting-started. Accessed: 2019-06-09.

[10] Heroku. Heroku: Platform as a service. https://devcenter.heroku.com/start. Accessed: 2019-06-09.

[11] npmjs. Dependencia vue-tweet-embed. https://www.npmjs.com/package/vue-tweet-embed. Accessed: 2019-06-09.

[12] Prettier. Prettier. https://prettier.io/docs/en/install.html. Accessed: 2019-06-09.

[13] Pivotal Tracker. Pivotal tracker. https://www.pivotaltracker.com/. Accessed: 2019-06-09.

[14] Twitter. Twitter javascript api. https://developer.twitter.com/en/docs/twitter-for-websites/javascript-api/guides/set-up-twitter-for-websites. Accessed: 2019-06-09.

[15] Vue-Router. Vue-router. https://router.vuejs.org/. Accessed: 2019-06-09.

[16] Vuetify. Framework based on vue.js with material disgn. https://vuetifyjs.com/en/. Accessed: 2019-06-09.

[17] Webpack. Webpack: bundle your scripts, images, styles, and assets. https://webpack.js.org/. Accessed: 2019-06-05.

[18] Wikipedia. Información sobre los atletas. https://es.wikipedia.org/wiki/Wikipedia:Portada. Accessed: 2019-06-09.

[19] Wikipedia. Patrón modelo-vista vista-modelo. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93modelo_de_vista. Accessed: 2019-06-09.