EASY EVENTS

TRABAJO FIN DE GRADO

Autor:
Andrés Sanz Rodríguez

Tutor: Mario Marugán Cancio

Desarrollo de Aplicaciones Multiplataforma
Curso 2019-2020

Centro Educativo: IFP
Fecha: Junio de 2020

Resumen

En la actualidad las personas organizan reuniones sociales para celebrar acontecimientos importantes para ellos, ya sea un cumpleaños, boda o el futuro nacimiento de sus hijos, ya sea propio o de algún familiar o amigos, y necesitan tiempo para poder organizarlo, sin embargo, no lo pueden realizar por falta de tiempo, porque no saben cómo organizar y crear un evento, por las necesidades que les pueda surgir. Tienen que pensar la forma de como invitar a sus familiares o amigos, buscar el lugar donde se realizará el evento, establecer una hora adecuada para los invitados.

En el presente proyecto, se va a desarrollar una aplicación para teléfonos inteligentes y para navegadores web, donde las personas particulares puedan planificar de una forma fácil y rápida, de manera que puedan ahorrar tiempo y puedan disponer del tiempo suficiente. Esto les podría ayudar a muchas personas a la hora de poder crear un evento.

La principal novedad que incluirá este proyecto, será que cada vez que se cree un acontecimiento se genera un identificador que se le podrá compartir a cada invitado, para que desde su móvil u ordenador puedan acceder al evento sin complicaciones y puedan observar los datos del mismo, sin ningún tipo de dificultad.

ÍNDICE

Contenido

Índice de ilustraciones

1 - Introducción

¿Por qué utilizar EasyEvents?

En la actualidad las personas no disponen de tiempo para realizar los eventos que querrían. Con esta aplicación web y móvil un usuario puede planificar un evento en cuestión de minutos.

En la vida diaria, siempre se presenta un evento ya sea un cumpleaños, un baby shower, una boda u otra fiesta sin motivo especial. La mayoría de personas no disponen de suficiente tiempo para realizar estos eventos, ya que tienen que trabajar o tienen otros deberes por cumplir.

Dados estos problemas se crea esta aplicación web y móvil, la cual busca la solución a este problema de falta de tiempo.

2 – Objectivo general y objectivos específicos

En este capítulo se desarrollará el objetivo general del Proyecto, así como los objetivos específicos, los mismos que detallan a continuación:

2.1 - Objetivo general

El objetivo principal es desarrollar una aplicación multiplataforma que ayude a las personas a gestionar sus eventos de una forma fácil y sencilla. También llegar a todos los particulares de una manera responsable para que la aplicación pueda llegar a utilizarse con seguridad de que sea una ayuda para ellos y que cuando ellos utilizan nuestra aplicación puedan crear y gestionar su evento sin ninguna complicación.

2.2 - Objetivos específicos

Para poder obtener el objetivo general realizaremos los siguientes objetivos específicos que detallamos a continuación:

- Se desarrollará un plan de proyecto para la aplicación.
- Se desarrollará la elicitación de requisistos para la aplicación.
- Se desarrollará un diseño o prototipo adecuado para la aplicación.
- Se diseñarán los modelos de datos persistentes que se almacenarán en la BBDD con la librería Mongoose.
- Se desarrollará una API REST en NodeJS.
- Se desarrollará el sitio web con el framework ReactJS.
- Se creará la aplicación para dispositivos móviles Android.

3 - Estado del arte

En esta apartado, se va hacer una revisión al estado actual de las tecnologías que han servido como base teórica para realizar el proyecto. Primero se abordará la evolución de las aplicaciones móviles, evolución del desarrollo web, la arquitectura REST y la arquitectura MERN stack en el desarrollo de aplicaciones móviles y desarrollo web. Por último, se comparará diferentes aplicaciones móviles y portales webs para la creación de un evento.

3.1 - Evolución de las aplicaciones móviles

En los últimos años las aplicaciones móviles o apps han constituido un ecosistema propio y un potente motor de innovación. La primera aplicación móvil surgió a finales de los años 90 algunas apps eran de contactos, agenda, editores de ringtone entre otras. Fueron y son apps que cumplen funciones muy básicas, pero que nos facilitan la gestión de nuestro día a día. Sin embargo, ha evolucionado rápidamente hacia las aplicaciones en detrimento de los navegadores. Actualmente la generación de teléfonos móviles y smartphones tienen distintos SO (Sistemas Operativos) propios como Android, IOS, Windows Phone, entre otros. [1] [2]

3.1.1 - Android

Android es un sistema operativo, inicialmente diseñado para teléfonos móviles como los sistemas iOS(Apple), FireFoxOS(Mozilla) y Blackberry OS. Está basado en Linux, que es un núcleo de sistema operativo libre, gratuito y multiplataforma.

Hoy en día se usa Android el teléfonos móviles, tablets, wearables, coches y televisiones.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android SDK (Android Software Development Kit, pero también lo puedes encontrar en otras herramientas de desarrollo, incluyendo un kit de Desarrollo Nativo. Se puede apreciar la visión global por capas de la arquitectura empleada en android(**Ilustración 1 : Arquitectura interna del sistema operativo** Android**)**. [3]

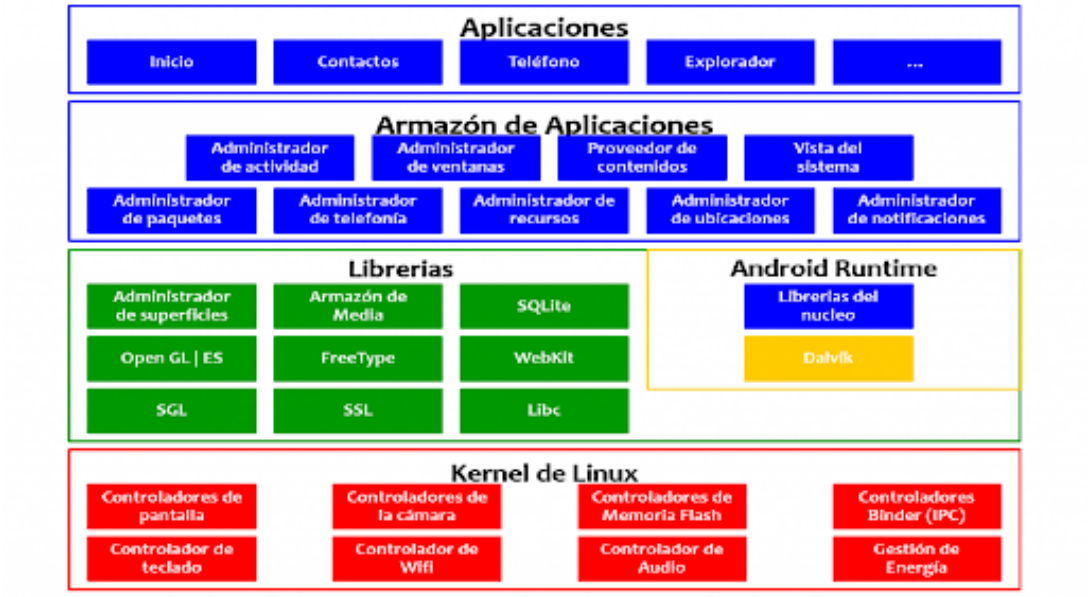

Ilustración 1 : Arquitectura interna del sistema operativo Android

3.1.1.1 - Android Studio

Android Studio es un entorno de desarrollo integrado oficialmente para la plataforma de Android. Se encuentra basado en el software IntelliJ IDEA de JetBrains y está disponible gratuitamente a través de la Licencia Apache 2.0, también lo puedes encontrar en las siguientes plataformas Microsoft Windows, macOS y GNU/Linux. [4]

Los aspectos fundamentales para el desarrollo de una aplicación en Android Studio, se estará detallando tanto como su distribución o estructura:

Componentes

Se clasifican, básicamente, en:

- ***Pantallas(Activity)***: Una de sus funciones principal es la creación de la interfaz del usuario. Las diferentes pantallas son independientes entre sí, aunque se puedan encontrar interrelaciones gracias a los Intents o a las solicitudes, lo cual es lo que permite la navegación a través de la aplicación. Toda pantalla puede pertenecer a una clase descendiente de la clase Activity.

  Las pantallas los puedes encontrar en 4 estados diferentes, también conocido como el ciclo de vida de una actividad como se puede ver en la siguiente imagen ():

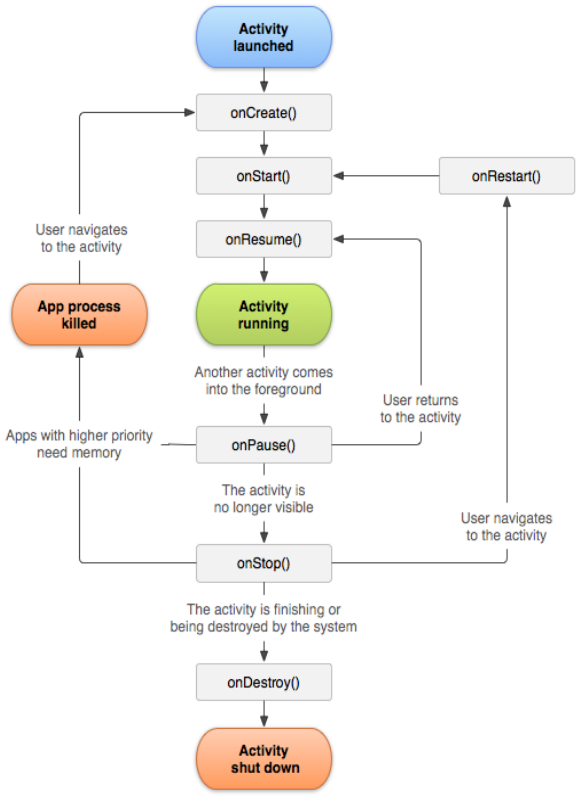

Ilustración 2 : Ciclo de vida de una actividad

- ***Fragmentos (Fragments)*:** Es considerado como una sección modular de una actividad que tiene ciclo de vida propio, están constituidos por la unión de varias vistas para crear un bloque funcional de la interfaz del usuario.
- ***Servicio (Service):*** Es un proceso que puede realizar operaciones que se ejecuta en segundo plano, sin necesidad de que tenga alguna interacción con el usuario. Estos servicios pueden ser locales, ejecutándose en el mismo proceso o remotos.

- ***Solicitudes(Intent):*** Los intents describen la actividad que se debe iniciar, también expresa la voluntad de realizar una determinada acción , como lanzar una nueva pantalla, un servicio o intercambiar información con componentes.

[5]

3.2 - Evolución del desarrollo web

El desarrollo de aplicaciones Web ha ido creciendo con el paso del tiempo. Las primeras páginas web estaban hechas con texto, lenguaje de hipertexto HTML y una gama de colores muy limitada.

Una aplicación web de hoy en día, para el correcto funcionamiento de una aplicación web está basada en MVC (Model View Controller) o también conocido en español patrón de diseño Modelo-Vista-Controlador. En la capa modelos incluye todos los aspectos relacionados con los datos, se realiza la extracción e inserción de datos, como también acceso a la información o actualización de cifras. En la capa de vista es la interfaz que interactúa con el usuario o el resto de sistemas, aquí se organizan los datos y se selecciona el modo en que éstos se desplegarán. Es decir en la capa vistas es la presentación visual de la capa Modelos. En la capa de controladores, se encarga de gestionar todo el flujo de trabajo de la aplicación. Organiza entre las dos capas anteriores y establece los parámetros en los que se realiza, esta capa reacciona a las solicitudes del cliente de la aplicación, ejecutando las acciones adecuadas y creando los modelos necesarios. [6]

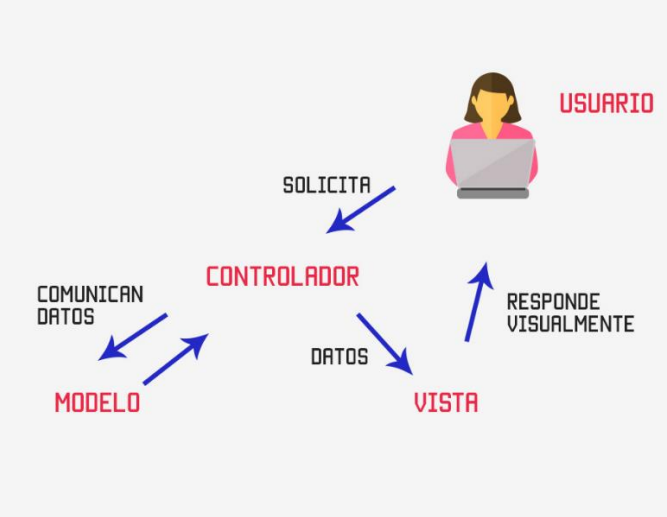

Ilustración 3 : Modelo-Vista-Controlador

3.3 - Arquitectura MERN

Cualquier aplicación web se realiza utilizando múltiples tecnologías. La combinación de estos las tecnologías se denominan "pila", popularizadas por la pila LAMP, que es un

acrónimo para Linux, Apache, MySQL y PHP, que son todos componentes de código abierto.

MERN es una pila de software para crear sitios web activos y aplicaciones web. Es una colección de tecnologías basadas en JavaScript para desarrollar aplicaciones web. Por lo tanto, desde el cliente hasta la base de datos, todo se basa en JavaScript. Esta pila de código abierto y gratuito es un acrónimo que se compone de las primeras letras de MongoDB, Express.js, React y Node.js(**Ilustración 4 : La pila MERN**). Para cada uno de estos componentes, se detalla sus características / ventajas y limitaciones.

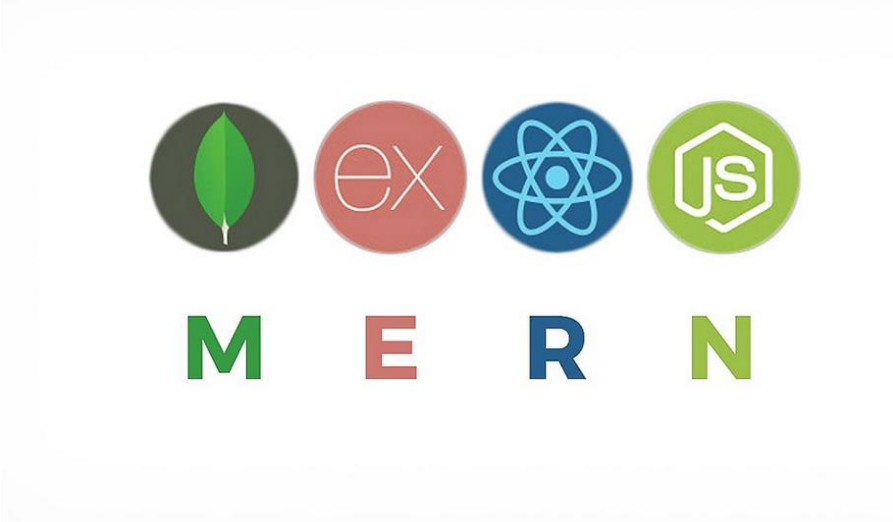

Ilustración 4 : La pila MERN

3.3.1 - NodeJS

NodeJS es un entorno de tiempo de ejecución, concebido como un entorno de ejecución de javascript y es de código abierto. Fue creado por Ryan Lienhart Dahl en 2009, está programado en c++ y javascript. Además es multiplataforma se puede usar en Windows, Mac OS X, Linux, Solaris, FreeBSD, OpenBSD y webOS. Un bucle de eventos es un mecanismo eficiente en Node que permite la ejecución de un solo subproceso. Todo lo que sucede en Node es una reacción a un evento. [7]

Entonces, podemos decir que Node es una plataforma basada en eventos y que el bucle de eventos garantiza que Node siga ejecutándose. Node usa bucles de eventos similares a una cola FIFO (primero en entrar, primero en salir) para organizar las tareas que tiene que hacer en su tiempo libre, tal como se mostrará en la siguiente (**Ilustración 5 : Ciclo de eventos NodeJS**):

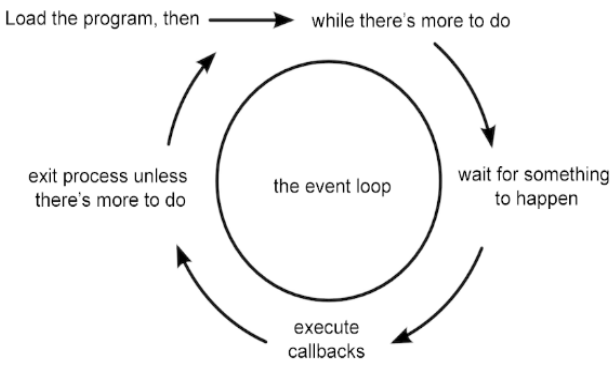

Ilustración 5 : Ciclo de eventos NodeJS

Algunas de las características que lo destaca es:

- Es un base de código única y potente, NodeJS funciona como una tecnología innovadora en el campo de desarrollo web, tanto en el lado del servidor como en el lado del cliente con codificación JavaScript.
- NodeJS incluye NPM(Node Package Manager) con el repositorio de casi 50.000 paquetes. Los desarrolladores pueden compartir, actualizar o reutilizar fácilmente el código con la ayuda de este NPM incorporado, es decir enriquece el intercambio.
- NodeJs es extremadamente modular, esto quiere decir que se divide en varios módulos por cada característica que tiene, por ejemplo, existe un módulo para usar el File System, otro para conexión base de datos, entre otros.

3.3.2 - Mongo

MongoDB una base de datos distribuida, orientada a documentos ya que todos los datos son guardados en ellos.

En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Para crear la base de datos no es necesario seguir ningún esquema en concreto ya que puede tener diferentes esquemas.

CARACTERÍSTICAS:

- Permite adaptar el esquema de la BBDD a las necesidades de la aplicación.
- Alta disponibilidad.
- Gran escalabilidad.
- Replicación, sincroniza los datos entre servidores.

3.4 - Aplicaciones y servicios web similares

A continuación, se realizará una comparación con una app similar:

3.4.1 - Pro party planner

Es una app ideal para reuniones, conferencias, fiestas y eventos, también es una de las apps más completas que encontrarás a la hora de organizar un evento. Desde importar lista de invitados delegar tareas, establecer recordatorios, sincronizar con otros usuarios, parecido a la red social de Facebook. Una de sus características que tienes es que te permite seguir en tiempo real lo que tiene que estar ocurriendo.

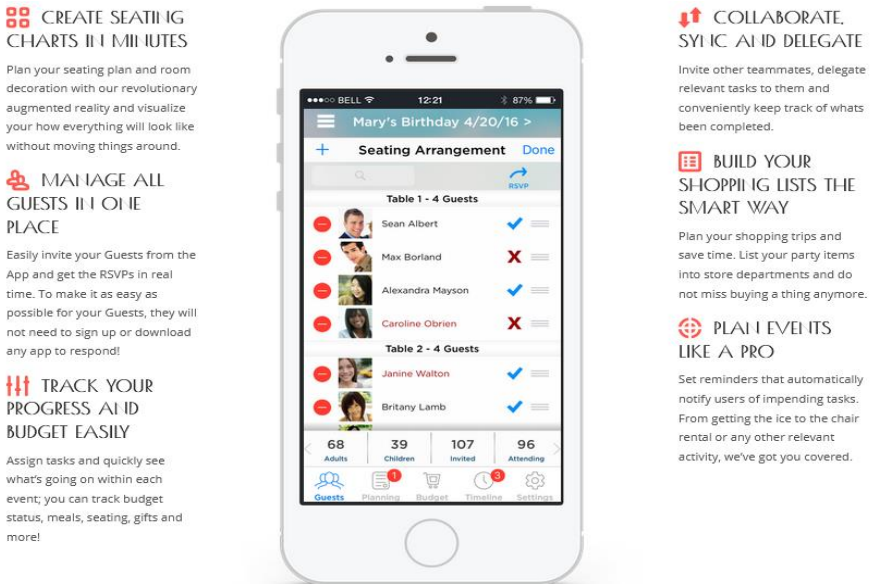

Ilustración 6 : Pro Party Planner

Características a destacar:

- Permite diseñar el salón de fiestas o cómo estará organizado y utilizar mapas de asientos, arrastrando a los lugares como desees.
- Tiene una buena gestión de la lista de invitados.
- Administra presupuestos y la lista de invitados, podrá confirmar las asistencias con llamadas telefónicas o mediante correos.

Ventajas de EasyEvents:

- Coste anual 0€ sobre 5,49€ de Pro Party Planner.
- Interfaz sencilla y fácil de usar.
- Código de evento público.

- Posibilidad de elegir el tipo de evento.
- Muestra la ubicación del evento en GoogleMaps.

4 - Métodos de trabajo

En este punto se detalla los métodos de trabajo aplicados a este proyecto.

4.1 - Marco Tecnológico

Aquí se detallarán las herramientas utilizadas en el proyecto.

Gestión de proyectos	Modelado	Desarrollo de software
git	draw.io	mongoDB React express android node

4.2 - Herramientas para el modelado software

4.2.1 - Draw.io

Herramienta para el desarrollo de los diagramas UML y Entidad/Relación.

4.3 - Herramientas y tecnologías para el desarrollo del proyecto

- **Android Studio**: Entorno de desarrollo integrado para el desarrollo de apps solo para Android, basado en IntelliJ IDEA, usado para el proyecto.
- **NodeJS**: Crea aplicaciones en el servidor, alta rapidez cuando se realizan grandes cantidades de peticiones.
- **Express**: framework para crear rápidamente apliaciones servidor en NodeJS.
- **MongoDB**: Base de datos noSQL, almacena datos en documentos.

- **React.js**: framework Javascript para desarrollo rápido de interfaces web interactivas.
- **Mongoose**: librería con funciones predefinidas y modelos de datos para usar con MongoDB, ofrece un rápido desarrollo gracias a sus funciones predefinidas.

4.4 - Herramientas para la gestión del proyecto

4.4.1 - Git

Git permite el control de versiones del código fuente, es decir administra las diferentes versiones de nuestra aplicación, por ejemplo ayuda a soluccionar nuevos errores que antes no había o a colaborar con otros desarrolladores.

Ilustración 7 : Git

4.5 – Planificación del proyecto

A continuación de explicarán las fases de proyecto que se han llevado para el desarrollo del proyecto.

Iteracción 1: Se desarrollará el plan de proyecto.

Iteracción 2: Se desarrollará la elicitación de requisistos.

Iteracción 3: Se diseñarán los modelos de datos persistentes que se almacenarán en la BBDD con la librería Mongoose.

Iteracción 4: Se desarrollará una API REST en NodeJS.

Iteracción 5: Se desarrollará el sitio web con el framework ReactJS.

Iteracción 6: Se desarrollará la aplicación para dispositivos móviles Android.

5 - Resultados

En este apartado se explicará la función final del proyecto, se mostrará los diagramas y la viabilidad del mismo.

5.1 – Iteraciones

Iteracción 15.1.1.1 Estudio de viabilidad

Con el siguiente estudio se pretende demostrar la vialibilidad del proyecto presentado.

Se tienen en cuenta 3 factores:

- **Viabilidad económica**: Evaluación de los costos del desarrollo.
- **Viabilidad técnica**: Estudio de los conocimientos y aptitudes adquiridas durante los estudios.
- **Viabilidad legal**: Determinar cualquier posibilidad de infracción o responsabilidad legal.

5.1.1.1.1 - Viabilidad económica

El análisis económico es una tarea complicada en los proyectos de desarrollo de software, debido a la imposibilidad de calcular el tiempo exacto que se va a tardar en acabar el proyecto y a los contratiempos que puedan surgir durante la realización del mismo.

Se han determinado los siguientes **gastos directos**:

- 800€ de Sueldo de desarrollador x 3 meses de trabajo = 2400€

Se han determinado los siguientes **gastos indirectos**:

- 50€ consumo de Luz x 3 meses de trabajo = 150€
- 40€ consumo de gas x 3 meses de trabajo = 120€
- 60€ cuota Internet x 3 meses de trabajo = 180€

Sumando ambos costes resulta el **coste total** del proyecto = 2400€ + 450€ = **2850€**

5.1.1.1.2 - Viabilidad técnica

El análisis técnico consiste en realizar una evaluación del software, personas disponibles y si tienen las capacidades técnicas requeridas.

Para la realización de este Trabajo Fin de Grado se cuentan con las siguientes herramientas de trabajo:

- Ordenador portátil Lenovo Thinkpad 20FMS6UU06

- Windows 10 Pro
- Framework NodeJS
- Framework React
- Microsoft Visual Studio Code 1.45
- Android Studio 3.6.0
- Google Chrome 83.0
- Github

Todas las tecnologías son *OpenSource* con lo que no se ha comprado ninguna licencia de software, con ello ahorrando costes.

En cuanto a las capacidades técnicas de la persona encargada de realizar este proyecto, el alumno ha adquirido el conocimiento necesario a lo largo de los años de Grado Superior.

5.1.1.1.3 - Viabilidad legal

En este estudio se asegura que el proyecto no infringe ninguna norma o ley establecida.

Además se debe garantizar el respeto a los acuerdos relacionados con el ámbito del proyecto. También se deben tener en cuenta que las licencias para el software empleado debe ser

auténticas para evitar inconvenientes legales futuros. Y es por estas razones por las que se ha decidido poner atención en la *Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal*.

5.1.3 - Iteracción 2

Se desarrollará la elicitación de requisistos.

Se han observado los requisitos de la aplicación que se detallan a continuación.

- Un Evento tiene un código público generado automáticamente por la API que se puede compartir para ver los detalles de dicho evento.
- Un Usuario puede crear y compartir eventos.
- Los Eventos tendrán los siguientes atributos: código id público, nombre, anfitrión, tipo, mensaje, fecha y hora , fecha y hora de creación, lugar, nombre del anfitrión y email de los invitados.
- La API recibirá peticiones POST al crear eventos y GET para recuperar información de un Evento.

- La librería que conectará la API con la BBDD es Mongoose.
- La BBDD será MongoDB de tipo **NoSQL almacenada en la nube** con su nodo principal en Frankfurt(Alemania).

5.1.3 - Iteracción 3

Se diseñarán los modelos de datos persistentes que se almacenarán en la BBDD con la librería Mongoose.

Después de revisar la documentación de Mongoose se crea el **EsquemaEvento** con sus atributos, tipos de dato y restricciones.

Se incluye un validador para el tipo de evento disponiendo de 5 opciones: Cumpleaños, Boda, Despedida de Soltero, Babyshower y Fiesta.

De acuerdo con el patrón **MVC**(Modelo-Vista-Controlador) este sería el **Modelo**.

```
const EventSchema = new Schema({
  type: {
    type: String,
    required: true,
    validate: {
      validator: (v) => {
        if (
          v === "cumpleanos" ||
          v === "boda" ||
          v === "despedidadSoltero" ||
          v === "babyShower" ||
          v === "fiesta"
        )
          return true;
      },
    },
  },
  name: { type: String, required: true },
  message: { type: String, required: true },
  publicIdCode: { type: Number, unique: true, required: true },
  date: { type: Date, required: true },
  dateCreated: { type: Date, required: true },
  place: { type: String, required: true },
  hostName: { type: String, required: true },
  guestsEmails: { type: Array, required: false },
});
```

Ilustración 8 : Modelo Evento Mongoose

5.1.4 - Iteracción 4

Se desarrollará una API REST en NodeJS.

En la API REST se hará uso del framework **Express**, que proporciona un **desarrollo muy rápido** e incluye funciones predefinidas que se emplearán en este proyecto como las funciones de Express Router.

5.1.4.1 – Petición GET

En esta ilustración (**Ilustración 9 : Petición GET a API REST**) se emplea el **Router de Express** que utilizaremos para la funcionalidad de las rutas.

Al realizar posteriormente la aplicación web o móvil una **petición GET** con un parámetro, el código del evento, la API llama al método getEventInfo del modeloEvento que obtendrá la información de la BBDD.

De acuerdo con el patrón **MVC**(Modelo-Vista-Controlador) este sería el **Controlador**.

```
router.get("/:publicId", (req, res) => {
  var publicId = req.params.publicId;
  if (publicId) {
    EventModel.getEventInfo(publicId)
      .then((resolve) => {
        res.end(createResponseMsg("success", resolve));
      })
      .catch((err) => {
        res.end(createResponseMsg("error", err));
      });
  } else {
    res.end(createResponseMsg("error", "no results"));
  }
});
```

Ilustración 9 : Petición GET a API REST

Al realizar una petición GET satisfactoria con un código de evento que existe la API devuelve una respuesta en formato JSON con los datos del evento.(**Ilustración 10 : Respuesta peticion GET a la API formato JSON**).

```
{
    type: "success",
  - msg: {
      - guestsEmails: [
            "andrewelamopolski@gmail.com",
            "jose@gmail.com",
            "mariomarugan10@hotmail.com"
        ],
        _id: "5ee367eb26a0f12d68100b01",
        type: "cumpleanos",
        name: "Mi Cumple",
        publicIdCode: 4110,
        message: "Os invito a mi cumple en Madrid, Â¡Â¡ os espero !!",
        date: "2020-06-29T04:30:00.000Z",
        dateCreated: "2020-06-12T11:32:59.129Z",
        place: "calle del arenal 32 madrid",
        hostName: "Manuel",
        __v: 0
    }
}
```


Ilustración 10 : Respuesta peticion GET a la API formato JSON

En la API se hace uso de la **programación asíncrona** nativa en Node.js y en Javascript con la función **Promise**(Promesa), que tiene dos callbacks la función resolve y la función response.

En esta ilustración (**Ilustración 11 : Uso de Promesas al obtener información del evento**) se representa la función getEventInfo (obtener información del evento) que recibe el código de evento como argumento y devuelve una promesa.

Esta promesa se resuelve o finaliza cuando desde el EventModel la función find devuelve un resultado, si el evento existe devuelve sus datos con la función **resolve**, en el caso de que el evento no exista o haya algún error devuelve el error con la función **reject**.

```
getEventInfo: (publicIdCode) => {
  return new Promise((resolve, reject) => {
    EventModel.find({ publicIdCode: publicIdCode }, (err, docs) => {
      if (err) {
        console.log(err);
        console.log("error");
        reject(err);
      } else {
        if (docs.length === 0) {
          console.log("error no results");
          reject("no results");
        } else {
          resolve(docs[0]);
        }
      }
    });
  });
},
```

Ilustración 11 : Uso de Promesas al obtener información del evento

5.1.5 - Iteracción 5

Se desarrollará el sitio web con el framework ReactJS.

Con el framework React JS se ha desarrollado una aplicación estructurada en componentes, que proporciona una alta escalabilidad y un fácil mantenimiento.

Para el manejo de rutas en el sitio web se utiliza React Router una librería potente y fácil de usar, no necesita refrescar la página como un link normal, en su vez carga los componentes dinámicamente.

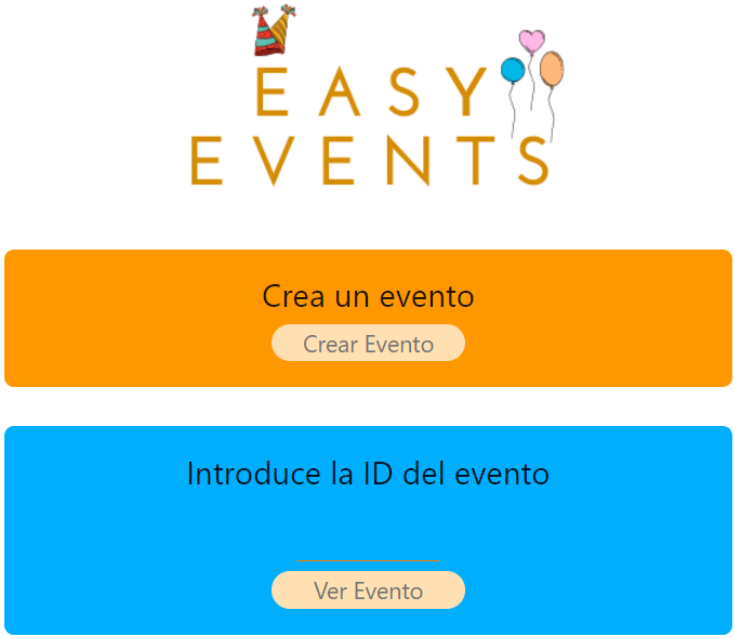

Ilustración 12 : Aplicación Web Inicio

La interfaz para crear un evento utiliza componentes de la librería **MaterialDesign** y la librería **Axios** para enviar peticiones POST a la API ver(**Ilustración 14 : Interfaz Crear Evento 1** , **Ilustración 15 : Interfaz Crear Evento 2**).

También se incluye un componente GoogleMaps que dada una dirección, la muestra en el mapa ver (**Ilustración 13 : Componente MapGoogle**). La dirección se pasa de un componente a otro a través de props, se puede entender como argumentos que se pasan a una función (componente) en React.

```jsx
import React, { Component } from "react";

export class MapGoogle extends Component {
  render() {
    return (
      <div>
        <iframe
          title="Mapa"
          width="100%"
          height="500"
          frameBorder="0"
          style={{ border: "0" }}
          src={
            "https://www.google.com/maps/embed/v1/place?key=AIzaSyDKe59cKaR_I1I593KQHxXjK7tbkr81SnI&q=" +
            this.props.query
          }
          allowFullScreen
        ></iframe>
      </div>
    );
  }
}

export default MapGoogle;
```

Ilustración 13 : Componente MapGoogle

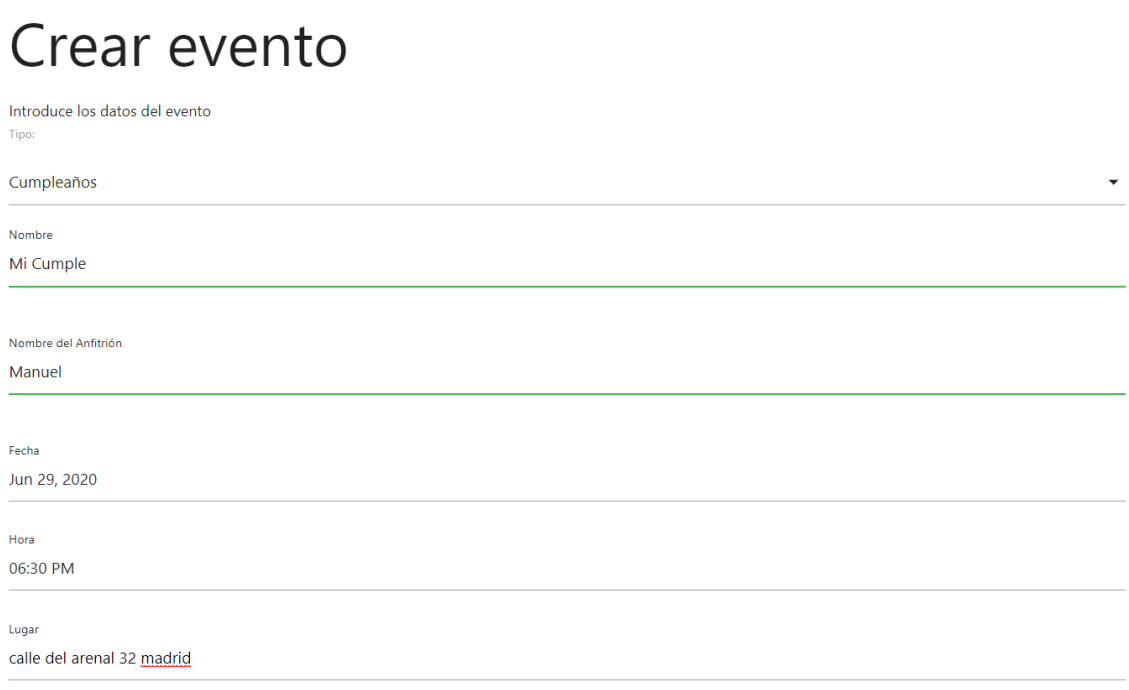

Ilustración 14 : Interfaz Crear Evento 1

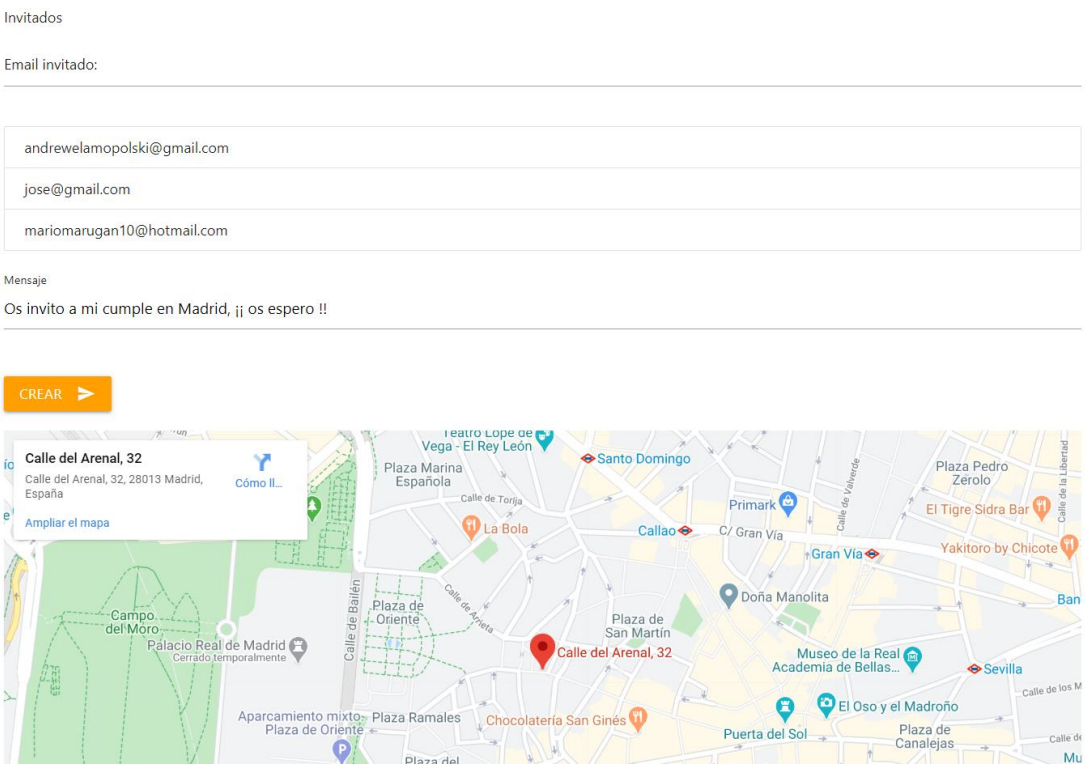

Ilustración 15 : Interfaz Crear Evento 2

Al colocar el código de un evento existente y se hace click ver Evento se muestra la información del evento ver (**Ilustración 16 : Interfaz información evento**).

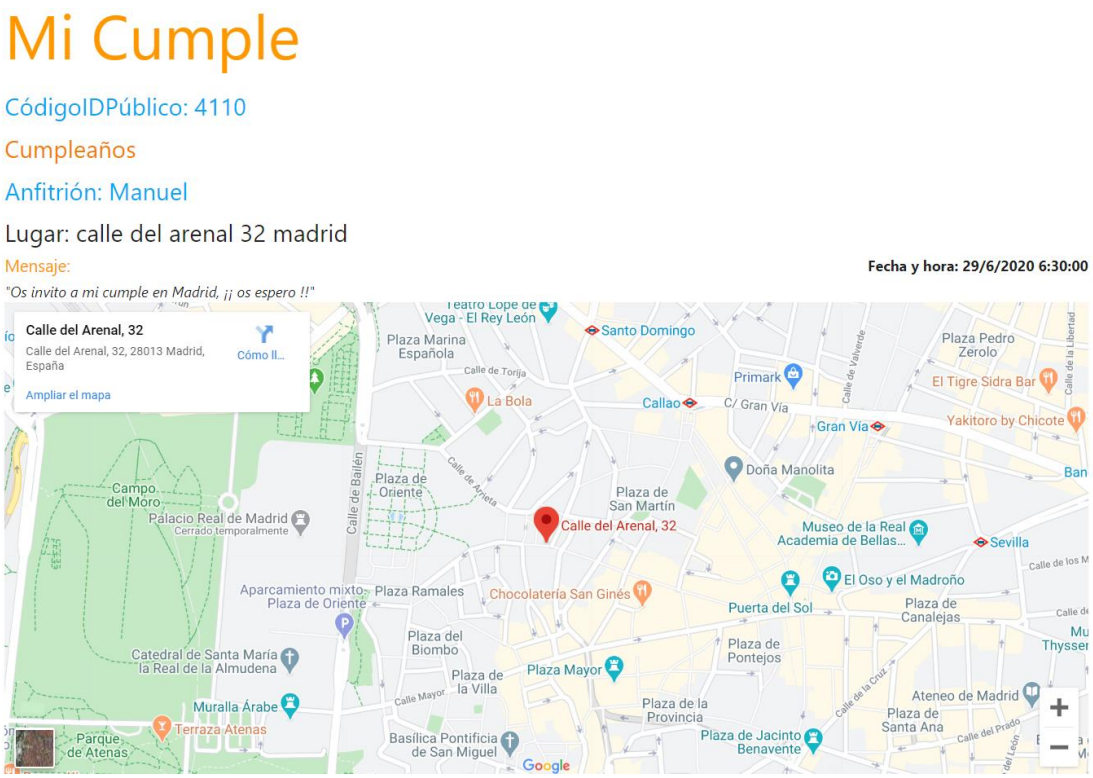

Ilustración 16 : Interfaz información evento

5.1.6 - Iteracción 6

Se desarrollará la aplicación para dispositivos móviles Android.

La aplicación Android se comunica con la API con ayuda de la librería Retrofit, que simplifica las peticiones GET y POST.

Para parsear la respuesta del servidor con Retrofit se utilizan anotaciones indicando el nombre del atributo en la BBDD.

```
public class Evento  implements Serializable {
    @SerializedName("type")
    @Expose
    private String type;
    @SerializedName("name")
    @Expose
    private String name;
    @SerializedName("message")
    @Expose
    private String message;
    @SerializedName("publicIdCode")
    @Expose
    private int publicIdCode;
    @SerializedName("date")
    @Expose
    private Date date;
    @SerializedName("dateCreated")
    @Expose
    private Date dateCreated;
    @SerializedName("place")
    @Expose
    private String place;
    @SerializedName("hostName")
    @Expose
    private String hostName;
    @SerializedName("guestsEmails")
    @Expose
    private String[] guestsEmails;
```

Ilustración 17 : parseo Evento Retrofit

En la aplicación móvil se pueden crear y ver información de eventos, ver (**Ilustración 18 : Información Evento Android**).

Ilustración 18 : Información Evento Android

Con la librería Butterknife se buscan las vistas con anotaciones, facilita el desarrollo ahorrando tiempo al programar ver(**Ilustración 19 : Libreria Butterknife**).

```
public class EventoDetail extends AppCompatActivity {
    @BindView(R.id.detalle_tipo)
    TextView tipo;
    @BindView(R.id.detalle_titulo)
    TextView tvTitulo;
    @BindView(R.id.detalle_codigo)
    TextView codigo;
    @BindView(R.id.detalle_Anfitrion)
    TextView anfitrion;
    @BindView(R.id.detalle_fecha)
    TextView fecha;
    @BindView(R.id.detalle_Lugar)
    TextView lugar;
    @BindView(R.id.detalle_descripcion)
    TextView descripcion;
    @BindView(R.id.detalle_fecha_creado)
    TextView fechaCreado;
```

Ilustración 19 : Libreria Butterknife

5.2 - Diagramas

5.2.1 - Diagrama Entidad-Relación:

En este diagrama detallamos la relación entre entidades de la aplicación.

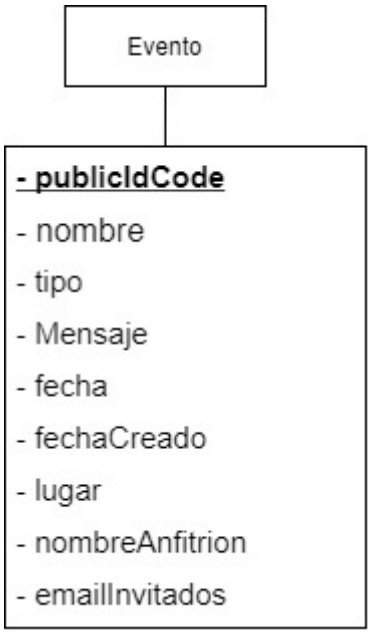

Ilustración 20 : Entidad-Relación

5.2.2 - Diagramas UML:

Clases

En este esquema se detallan los atributos y sus correspondientes tipos de dato de un Evento.

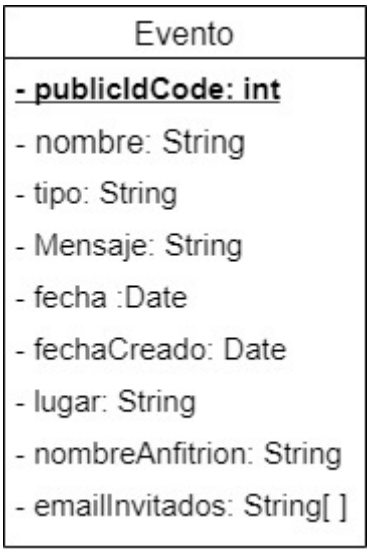

Ilustración 21 : Modelo Evento

6 - Conclusiones

En este punto se muestran las conclusiones extraídas del Proyecto a fin de que en un futuro se pueda mejorar.

6.1 - Desarrollo de la conclusión

La resolución extraída con la finalización del proyecto ha sido buena se ha seguido la planificación establecida aunque hubo una separación de los miembros del grupo.

Gracias a la elaboración de este proyecto se ha aprendido a trabajar en diferentes plataformas web y móvil compartiendo un servidor y base de datos comunes.

En las siguientes versiones de la app y web se mejorarán las funcionalidades y añadirán los módulos que se estimen convenientes.

7 - Referencias

[1] «Evolocion App Moviles,» [En línea]. Available: https://www.correosecommerce.com/blog/historia-aplicaciones-moviles/. [Último acceso: 15 Mayo 2020].

[2] [En línea]. Available: https://recyt.fecyt.es/index.php/EPI/article/viewFile/41718/23781.

[3] [En línea]. Available: https://sede.educacion.gob.es/publiventa/PdfServlet?pdf=VP18117.pdf&area=E.

[4] [En línea]. Available: https://es.wikipedia.org/wiki/Android_Studio.

[5] [En línea]. Available: https://developer.android.com/guide/components/fragments.

[6] [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html.

[7] [En línea]. Available: https://es.wikipedia.org/wiki/Node.js.

[8] [En línea]. Available: https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml#section_10.