

Proyecto Integrado CFGS



Desarrollo de Aplicaciones Multiplataforma

Campos Magro David



Listado de cambios	3
Introducción	3
Estudio de viabilidad	3
Descripción del sistema actual	3
Descripción del sistema nuevo	4
Identificación de Requisitos del Sistema	5
Requisitos de información	5
Requisitos funcionales	8
Otros requisitos	10
Descripción de la solución	10
Planificación del proyecto	11
Equipo de trabajo	11
Planificación temporal	11
Estudio del coste del proyecto	12
Análisis del Sistema de Información	13
Identificación del entorno tecnológico	13
Modelo de datos	14
Modelo Entidad-Relación	14
Esquema de la base de datos	14
Datos de prueba	15
Identificación de los usuarios participantes y finales	16
Identificación de subsistemas de análisis	16
Establecimiento de requisitos	17
Diagrama de Análisis	19
Definición de interfaces de usuarios	23
Especificación de principios generales de interfaz	23
Especificación de formatos individuales de la interfaz de pantalla	24
Identificación de perfiles de usuario	44
Especificación de formatos de impresión	47
Especificación de la navegabilidad entre pantallas	48
Construcción del sistema	49



Conclusiones	58
Glosario de términos	60
Bibliografía	61



1. Listado de cambios

Fecha	Cambios
18/03/2022	Establecimiento de requisitos iniciales
22/04/2022	Se opta por eliminar la entidad Músico y añadirla como un atributo de la entidad Usuario.
09/05/2022	Se piensa añadir un calendario de eventos y una nueva entidad para el mismo.
30/06/2022	Se opta por añadir una barra de búsqueda en distintas pantallas para facilitar el trabajo al usuario.

2. Introducción

El objetivo principal de esta aplicación es que facilite la gestión de partituras, noticias y eventos de la banda y la promoción de esta para posibles futuros contratos gracias a lo anteriormente mencionado.

3. Estudio de viabilidad

3.1. Descripción del sistema actual

Actualmente el sistema actual que existe en la banda de música para el reparto de partituras es bastante primitivo ya que si alguien ha perdido una partitura o no la tiene, se escribe un mensaje por el grupo de Whatsapp de la banda de música y en el próximo ensayo se le entrega en papel.



Para poder mostrar previos eventos y/o las obras musicales que contiene el repertorio, actualmente se realiza un pdf con el listado del repertorio y las redes sociales de la banda de música y se le envía al interesado.

3.2. Descripción del sistema nuevo

Como se ha dado a entender en el apartado anterior, no existe ninguna aplicación con la que se pueda facilitar lo ya mencionado. Al ver que se podría simplificar el reparto de partituras al poder descargarla directamente el músico.

Así mismo, también facilitaría el poder mostrar el repertorio, eventos y noticias de la banda a futuros contratantes.

Se dará solución a los siguientes problemas de la siguiente manera:

- Reparto de partituras: Cada músico registrado podrá descargar las partituras.
- Muestra del repertorio: La aplicación tendrá un apartado en el que aparecerá un listado de todo el repertorio de la banda de música.
- Muestra de eventos: La aplicación contará con un apartado con un calendario con los futuros eventos y anteriores en los que hemos participado.
- Noticias relevantes: Al iniciar la aplicación aparecerán todas las noticias de la banda tales como nuevos contratos, conciertos, etc.

Los datos serán almacenados en la base de datos no relacional Firebase usando también sus posibilidades de almacenamiento de archivos, registro y autenticación de usuarios.

La herramienta de desarrollo a usar será Visual Studio Code, ya que es de las más sencillas y cómodas para desarrollar una aplicación con el framework Ionic.

3.3. Identificación de Requisitos del Sistema



A continuación se describirán los requisitos en el sistema clasificándolos en las siguientes categorías:

- ❖ Requisitos de información: Será descrito toda la información que necesita ser almacenada en la base de datos para el uso de la aplicación.
- ❖ Requisitos funcionales: Especifican el funcionamiento de la aplicación con la interacción del usuario.

3.3.1. Requisitos de información

Usuario	
Descripción	La base de datos almacenará la información de cada usuario registrado.
Datos específicos	<ul style="list-style-type: none"> ➤ ID * ➤ Email. ➤ Nombre. ➤ Apellidos. ➤ Músico.
Volumen de información	Ilimitado
Observaciones	<ul style="list-style-type: none"> ➤ El campo Músico puede ser nulo. ➤ Puede haber dos tipos de usuarios: <ul style="list-style-type: none"> ○ Usuario → Si el campo Músico es nulo. ○ Músico → Si el campo Músico no es nulo.



Partitura	
Descripción	La base de datos almacenará el nombre de la partitura y el fichero pdf correspondiente.
Datos específicos	<ul style="list-style-type: none"> ➤ ID * ➤ Título. ➤ Autor. ➤ Fichero. ➤ Audio. ➤ Tipo.
Volumen de información	Ilimitado
Observaciones	El tipo puede ser: <ul style="list-style-type: none"> ➤ Marcha. ➤ BSO. ➤ Pasodoble. ➤ Pasacalles.

Noticia	
Descripción	La base de datos almacenará publicaciones de las nuevas noticias relacionadas con la banda de música.
Datos específicos	<ul style="list-style-type: none"> ➤ ID * ➤ Título. ➤ Contenido. ➤ Imagen. ➤ Fecha.
Volumen de información	Ilimitado
Observaciones	<ul style="list-style-type: none"> ➤ Una noticia puede también ser sustituida por un evento o viceversa.



Evento	
Descripción	La base de datos almacenará publicaciones de los eventos relacionados con la banda de música.
Datos específicos	<ul style="list-style-type: none"> ➤ ID * ➤ Title. ➤ StartTime. ➤ EndTime. ➤ AllDay.
Volumen de información	Ilimitado
Observaciones	<ul style="list-style-type: none"> ➤ Un evento puede también ser sustituido por una noticia o viceversa.

Usuarios Borrados	
Descripción	La base de datos almacenará un registro con el email y la fecha de los usuarios eliminados de la base de datos.
Datos específicos	<ul style="list-style-type: none"> ➤ Email. ➤ Fecha Borrado.
Volumen de información	Ilimitado
Observaciones	<ul style="list-style-type: none"> ➤ Se utilizará para llevar el control de que usuarios se han borrado y un administrador pueda eliminarlo del sistema de autenticación.



3.3.2. Requisitos funcionales

RF_01 – Registro de usuarios	
Descripción	La aplicación dará la opción de poder registrar un usuario nuevo para poder acceder a sus derechos.

RF_02 – Autenticación de usuarios	
Descripción	La aplicación dará la opción de poder autenticar a un usuario ya existente en la base de datos.

RF_03 – Ver noticias	
Descripción	La aplicación mostrará las nuevas noticias de la banda de música.

RF_04 – Ver eventos	
Descripción	La aplicación mostrará los anteriores y futuros eventos de la banda de música.

RF_05 – Ver repertorio	
Descripción	La aplicación mostrará todo el repertorio musical junto con un enlace a un audio/video del mismo.



RF_06 – Descargar partituras	
Descripción	La aplicación permitirá descargar a cada músico las partituras.

RF_07 – Subir partitura	
Descripción	Los administradores podrán subir nuevas partituras.

RF_08 – Gestión de usuarios	
Descripción	Los administradores podrán modificar los datos de un usuario, eligiendo su instrumento y categoría e incluso eliminar al usuario.

RF_09 – Gestión de noticias	
Descripción	Los administradores podrán crear, modificar los datos de una noticia y eliminarla.

RF_10 – Gestión de eventos	
Descripción	Los administradores podrán crear, modificar los datos de un evento y eliminarlo.



3.3.3. Otros requisitos

OR_01 – Conectividad	
Descripción	El dispositivo que esté usando la aplicación necesitará conexión a internet.

OR_02 – Galería	
Descripción	La aplicación debe de poder acceder a la galería del dispositivo para poder subir imágenes a las noticias.

OR_03 – Autenticación	
Descripción	La aplicación debe de poder registrar y autenticar a los usuarios a través de FirebaseAuth.

3.4. Descripción de la solución

La solución propuesta consiste en una aplicación en el framework de Ionic con Visual Studio Code en el que existan distintos usuarios que dependiendo de su categoría puedan realizar ciertas actividades tales como el poder descargar una partitura, crear y modificar los eventos y ver el repertorio.

Al iniciar la aplicación y haber logueado, esta se conecta a la base de datos de Firebase y mostrará las noticias de la banda de música.



3.5. Planificación del proyecto

3.5.1. Equipo de trabajo

La aplicación tiene como objetivo servir de proyecto de fin de curso para el CFGS de Desarrollo de Aplicaciones Multiplataforma impartido en el I.E.S Julio Verne de Sevilla y ayudar con ella a la banda de música San Sebastián de Villaverde del Río. Por lo tanto, el equipo de trabajo estará integrado por el alumno David Campos Magro.

3.5.2. Planificación temporal

Se ha fijado de plazo hasta el día 9 de junio de 2022. Durante ese tiempo, el desarrollo del proyecto se dividirá en distintas tareas:

Plazo	Tarea
22/04/2022	Análisis funcional y diseño de la solución.
17/06/2022	Finalización de la aplicación.
17/06/2022	Elaboración de la documentación

Los principales hitos son:

- 18/03/2022 → Entrega del anteproyecto.
- 22/04/2022 → Entrega de la documentación (Hasta el apartado 3.6)
- 17/06/2022 → Entrega del proyecto.



3.6. Estudio del coste del proyecto

El coste del proyecto sería gratuito ya que lo voy a hacer yo mismo para mi banda de música. Además el alojamiento de la base de datos será en firebase que es gratuito hasta 5GB y cómo se tratan de PDFs no ocupan demasiado espacio de almacenamiento y calculo que podrán ser almacenados todas las partituras. En caso que fuese necesario, al ser solo la aplicación para la propia banda, estimo que con 35 GiB, el cual es lo mínimo para el almacenamiento de datos y con 10GB de almacenamiento para los pdfs e imágenes, serían un total de 5.79€ al mes. A esto se le debería de añadir el precio por subir la aplicación si se quisiera a Google Play que son 23.15€.

La suma total del proyecto si fuese necesario sería de 23.15€ para subir la aplicación a Google Play una única vez y 5.79€ al mes para Firestore.



4. Análisis del Sistema de Información

4.1. Identificación del entorno tecnológico

Solo se requiere de una estación de trabajo para la realización del proyecto la cual tiene los siguientes componentes:

Hardware

- Procesador: AMD Ryzen 5 1600.
- 16GB de RAM DDR4.
- Almacenamiento:
 - 1TB de M.2.
 - 250GB SSD.
 - 1TB Disco duro.
 - 500GB Disco duro.
- Gráfica: GeForce RTX 2060.

Software

- Sistema Operativo: Windows 10 Professional 64 bits.
- IDE Visual Studio Code.
- Firebase de Google.



4.2. Modelo de datos

4.2.1. Modelo Entidad-Relación



4.2.2. Esquema de la base de datos

➤ Usuario

- ID → Varchar(45).
- Email → Varchar(45).
- Nombre → Varchar(45).
- Apellidos → Varchar(45).
- Músico → Músico.

➤ Partitura

- ID → Varchar(45).
- Título → Varchar(45).
- Autor → Varchar(45).
- Fichero → Varchar(45).
- Audio → Varchar(45).
- Comentario → Varchar(45).



➤ **Noticia**

- ID → Varchar(45).
- Título → Varchar(45).
- Contenido → Varchar(45).
- Imagen → Varchar(45).
- Fecha → Varchar(45).

➤ **Evento**

- ID → Varchar(45).
- Title → Varchar(45).
- StartTime → TimeStamp.
- EndTime → TimeStamp.
- AllDay → Boolean.

➤ **Usuarios Borrados**

- Email → Varchar(45).
- FechaBorrado → TimeStamp.

4.2.3. Datos de prueba

Se usará otra base de datos diferente a la oficial, aunque se usarán datos reales y algunos inventados como los usuarios o algunos eventos.



4.3. Identificación de los usuarios participantes y finales

Usuarios finales

- Usuario anónimo.
- Usuario registrado.
- Músico.
- Gestor.

Usuarios participantes

- Administrador.
- Firebase.

4.4. Identificación de subsistemas de análisis

Dentro de la aplicación se desarrollan distintos subsistemas y cada uno de ellos desempeña una función diferente. Aunque la gran mayoría son independientes, todos se basan en un subsistema base, el cual es la gestión de datos.

- **Gestión de usuarios.** Sus funciones son la creación y modificación y borrado de usuarios por parte de un usuario gestor para poder así confirmar que usuario es un músico perteneciente a la banda de música.
- **Gestión de noticias.** Engloba la creación, modificación y borrado de noticias por parte de un usuario gestor.
- **Gestión de partituras.** Al igual que las noticias sus funciones son la creación, modificación y borrado de partituras.
- **Gestión de eventos.** Al igual que las noticias y partituras, sus funciones son la creación, modificación y borrado de eventos.



4.5. Establecimiento de requisitos

Subsistema de gestión de usuarios	
Registrar usuario	Cualquier persona podrá registrarse como un nuevo usuario de la aplicación.
Modificar usuario	Un usuario registrado podrá modificar sus datos, y un usuario gestor podrá modificar los datos de cualquier usuario para confirmar su categoría y su instrumento.
Eliminar usuario	Un usuario administrador podrá eliminar cualquier usuario que sea necesario.

Subsistema de gestión de noticias	
Crear noticia	Un usuario gestor podrá crear una nueva noticia.
Modificar noticia	Un usuario gestor podrá crear una modificar una noticia.
Eliminar noticia	Un usuario gestor podrá eliminar una noticia.

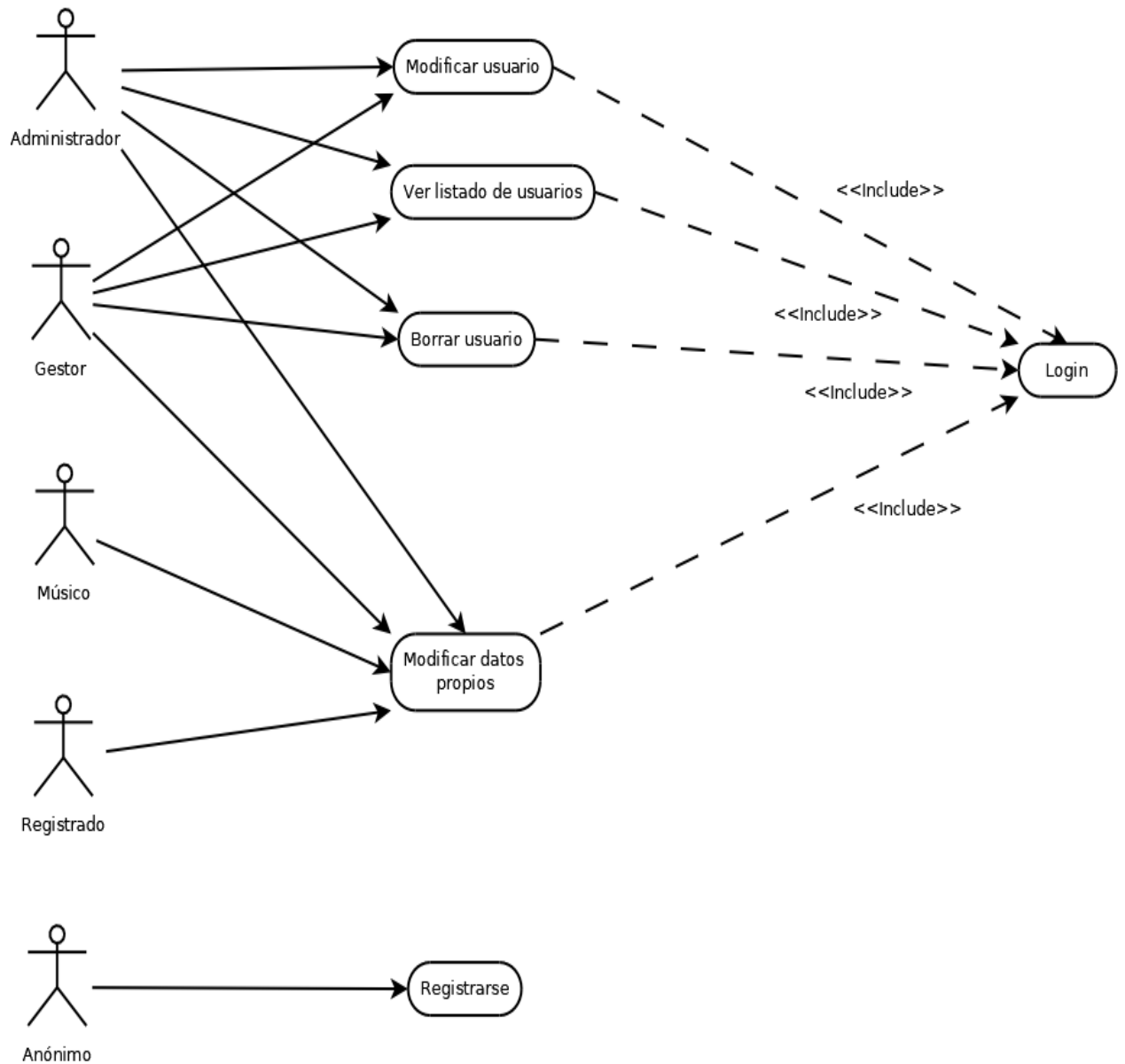


Subsistema de gestión de partituras	
Añadir partitura	Un usuario gestor podrá añadir una nueva partitura al repertorio.
Modificar partitura	Un usuario gestor podrá modificar los datos de una partitura del repertorio.
Eliminar partitura	Un usuario gestor podrá eliminar los datos de una partitura del repertorio.

Subsistema de gestión de eventos	
Añadir evento	Un usuario gestor podrá añadir un nuevo evento.
Modificar evento	Un usuario gestor podrá modificar los datos de un evento ya existente.
Eliminar evento	Un usuario gestor podrá eliminar los datos de un evento creado anteriormente.

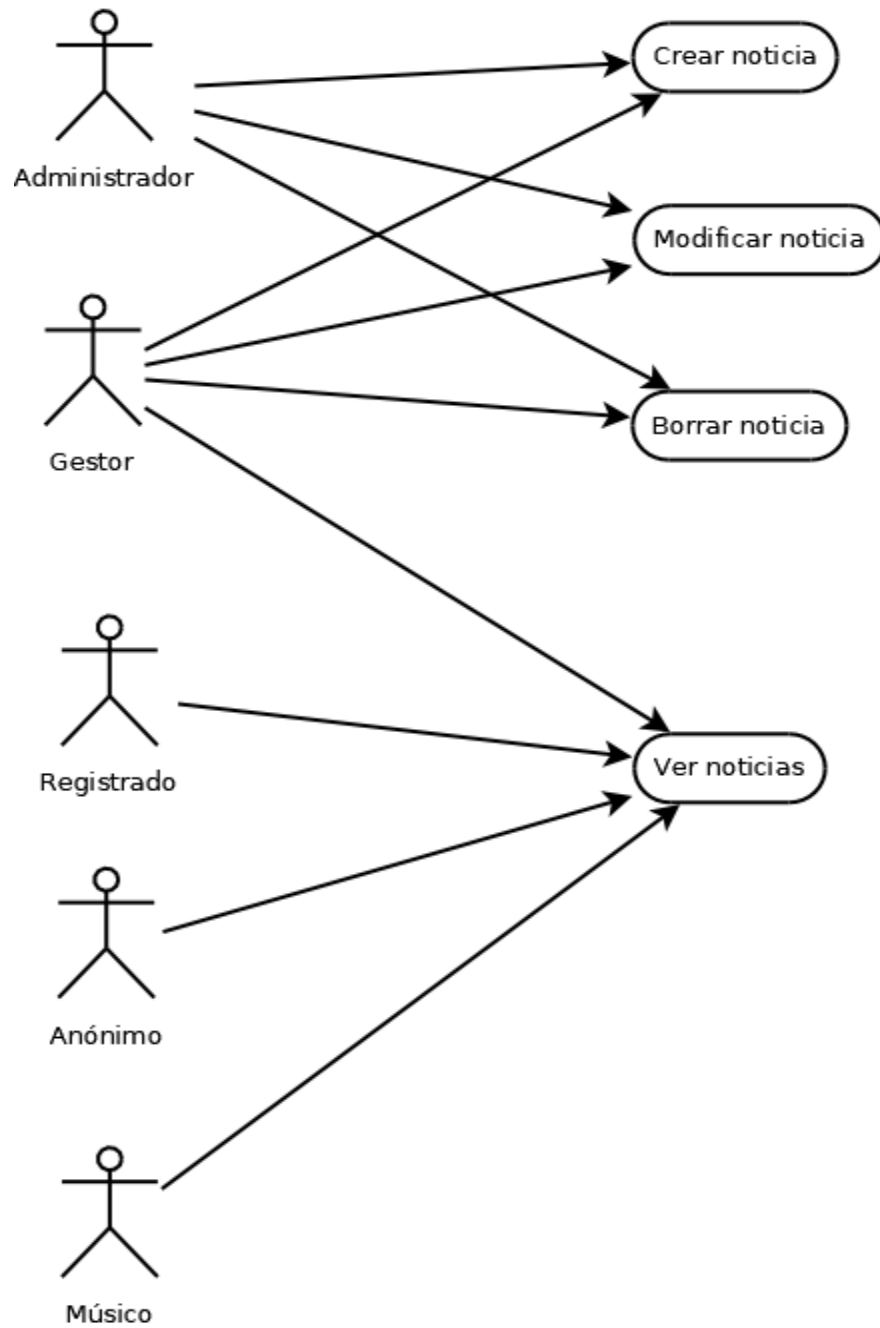


4.6. Diagrama de Análisis

> Subsistema de gestión de usuarios

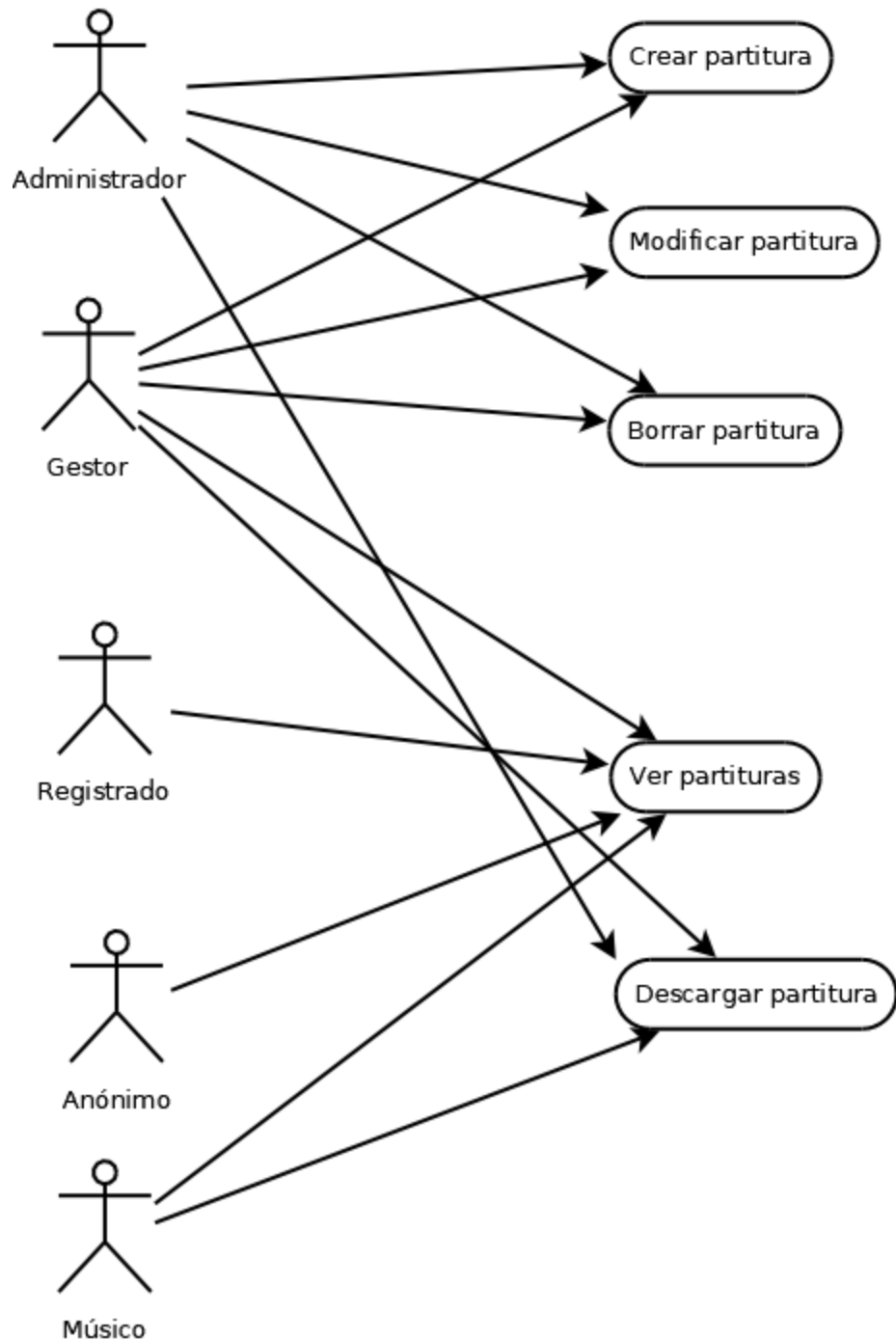


➤ Subsistema de gestión de noticias



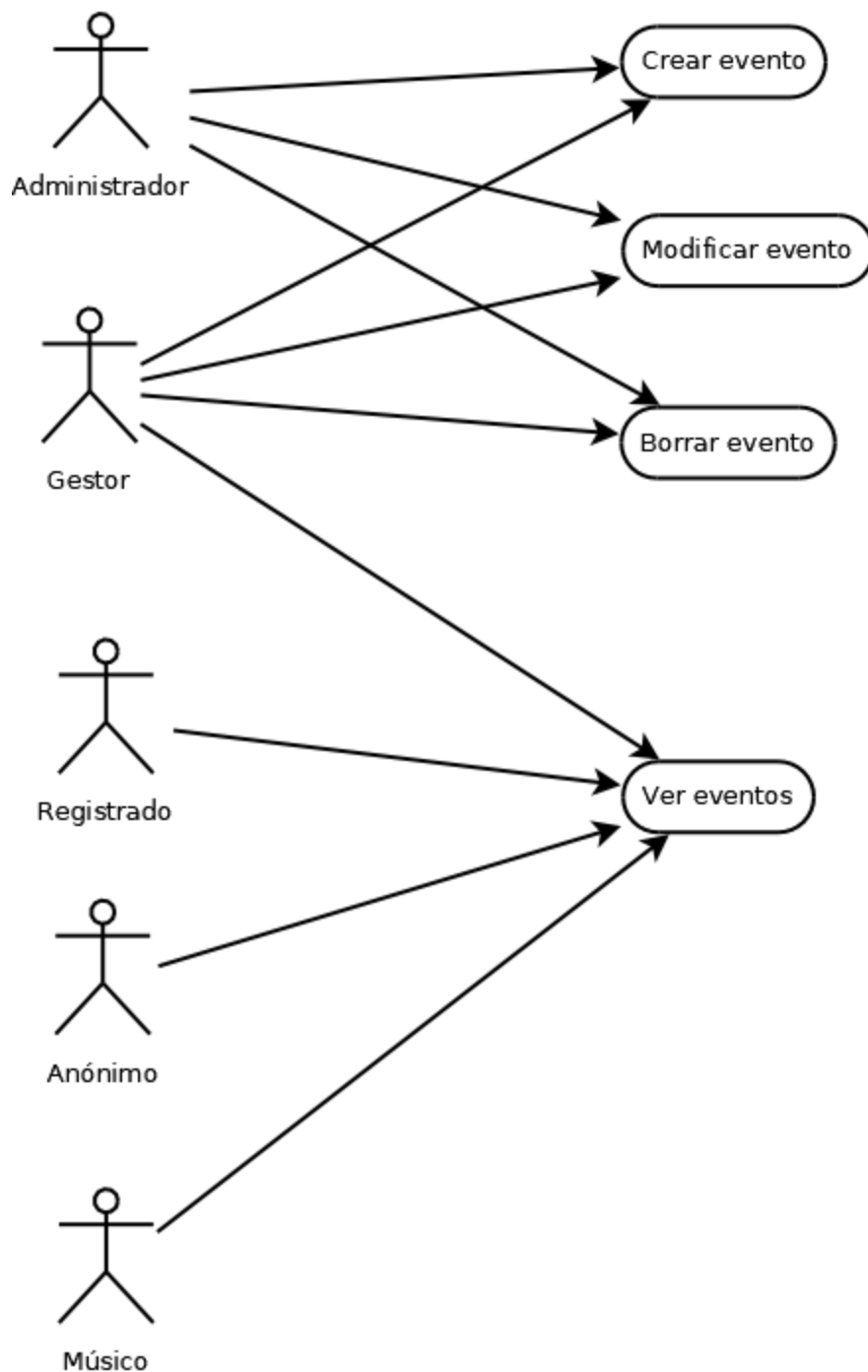


➤ Subsistema de gestión de partituras





➤ Subsistema de gestión de eventos











4.7. Definición de interfaces de usuarios

4.7.1. Especificación de principios generales de interfaz

Todas las vistas de la aplicación comparten un esquema de colores comunes con el fin de crear una estética uniforme, que además van acorde con los colores de la propia banda de música.

Colores

-  **#008c2b** → Color usado para iconos, botones y etiquetas de formularios.
-  **#1f1f1f** → Color usado para el fondo de toolbar y menú.
-  **#161616** → Color usado para el fondo de la aplicación.
-  **#1e1e1e** → Color usado para el fondo de los ion-cards.
-  **#ffffff** → Color de textos principales como títulos, nombre de la página, etc.
-  **#9d9d9d** → Color para el resto de textos.

Tipografías

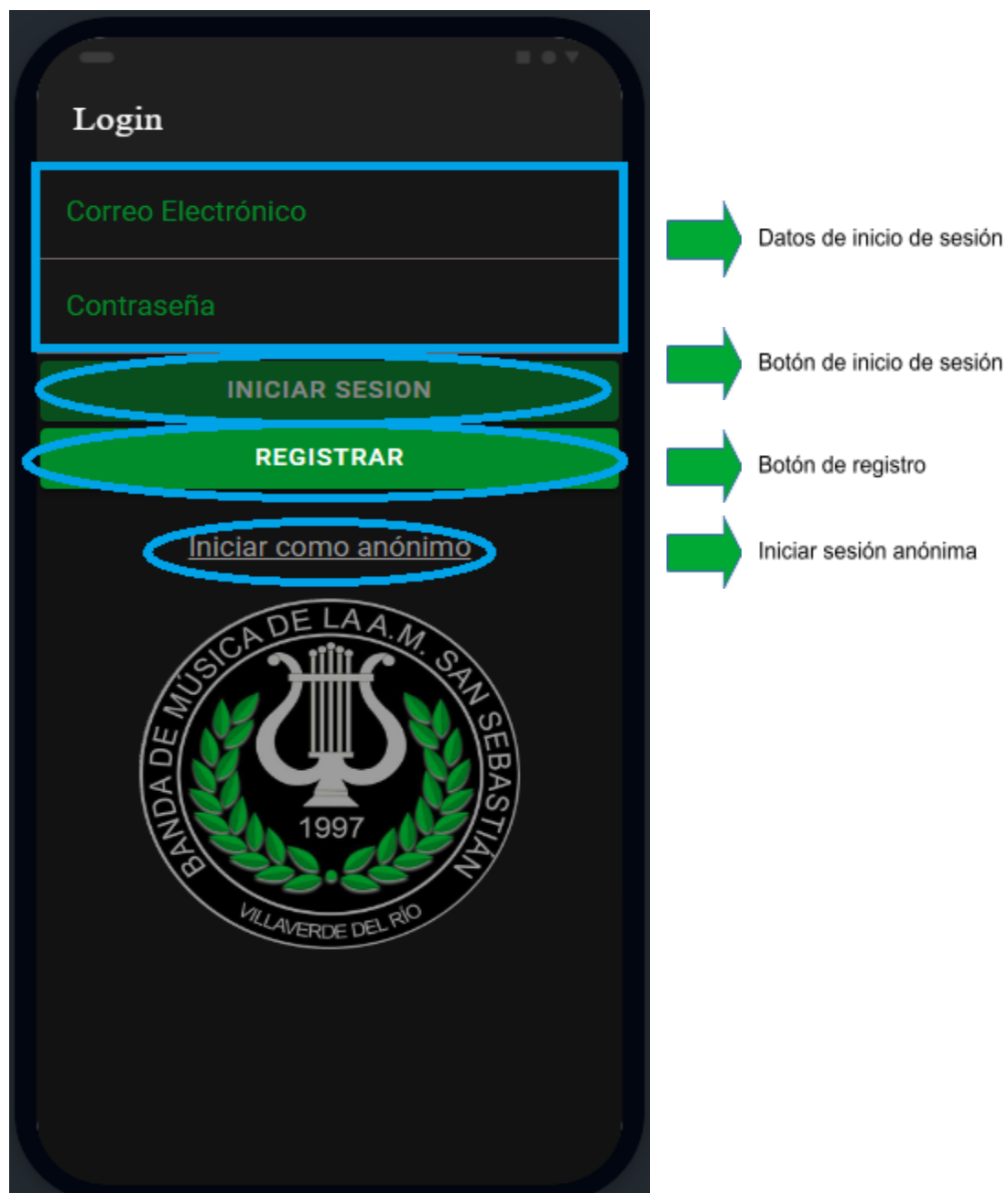
Existen dos tipografías para la aplicación:

- **Garamond** → Se usa para cualquier título de la aplicación.
- **Roboto** → Utilizado para el resto de textos.



4.7.2. Especificación de formatos individuales de la interfaz de pantalla

➤ Pantalla de login.





➤ Pantalla de registro.



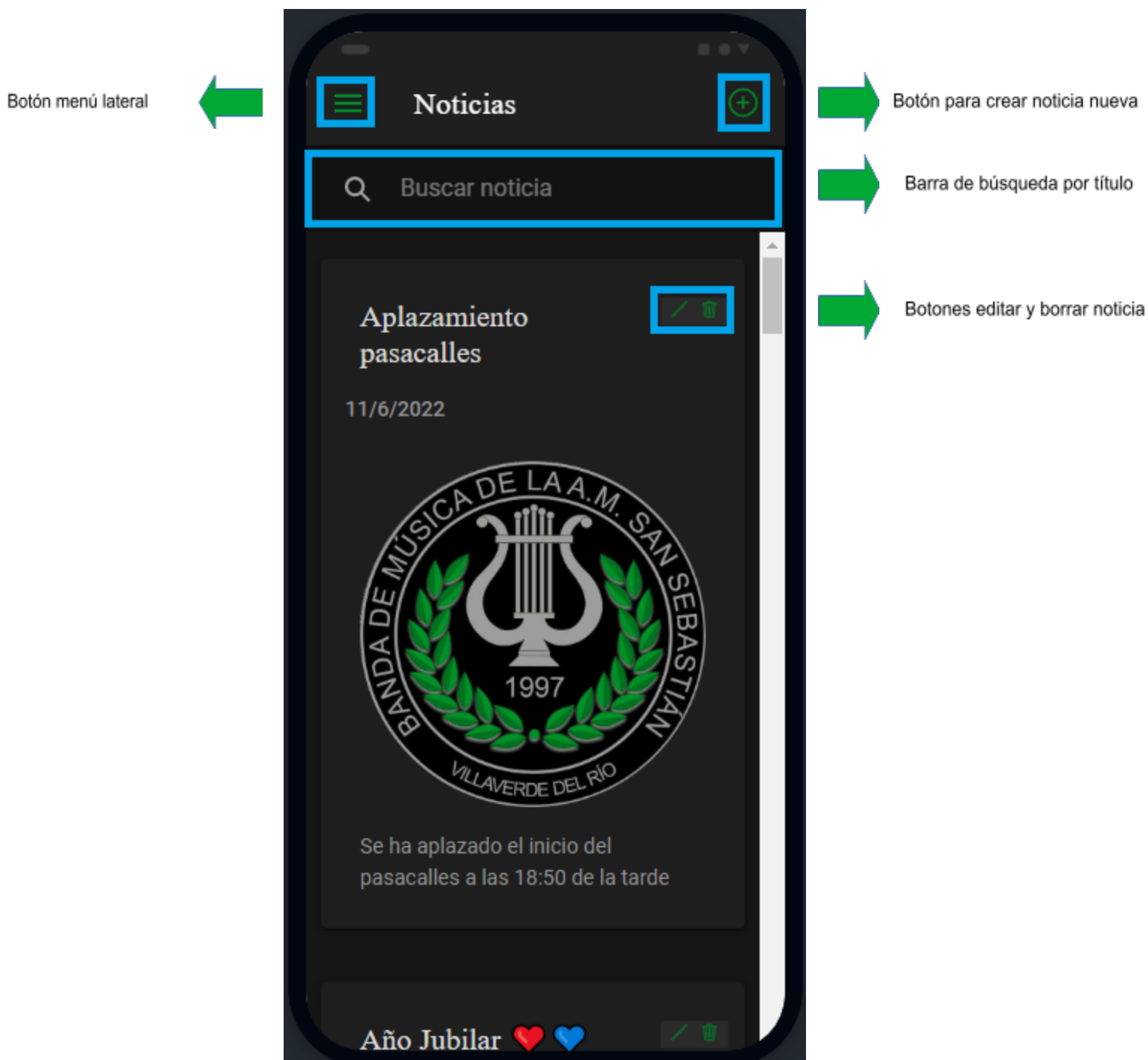
Datos de registro



Botón de registro



- Pantalla de noticias.
 - Administrador.





○ Anónimo/Registrado/Músico.

Botón menú lateral



Barra de búsqueda por título



➤ Crear/Modificar noticia.

- Administrador.

Volver atrás



Año Jubilar ❤️💙

Año Jubilar ❤️💙

Mañana Sábado 11 de Junio será un día histórico para nuestro pueblo de Villaverde del Día, pero también

Seleccionar archivo Ninguno archivo

LISTO



Contenido noticia



Botón subir imagen



Botón confirmar



- Menú.
 - Administrador.

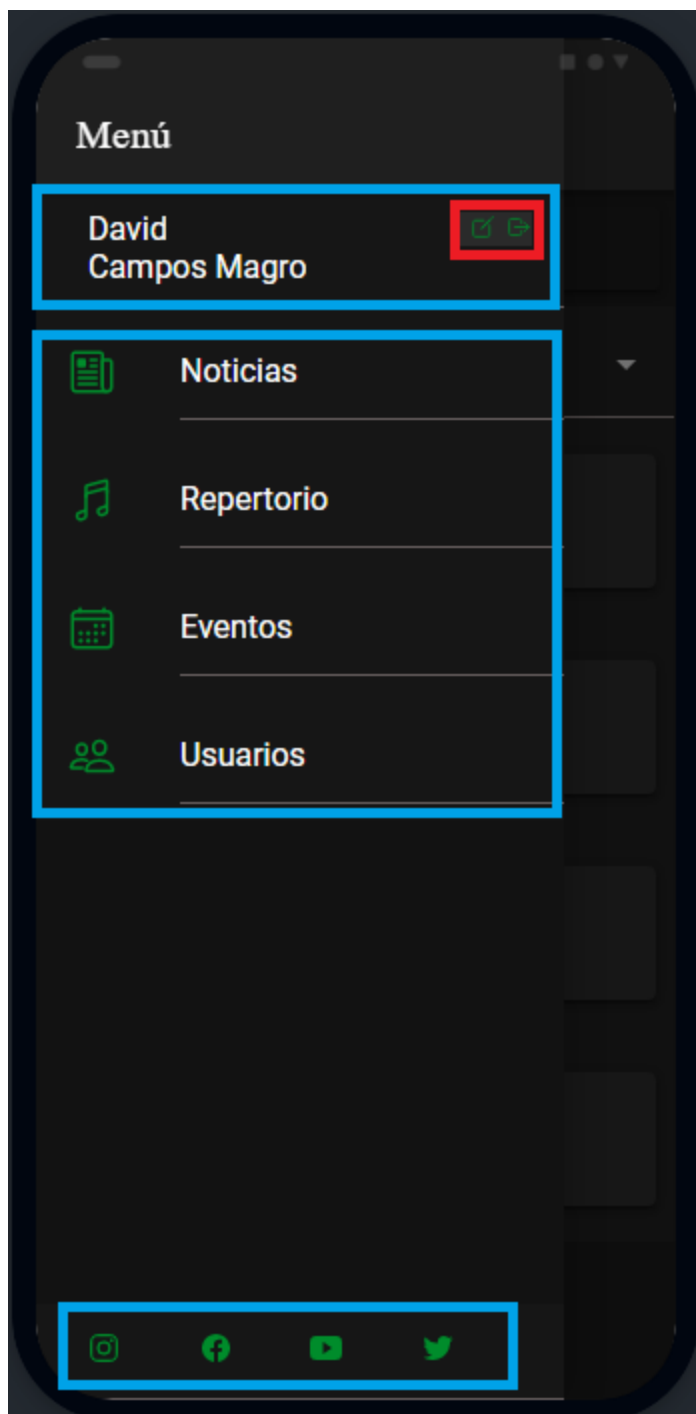
Datos del usuario actual



Opciones del menú



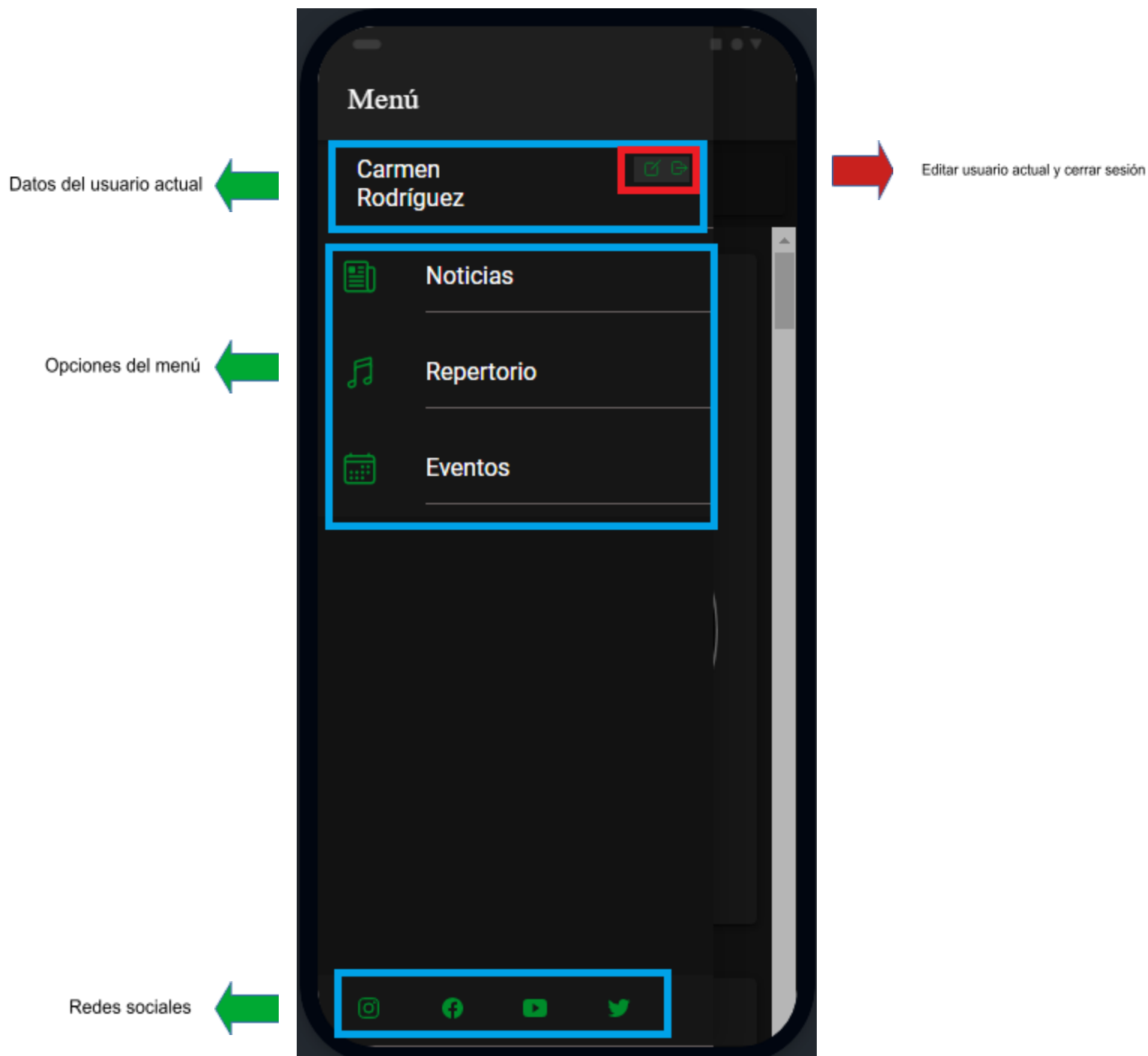
Redes sociales



Editar usuario actual y cerrar sesión

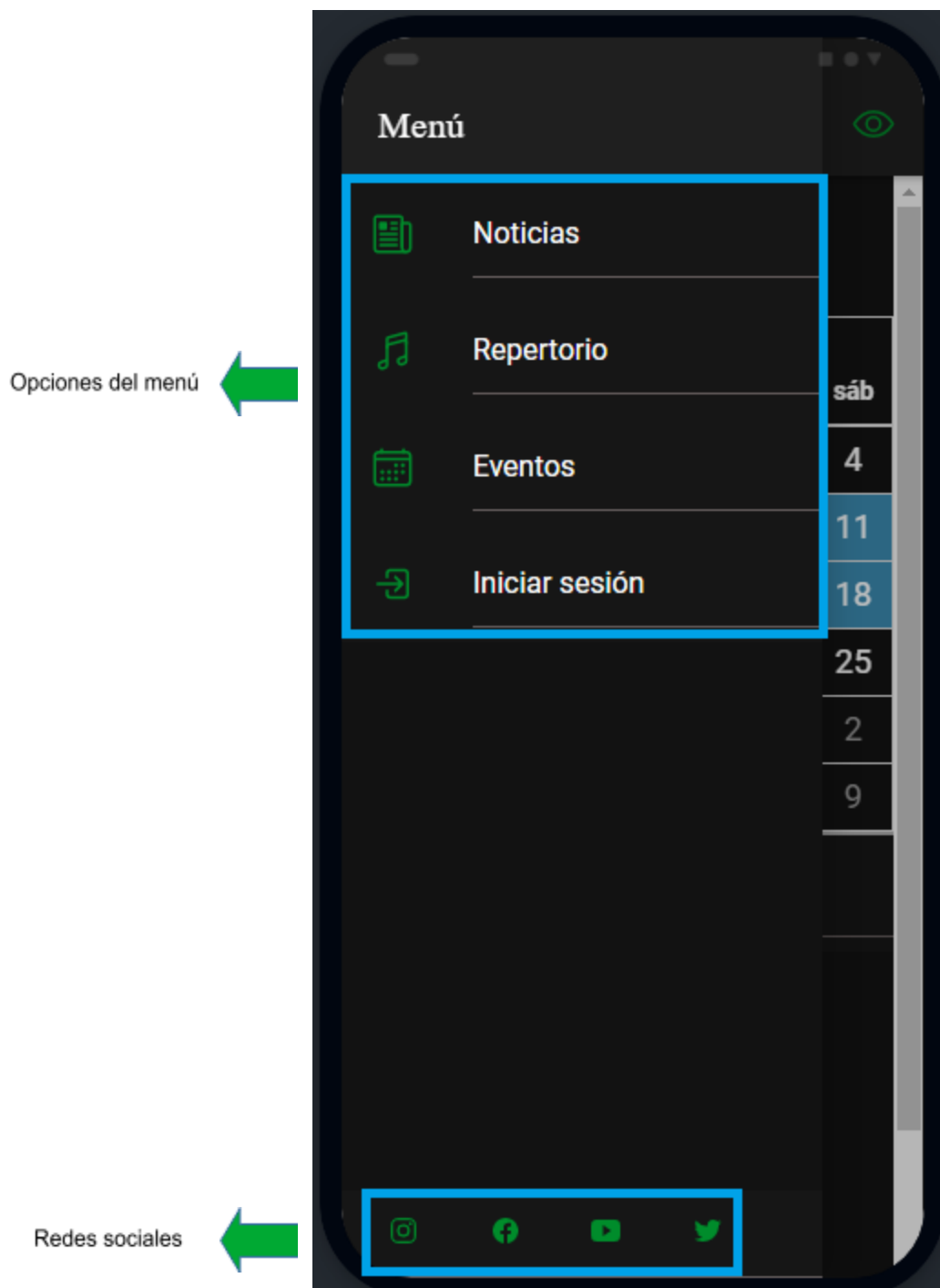


- Músico/Registrado.





○ Anónimo





➤ Modificar usuario.

- Administrador.

Volver atrás



Campos usuario



Campos músico



Botón confirmar





- Registrado no administrador.

Volver atrás

Campos usuario

Botón confirmar

Carmen Rodríguez

Nombre
Carmen

Apellidos
Rodríguez

LISTO



➤ Repertorio

- Administrador.

The screenshot shows the 'Repertorio' app interface. The title 'Repertorio' is at the top. Below it is a search bar with the placeholder text 'Buscar en el repertorio'. A dropdown menu labeled 'Tipo' is set to 'Todo'. Below the menu is a list of items, each with a title and author, and a set of action icons (play, download, edit, delete). The annotations point to the following elements:

- Botón menú**: Points to the hamburger menu icon in the top left corner.
- Barra de búsqueda**: Points to the search bar.
- Filtrador por tipo**: Points to the 'Tipo' dropdown menu.
- Botón crear partitura nueva**: Points to the '+' icon in the top right corner.
- Botones reproducir vídeo, descargar partitura, editar partitura y borrarla**: Points to the action icons (play, download, edit, delete) for the first item.



○ Música.

Botón menú

Barra de búsqueda

Filtrador por tipo

Repertorio

Buscar en el repertorio

Tipo
Todo

A mi virgen de aguas santas - Antonio Torres Vergara

A ti... Manué - Juan José Puntas

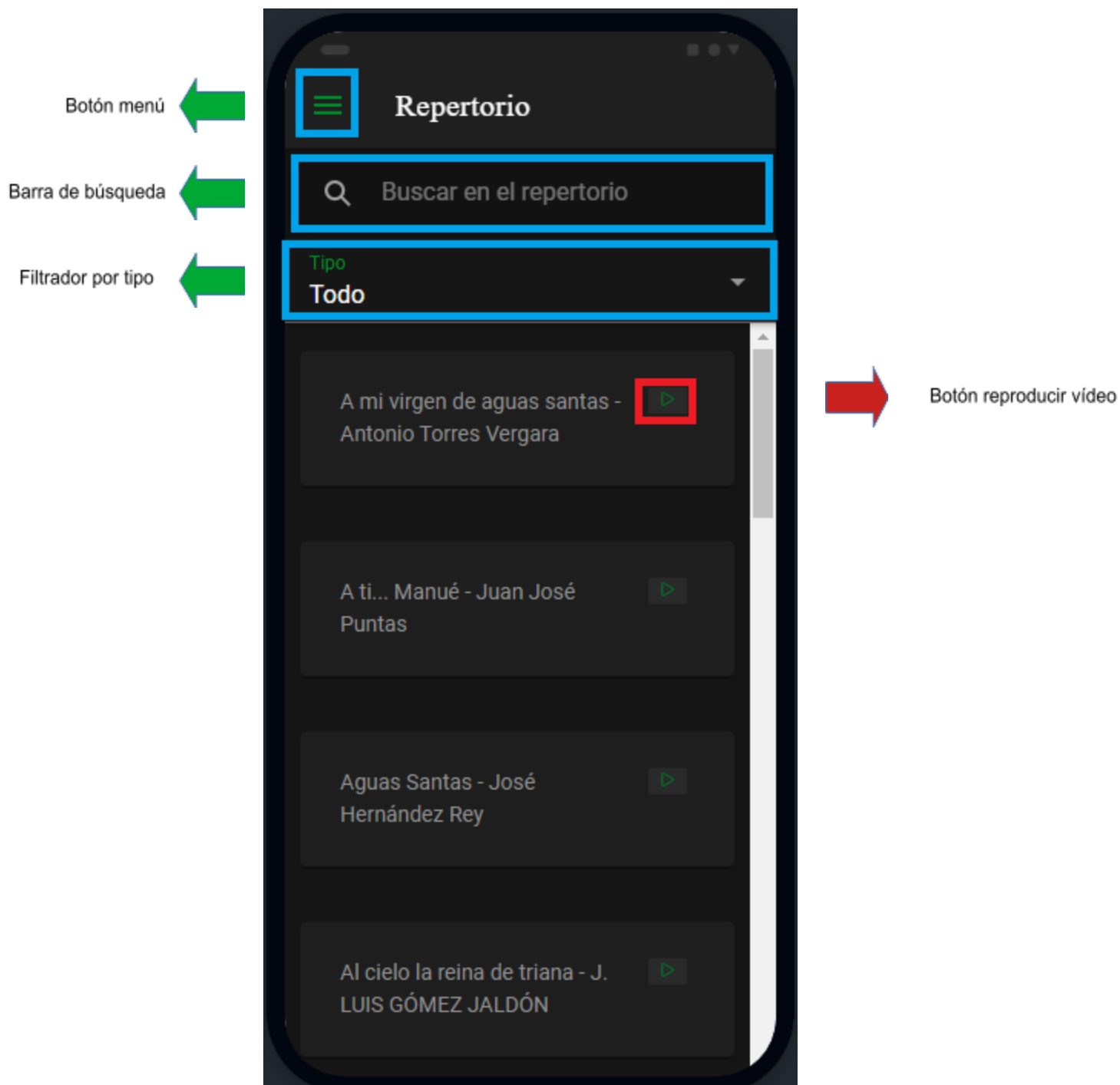
Aguas Santas - José Hernández Rey

Al cielo la reina de triana - J. LUIS GÓMEZ JALDÓN

Botones reproducir vídeo y descargar partitura



- Registrado o Anónimo.





➤ Añadir o editar partitura.

- Administrador.

Volver atrás



A mi virgen de aguas santas

Titulo
A mi virgen de aguas sant

Autor
Antonio Torres Vergara

Enlace a audio
<https://youtu.be/N3locEA>

Tipo
Marchas

Seleccionar archivo Ninguno archivo selec.

LISTO

Datos partitura



Botón subir pdf



Botón confirmar





➤ Eventos.

- Administrador.

Botón menú

Botones ver listado de eventos y crear evento nuevo

Día actual

Día seleccionado

Día con evento

Descripción del evento

do m	lun	mar	mié	jue	vie	sáb
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

19:00 - 05:00 | Año Jubilar ❤️💙



- Músico, Registrado o Anónimo.

The screenshot shows a mobile application interface for 'Eventos' (Events). At the top, there is a header with the title 'Eventos'. On the left side of the header is a menu button (three horizontal lines), and on the right side is a button with an eye icon. Below the header is a calendar for 'junio 2022'. The calendar grid shows days of the week (do m, lun, mar, mié, jue, vie, sáb) and dates. The date '11' is highlighted with a red box, and the date '13' is highlighted with a yellow box. Below the calendar is a section titled 'No Events'. Annotations with arrows point to various elements: 'Botón menú' points to the menu button; 'Botones ver listado de eventos y crear evento nuevo' points to the eye icon button; 'Día con evento' points to the date '11'; 'Día actual' points to the date '13'; and 'Descripción del evento' points to the 'No Events' section.

Botón menú

Botones ver listado de eventos y crear evento nuevo

Día con evento

Día actual

Descripción del evento

No Events



➤ Listado de eventos.

- Administrador.

Volver atrás

Barra de búsqueda

Botones de editar y borrar

Listado de Eventos

Buscar evento

Año Jubilar ❤️💙

Inicio: 11/6/2022, 19:00:00
Fin: 12/6/2022, 5:00:00

Corpus Christis

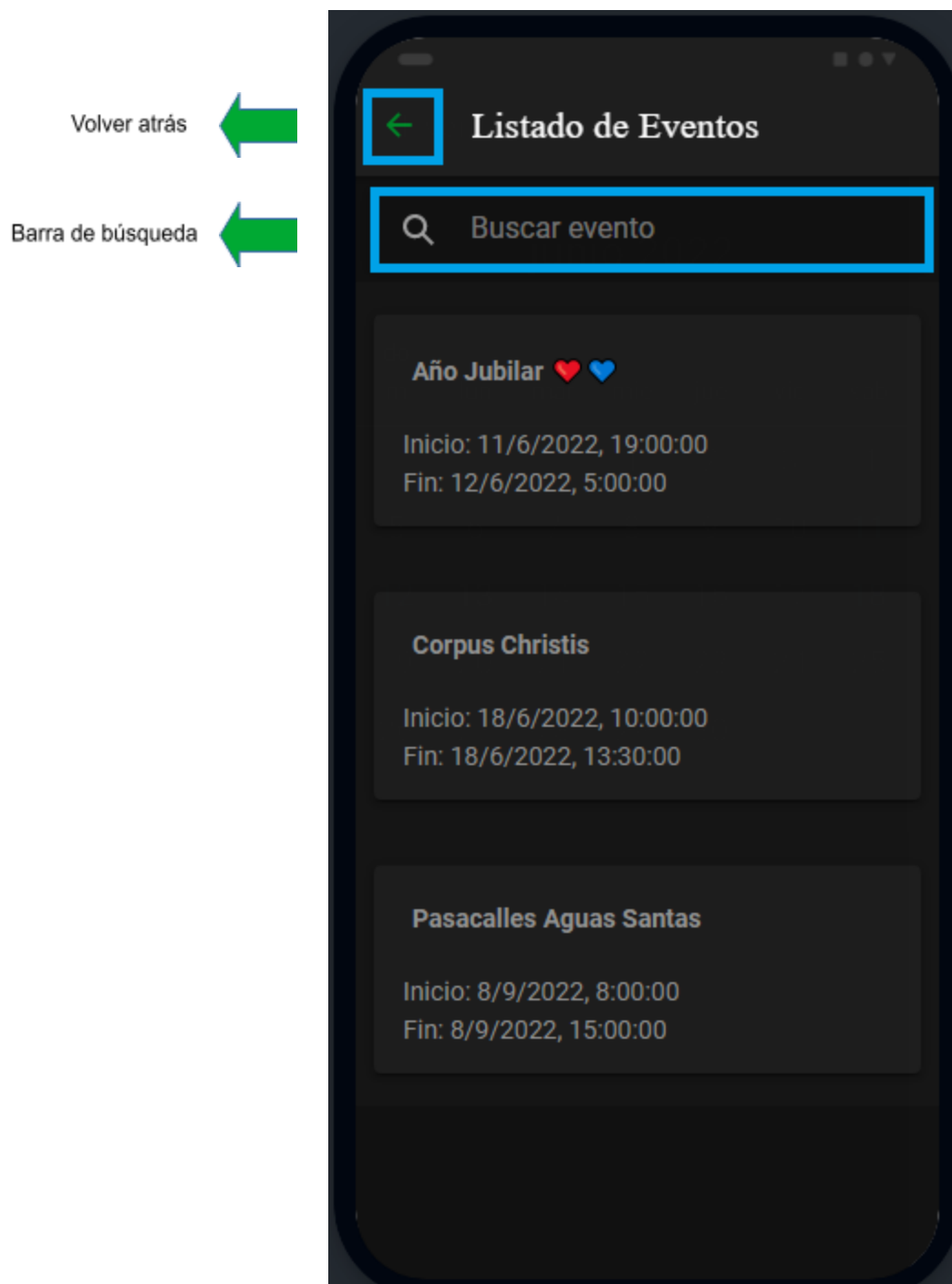
Inicio: 18/6/2022, 10:00:00
Fin: 18/6/2022, 13:30:00

Pasacalles Aguas Santas

Inicio: 8/9/2022, 8:00:00
Fin: 8/9/2022, 15:00:00



- Músico, Registrado o Anónimo.





➤ Añadir o editar evento.

- Administrador.

Volver atrás



Nuevo Evento

Título

Inicio

Time 22:04

junio de 2022

D	L	M	X	J	V	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Final

Time 22:04

junio de 2022

D	L	M	X	J	V	S

Botón confirmar



Datos del evento





➤ Usuarios.

- Administrador.

Botón menú

Barra de búsqueda

Filtrador por perteneciente a la banda o no

Nombre	Perteneciente a la banda
Carmen Rodríguez	✓
David Campos Magro	✓
Elena Rodríguez	✗
Pablo Haro Escobar	✗

Pertenece a la banda

No pertenece a la banda



4.7.3. Identificación de perfiles de usuario

Hay en total 5 tipos de usuarios, los cuales podrán acceder a distintas pantallas para poder ver o hacer distintas cosas tales como creación de noticias, descargar partituras, etc.

Cada tipo de usuario podrá acceder a las siguientes vistas:

➤ **Administrador:**

- **Sistema de usuarios.**
 - Login.
 - Modificar datos propios.
 - Modificar músico.
 - Ver listado de usuarios.
 - Borrar usuario.
- **Sistema de noticias.**
 - Ver noticias.
 - Crear noticia.
 - Modificar noticia.
 - Borrar noticia.
- **Sistema de partituras.**
 - Ver repertorio.
 - Ver vídeo.
 - Descargar partitura.
 - Crear partitura.
 - Modificar partitura.
 - Borrar partitura.
- **Sistema de eventos.**
 - Ver eventos.
 - Crear evento.
 - Modificar evento.
 - Borrar evento.



➤ **Gestor:**

- **Sistema de usuarios.**
 - Login.
 - Modificar datos propios.
 - Modificar músico.
 - Ver listado de usuarios.
 - Borrar usuario.
- **Sistema de noticias.**
 - Ver noticias.
 - Crear noticia.
 - Modificar noticia.
 - Borrar noticia.
- **Sistema de partituras.**
 - Ver repertorio.
 - Ver vídeo.
 - Descargar partitura.
 - Crear partitura.
 - Modificar partitura.
 - Borrar partitura.
- **Sistema de eventos.**
 - Ver eventos.
 - Crear evento.
 - Modificar evento.
 - Borrar evento.



➤ **Músico:**

- **Sistema de usuarios.**
 - Login.
 - Modificar datos propios.
- **Sistema de noticias.**
 - Ver noticias.
- **Sistema de partituras.**
 - Ver repertorio.
 - Ver vídeo.
 - Descargar partitura.
- **Sistema de eventos.**
 - Ver eventos.

➤ **Usuario Registrado:**

- **Sistema de usuarios.**
 - Login.
 - Modificar datos propios.
- **Sistema de noticias.**
 - Ver noticias.
- **Sistema de partituras.**
 - Ver repertorio.
 - Ver vídeo.
- **Sistema de eventos.**
 - Ver eventos.

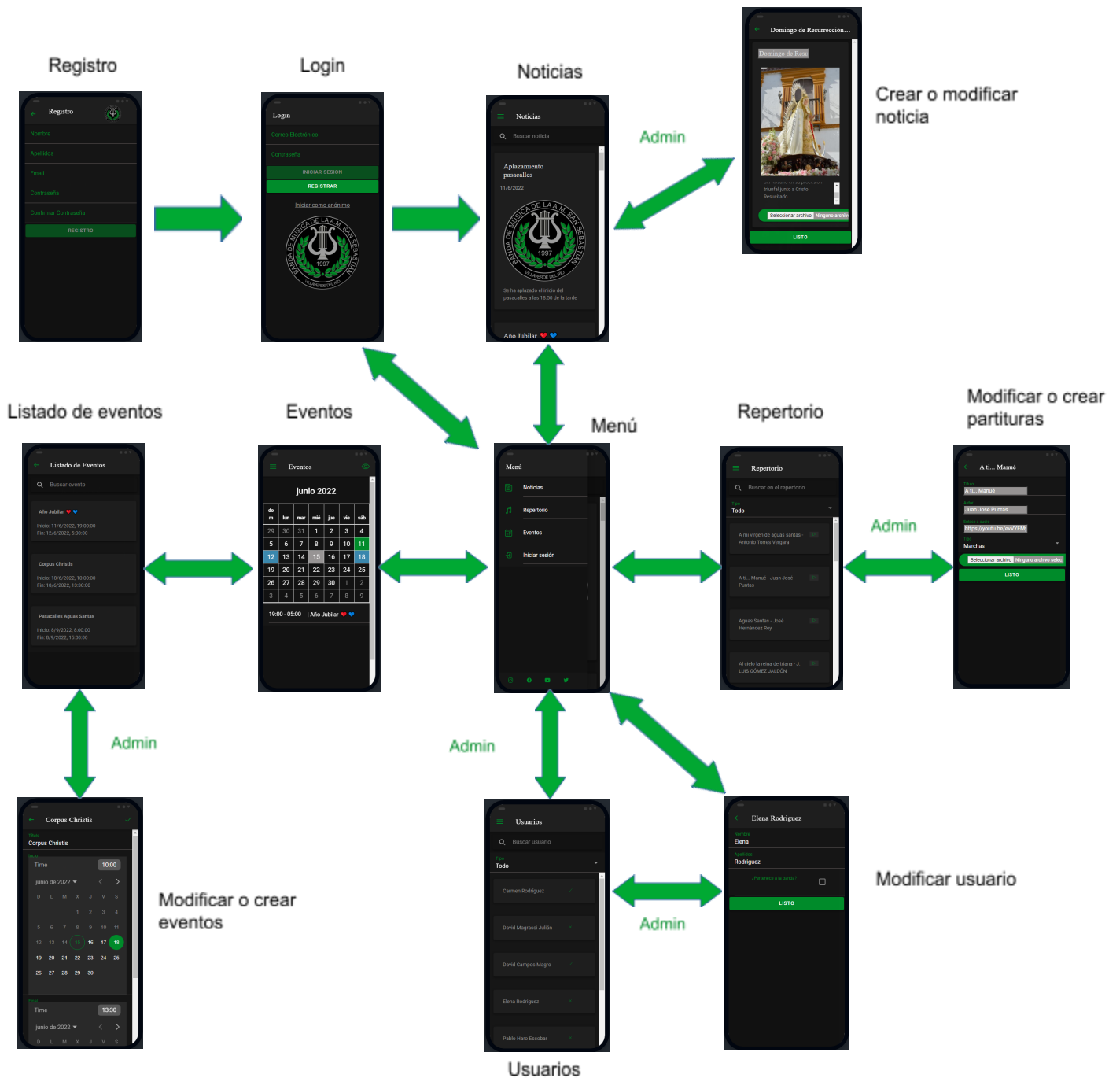


➤ **Usuario Anónimo:**

- **Sistema usuarios.**
 - Registrarse.
- **Sistema noticias.**
 - Ver noticias.
- **Sistema partituras.**
 - Ver repertorio.
 - Ver vídeo.
- **Sistema eventos.**
 - Ver eventos.

4.7.4. Especificación de formatos de impresión

La aplicación no generará documentos de ningún tipo de formato, solo mostrará la información por pantalla y permitirá descargar pdfs.





5. Construcción del sistema

➤ Módulo Calendar

El proyecto cuenta con un calendario el cual es un módulo existente dentro de Ionic 2, hecho por Twinssbc.

Para poder usar el módulo hay que instalarlo en el proyecto con el siguiente comando: ***npm install ionic2-calendar --save*** .

Una vez instalado en el proyecto, hay que importar ***CalendarModule*** y ***NgCalendarModule*** en *app.module.ts* . A posteriori en cada página que se desee usar el calendario hay que importar en su módulo ***NgCalendarModule*** . Para poder usar el calendario en nuestro idioma tenemos que importar varias cosas más en el módulo donde se ubique nuestro calendario:

```
import { registerLocaleData } from '@angular/common';  
  
import localeEs from '@angular/common/locales/es'  
  
registerLocaleData(localeEs)
```

y en el apartado providers:

```
providers: [  
  
  {provide: LOCALE_ID, useValue: 'ES-es'}  
  
]
```



También he de añadir que en el código de cada página donde se use el calendario hay que poner la siguiente anotación encima del constructor:

```
@ViewChild(CalendarComponent) myCal: CalendarComponent;
```

Una vez concluido los siguientes pasos, ya está listo para usar el calendario en nuestro proyecto, ahora voy a explicar como poder utilizarlo correctamente con firebase.

Lo primero es preparar la vista (html) con la siguiente etiqueta con los atributos necesarios, incluido el utilizado para traer el evento desde firebase:

```
<calendar  
  [eventSource]="eventos"  
  [calendarMode]="calendario.modos"  
  [currentDate]="calendario.fechaActual"  
  (onTitleChanged)="tituloCambiado($event)"  
></calendar>
```



A continuación en el código de la misma página tenemos que crear los atributos y métodos necesarios para hacerlo funcionar:

```
//|Atributos|

private eventos = [];

private titulo: string;

private calendario = {

  modo: 'month',

  fechaActual: new Date(),

};

tituloCambiado(titulo) {

  this.titulo = titulo;

} //end tituloCambiado

//|Firebase|

getEventos() {

  this.db

    .collection(`Eventos`)

    .snapshotChanges()

    .subscribe((colSnap) => {

      this.eventos = [];

      colSnap.forEach((snap) => {
```



```
const event: any = snap.payload.doc.data();

event.id = event.id;

event.startTime = event.startTime.toDate();

event.endTime = event.endTime.toDate();

this.eventos.push(event);});});});

//end getEventos
```

A continuación pondré un ejemplo de como crear un evento nuevo y subirlo a firebase. Para ello si la fecha se quiere ser seleccionada a través de un *ion-datetime* (etiqueta de ionic) es necesario parsear y convertir el valor recogido a Date ya que esta etiqueta solo funciona con strings.

Ejemplo:

```
//|Atributos|

@Input() eventoJson;

private minDate = new Date().toISOString();

private eventos = [];

private calendario = {

  modo: 'month',

  fechaActual: new Date(),

};

private evento: Evento = new Evento();

private eventoNuevo: Evento = new Evento();
```



```
private inicio: string;

private fin: string;

@ViewChild(CalendarComponent) myCal: CalendarComponent;

constructor(

    private modalCtrl: ModalController,

    private fireService: FireServiceProvider,

    private datePipe: DatePipe,

    private db: AngularFirestore,

    private menu: MenuController

) {} //end constructor

//|Fases Ionic|

ngOnInit() {

    this.evento =

Evento.createFromJsonObject(JSON.parse(this.eventoJson));

    this.eventoNuevo = this.evento;

    this.transformar(this.evento.startTime, true);

    this.transformar(this.evento.endTime, false);

} //end ngOnInit
```



```
//|Firebase|

private aceptar() {

    this.eventoNuevo.startTime = new Date(this.inicio);

    this.eventoNuevo.endTime = new Date(this.fin);

    //Es necesario que sea una clase anónima porque no se
    suben a la bbdd correctamente las fechas y por lo tanto
    tampoco los recoge bien

    const event = {

        id: '',

        title: this.eventoNuevo.title,

        startTime: this.eventoNuevo.startTime,

        endTime: this.eventoNuevo.endTime,

        allDay: false,

    };

    event.id =
this.db.collection('Eventos').ref.doc().id;

this.db.collection('Eventos').doc(event.id).set(event);

} //end aceptar

//|Otros métodos|
```



```
//Formatea la fecha con DatePipe para que la clase
Date la reciba bien y pueda ser usada en los dateTime

transformar(evento, inicio: boolean) {

  let arrayEventos = [];

  arrayEventos = evento.split('/');

  let eventoFormatear =

    arrayEventos[1] + '/' + arrayEventos[0] + '/' +
arrayEventos[2];

  let formato = this.datePipe.transform(

    eventoFormatear,

    'YYYY-MM-ddTHH:mm:ssZ'

  );

  let final = '';

  let principio = '';

  let bool = false;

  for (let inx: number = 0; inx < formato.length;
inx++) {

    if (formato[inx] == '+') {

      bool = true;

    }

  }

}
```




```
    if (bool) final += formato[inx];  
  
    else principio += formato[inx];  
  
  }  
  
  if (inicio)  
  
    this.inicio = principio + final[0] + final[1] +  
final[2] + ':00';  
  
    else this.fin = principio + final[0] + final[1] +  
final[2] + ':00';  
  
  } //end transformar
```

➤ Filtro de Ion SearchBar

El proyecto cuenta con varias barras de búsqueda en distintas vistas de la aplicación para facilitar la búsqueda de usuarios, partituras, noticias y eventos.

Para poder usar el filtro tenemos que crearnos un módulo y un pipe vinculado a dicho módulo.

Una vez vinculado, quedaría hacer la lógica de filtrado, importar el módulo en **app.module.ts** y en cada página que se vaya a utilizar y vincularlo con los respectivos htmls.

En sí lo que hace el filtrador es recoger el array indicado, buscar por el atributo indicado con el valor indicado y devolver un array con los objetos encontrados.



Ejemplo lógica de filtrado:

```
transform(array: any[], filtrador:string,
columna:string): any {

    if(filtrador==='')

        return array;

    //Poner todo a minúsculas para que no importe como se
    ha escrito

    filtrador=filtrador.toLocaleLowerCase();

    //Devolver array ya filtrado

    return array.filter(item=>{

        return
        item[columna].toLowerCase().includes(filtrador);

    });

} //end transform
```

Como conectar al html:

```
<ion-list *ngFor="let noticia of noticias | filtro:
textoBuscar: 'titulo'">
```



Se le añade al array en el html el símbolo de tubería | , el nombre del módulo, que en este caso es filtro y a partir de qué atributo va a filtrar el array.

6. Conclusiones

➤ Principales problemas.

Los principales problemas que han surgido durante la construcción de la aplicación han ido en torno al calendario.

El calendario en un principio aunque no daba ningún error, no aparecía en la pantalla pero el problema era que le faltaba la etiqueta encima del constructor

`@ViewChild(CalendarComponent) myCal: CalendarComponent;` .

El siguiente error en aparecer respecto al calendario era que no era capaz de añadir un evento nuevo para que apareciese en el calendario y haciendo pruebas y mirando un ejemplo con un calendario ya hecho, entendí que los `ion-datetime` devuelven y reciben Strings con un formato distinto al `Date` que necesita el calendario para poder añadir el evento, entonces cree un método junto con un parseador dentro del mismo para poder parsear correctamente el string al tipo `Date` concreto que necesita el calendario.

El siguiente y último error respecto al calendario fue que al usar una clase personalizada `Evento.ts` al subirla a firebase, firebase no reconocía el `Date` y había veces que no se subía el objeto correctamente o simplemente al volverlo a recoger, como no se ha subido correctamente, no se podía formatear para que el calendario lo reconociese.

La solución más propia que vi y realicé en el proyecto fue usar mi clase personalizada pero a la hora de la subida y recogida de datos de firebase respectivos al calendario, crear un objeto anónimo antes de subir el evento y



subir esa misma, de esta forma si reconocía el tipo Date y firebase lo guarda como timestamp.

Otro error encontrado durante la aplicación fue que al cerrar sesión del usuario, la aplicación seguía reconociendo el rol del antiguo usuario, su solución fue en el **app.component.ts** hacer el comando propio de angular **window.location.reload()** que se encarga de actualizar la vista de la aplicación.

El último error que surgió fue que yo pensaba que angular, al usar etiquetas `htmls`, reconocería las etiquetas para los enlaces pero al usarlas, aparecía un error diciendo que no tenía permiso para acceder a los enlaces que intentaba acceder. Buscando información resultó que hay un método propio de angular para poder abrir enlaces, que puede ser usado tanto como por inyección en el `html` como el método en sí dentro del código de cada página.

Es el siguiente método:

```
window.open('https://www.instagram.com/banda_villaverde/', '_system', 'Location=yes');
```

➤ **Futuras mejoras a implementar.**

Varias de las posibles mejoras a implementar serían las siguientes:

- Sistema de recuperación de contraseñas.
- Login automático.
- Borrado de autenticación del usuario eliminado automático.
- Sistemas de paginación de noticias que funcione a tiempo real.
- Añadir nuevas funcionalidades si la banda de música las requiere como por ejemplo un chat, subir la aplicación a la play store, chat privados para contratos, etc.



➤ **Aprendizaje.**

Gracias al hacer este proyecto he adquirido conocimientos de cómo poder filtrar con los pipes de angular y poder facilitar al usuario final la búsqueda y no tener que recorrer toda la lista hasta encontrarla.

También he aprendido a gestionar distintos roles de usuario usando el usuario con el que se ha logueado como variable global con un servicio de angular para que todas las clases puedan acceder a ella.

Evidentemente también he aprendido a utilizar el módulo del calendario probando todos o casi todos de sus métodos que contiene aunque no estén incluidos en el proyecto.

7. **Glosario de términos**

- BSO → Banda Sonora Original.
- Pipe → Tubería.



8. Bibliografía

➤ Documentación principal:

- Documentación:
 - [Ionic](#).

➤ Calendario:

- Módulo y documentación:
 - [GitHub](#).
- Tutoriales:
 - [Video 1](#).
 - [Video 2](#).
 - [Video 3](#).
- Ejemplo:
 - [GitHub](#).

➤ SearchBar:

- Documentación:
 - [Ionic](#).
- Tutoriales:
 - [Video 1](#).
 - [Video 2](#).

➤ Conversión a Android:

- Documentación:
 - [Ionic](#).