TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Creación de una página web y aplicación Android para la Cátedra del Agua de Almería

Curso 2020/2021

Alumno/a:

Ramón Francisco Ramos Tristán

Director/es:

José Fernando Bienvenido Bárcena

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

Con el motivo de la creación de una nueva Cátedra del Agua en Agricultura, Regadíos y Agroalimentación se ha hecho manifiesta la necesidad de la creación de un sitio web el cual permita tener localizada a la cátedra y aumentar de esta manera su presencia online.

Este trabajo se basará en la creación de la página de la nueva cátedra que permitirá mostrar las asociaciones con las cuales se trabaja y un texto de presentación el cual explicar la razón del nacimiento de la cátedra y su objetivos así como una serie de ventanas comunes a la mayoría de las cátedras del agua como pueden ser una sección de noticias en las cuales se mostrarán notas de prensa, las actividades organizadas por la cátedra, publicaciones de interés que estarán disponibles para la descarga por parte de los usuarios y una sección de contacta en la cual se expondrá la información relacionada de la catedra como pueden ser el email o un teléfono de contacto, además de la ubicación de la misma y un formulario que permitirá escribir mensajes para comunicarse de forma directa.

Dado que gran parte del contenido de la página será dinámica, es decir, tanto la sección de noticias, como la de publicaciones y actividades se irán modificando añadiendo más elementos es necesaria la creación de un panel de administración que permita gestionar el contenido con el cual trabaja la página. Debido a que las personas que tendrán que encargase de esta tarea no tienen una gran experiencia con la gestión de este tipo de tecnologías lo que se hará es crear una aplicación Android que permita administrar la página de una forma sencilla e intuitiva desde el móvil.

Trabajo de Fin de Grado

Creación de una página web y aplicación Android para la Cátedra de Agua de Almería

Alumno/a: Ramón Francisco Ramos Tristán

Director: José Fernando Bienvenido Barcena

Grado en Ingeniería Informática

Escuela Superior de Ingeniería

Universidad de Almería

Curso 2020/2021

Índice de contenidos

Índice de Figuras

1 Introducción

En este apartado se hará una breve explicación acerca de cual es la motivación principal por la cual se decidió y finalmente se llevó a cabo la elaboración de este proyecto, así como los objetivos que se pretendian alcanzar y se han logrado mediante la realización del mismo. En este apartado también se hablará acerca de la planificación temporal que se había planeado para las diferentes fases de las que consta el trabajo y su ejecución, así como de la estructura y las secciones de las que se encuentra constituido este documento.

1.1 Motivación

Las cátedras son un medio de cooperación en las cuales se promocionan las relaciones entre empresas, asociaciones y las unidades de investigación y conocimiento de las universidades, que al mismo tiempo contribuyen a la formación de futuros profesionales en las áreas de trabajo e investigación de interés de la cátedra, que además ayudan a reforzar la relación entre la universidad y la sociedad a través del sector productivo.

En este contexto se pueden ubicar las cátedras del agua que se centran normalmente en la realización de actividades formativas, de investigación, desarrollo y la promoción de acciones de difusión y divulgación vinculadas a la realidad, problemática y perspectivas del agua.

Un ejemplo de estas es la Cátedra Aqualia del Ciclo Integral del Agua que se formó mediante la unión de esfuerzos por parte de la Universidad de Almería, en especial del centro de investigación en energía solar (CIESOL) y Aqualia a principios de año, que se centra en la integración de la energía solar en los procesos del ciclo integral del agua a través de la depuración con microalgas así como su gestión inteligente, regeneración del agua con energía solar, aprovechamiento del agua residuales …

Sin embargo siendo la única existente en Almería de su tipo, su enfoque centrado exclusivamente en el agua y uso de la energía solar, hace que se manifieste la necesidad de la creación de una cátedra que exponga las carácteristicas propias del sector agrícola local como pueden ser la explotación familiar, la existencia de cooperativas, alhóndigas y empresas agroalimentarias, el uso y gestión del agua de regadio que conforman el denominado “modelo Almería”.

Con motivo de todo lo anterior expuesto se creó la Cátedra del Agua en Agricultura, Regadíos y Agroalimentación, a través de la unión de la Junta de Usuarios del Acuífero del Poniente Almeriense, Federación de Regantes de Almería y Aguas del Almanzora S.A. que tendrá como sede la Universidad de Almería y que tiene como objetivos el intercambio de conocimientos e información en el ámbito propio de las actividades desarrolladas, creación de programas de actividades conjuntas para la formación, investigación y divulgación, organizar foros de encuentro entre académicos…

Debido a su reciente creación, la cátedra no disponía entonces de una página web que ayudara a aumentar su visibilidad y en la cual mostrar y difundir los resultados de las actividades desarrolladas por la misma. En esta página aparte de las actividades organizadas, también se realizaría una presentación de la cátedra y otras secciones como noticias o publicaciones relacionadas.

Como gran parte del contenido de la página es dinámico también se hizo necesario la creación de un panel de administración que permitiera gestionar los contenidos, dado que se pretende facilitar esta tarea e introducirla a personal que no disponga de gran formación tecnológica, se optó por crear una aplicación móvil con la cual se pueda modificar los datos que se muestran en la página.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado ha sido la creación y puesta en marcha de una página web para la Cátedra del Agua en Agricultura, Regadíos y Agroalimentación asi como tambien la de una aplicación Android que ha permitido administrar el contenido de la página.

La página web debia de hacer uso de las tecnologías para las que provee soporte la Universidad de Almería e introducir los cambios sugeridos por el director de la cátedra, José Antonio Salinas Andújar que es la persona con quien se ha trabajado de forma periodica a traves de reuniones para cumplir con sus expectativas. Esta página tiene que seguir una estructura similar a la de otras cátedras relacionadas con el agua como pueden ser las de Alicante y Murcia, es decir, una página de inicio en la cual se muestren las organizaciones con las que se trabaje, una sección de información acerca de la justificación/razón de ser e integrantes de la cátedra, noticias, actividades, publicaciones y contacta, aparte de otras que el director vea adecuado añadir en el futuro.

La aplicación Android debía de poder ser capaz de mostrar las noticias, actividades y publicaciones actuales además de poder modificarlas o añadir otras. En ella se visualizan los mensajes que se han enviado a traves de la sección contacta y se puden responder enviando un email directamente al remitente.

Para cumplir con estos objetivos principales, se abría un abanico de objetivos secundarios que tambien se debian de alcanzar:

- Uso de las tecnologías provistas por la universidad para no recurrir a entidades externas y tener que hacer uso extra del presupuesto.
- Estudio de la tecnologia empleada para crear la mejor experiencia posible para los usuarios, en el cual se incluyen tiempos de carga, compatibilidad con distintos navegadores…
- Crear una base de datos bien estructurada que haga el mejor uso posible de los recursos de los que se dispone.
- Crear una página web amigable, esteticamente agradable y “responsive”.
- Publicar la página lo más pronto posible, dado que no se dispone de una forma de darse a conocer e interactuar con posibles interesados, la rapidez del despliegue de la página es vital.
- Adecuarse a las necesidades de la cátedra siguiendo el consejo del director a traves de una serie de reuniones con el mismo que han tenido lugar de manera regular y en las cuales se le informaba acerca del progreso del trabajo y las actualizaciones realizadas desde la reunión previa.

- Mejorar el posicionamiento SEO de la aplicación web para aumentar la visibilidad de la cátedra.
- Creación de una aplicación Android ligera y rápida, que sea compatible con la mayoría de dispositivos y con un consumo de recursos mínimo.
- Crear una comunicación entre el servidor y la aplicación a traves de una REST API que se alojará en el servidor donde se mantiene la página web.

Con la consecución de cada uno de los objetivos secundarios expuestos se debían de conseguir cumplir los principales y realizar de una forma satisfactoria el Trabajo de Fin de Grado.

1.3 Planificación temporal

Para poder lograr la finalización del proyecto se ha llevado a cabo una descomposición del trabajo del mismo en una serie de etapas. La mayor parte de las etapas corresponden con los objetivos secundarios dispuestos en el punto 1.2, la realización de las mismas es escalonada, es decir, unicamente una vez se finalice una etapa y se realice una comprobación de que la misma cumple con el objetivo con el que está ligada se podrá avanzar a la siguiente.

A continuación se mostrarán las etapas de las que está compuesto el trabajo y la duración prevista de cada una de estas se muestra en la figura 1:

- Analisis de las tecnologías disponibles, mediante el cual se realizara un estudio de las mejores opciones de desarrollo para el proyecto.
- Preparación de una serie de bocetos basados en cátedras similares, para presentar al director una base a partir de la cual realizar cambios e incorporar funcionalidades.
- Entrevistas con el director de la catedra que tendrán lugar de manera retroactiva y periodica en la cual obetener impresiones por su parte.
- Modificación de los diseños en base a las impresiones del director.
- Diseño y creación del backend de la aplicación el cual incluye tanto la base de datos como la API REST.
- Diseño de la aplicación Android para definir tanto el aspecto como las funcionalidades de la misma.
- Busqueda de información relacionada con la creación de aplicaciones Android (tecnologías y formas de desarrollo) .
- Implementación de la aplicación.
- Comprobaciones del resultado actual, tanto a nivel de diseño como implementación.
- Publicación de la página web.
- Redacción de la memoria TFG.

- Preparación de la exposición.

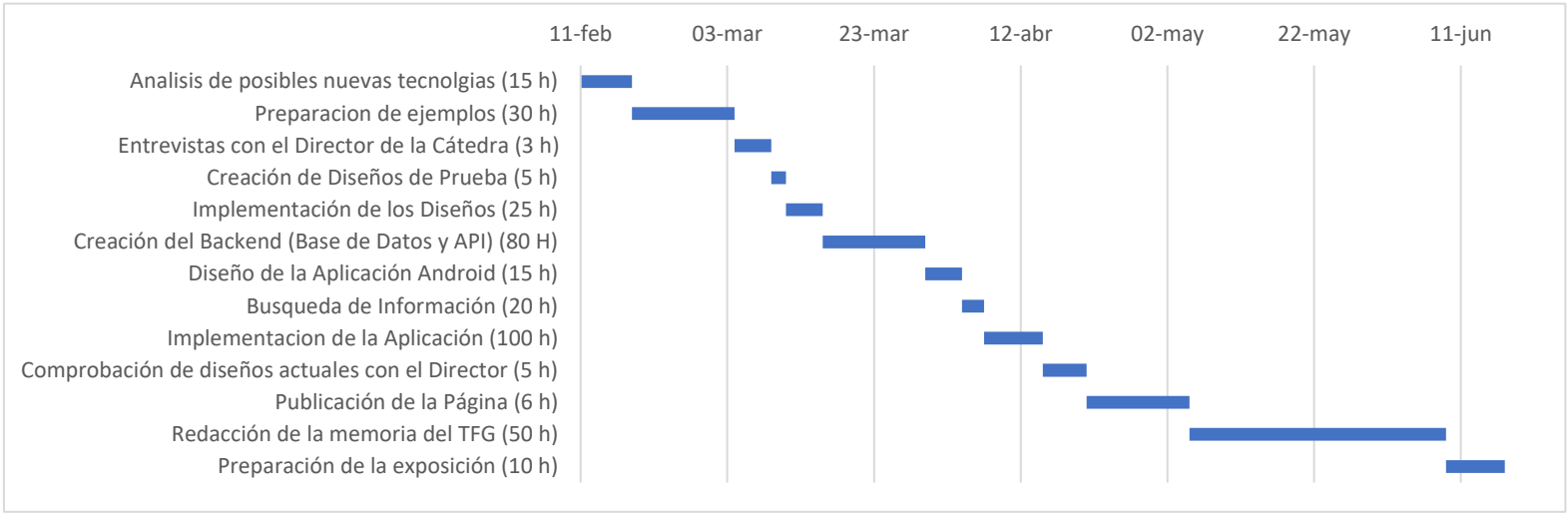

Fig. 1 Cronograma previsto del Trabajo de Fin de Grado

Mientras que en la figurar anterior se ha podido observar el cronograma de las tareas que se creó en un principio, en la figura 2 se puede visualizar el cronograma actualizado en el que se han modificado algunas de las tareas para mostrar cual ha sido la duración real de cada una.

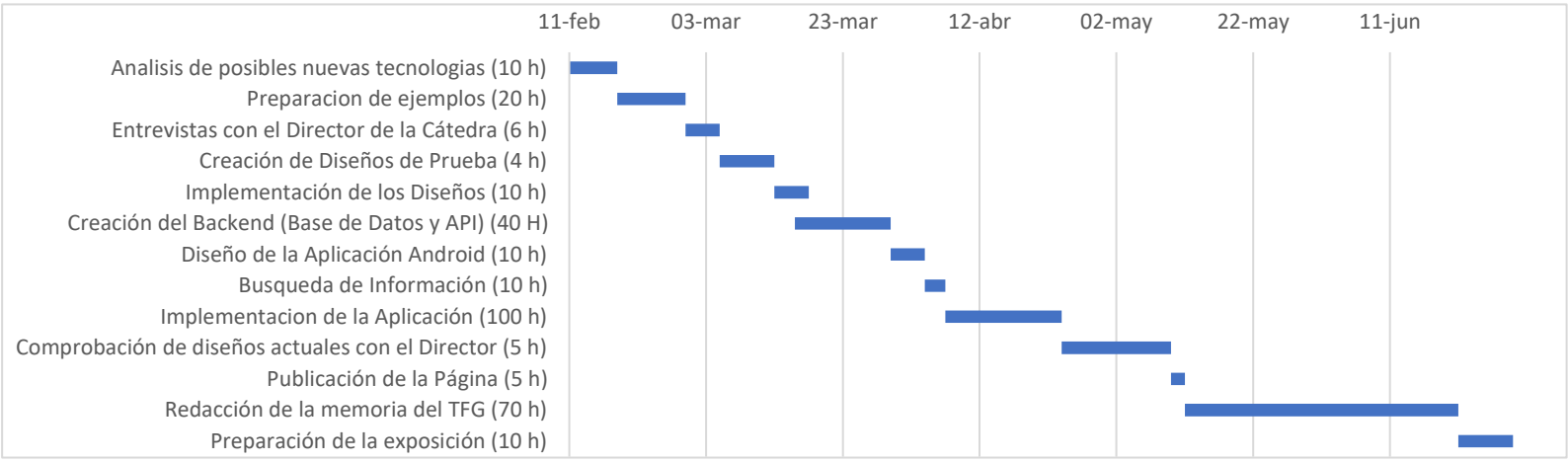

Fig. 2 Cronograma real del Trabajo de Fin de Grado

1.4 Estructura del documento

La memoria del proyecto se encuentra dividida en la siguiente serie de bloques:

Tecnologías, herramientas y servicios

En este apartado se describen tanto los diferentes lenguajes, aplicaciones y tecnicas que se han usado a lo largo del proyecto para su elaboración como de los servicios a los que se ha recurrido.

Desarrollo

En este apartado se describen los diferentes pasos que se han seguido para la creación de tanto la página web como de la aplicación móvil cubriendo desde el diseño inicial hasta la implementacion del código.

Conclusiones

En este apartado se comentan los resultados del trabajo, maneras de mejorarlo y otras posibles tecnologías que se podrían haber usado.

Bibliografía

Se muestra toda la bibliografia utilizada durante el desarrollo del proyecto.

2 Tecnologías, herramientas y servicios

Una vez se han definido cuáles son los objetivos que se pretenden alcanzar con la realización del proyecto, las tareas en las que este se descompone y como se organizan en el tiempo podemos pasar a explicar cuáles son los elementos de los cuales se ha hecho uso para la consecución del trabajo.

En este apartado se hará un desglose de cuáles son las tecnologías que se han utilizado, las herramientas, las aplicaciones de las que se ha hecho uso y los servicios a los cuales se han recurrido.

2.1 Tecnologías utilizadas

2.1.1 HTML

HTML (HyperText Markup Language) [1] es el componente más básico de las páginas webs en el cual se define el significado y la estructura del contenido.

Se trata de un estándar a cargo del W3C (World Wide Web Consortium), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. HTML se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la WWW (World Wide Web).

La estructura se encuentra definida por una serie de marcas que nos permiten etiquetar tanto texto como imágenes y otro tipo de contenido. Una de sus principales ventajas es que puede desarrollarse en HTML mediante el uso de un simple editor de textos.

En este caso se ha hecho uso de HTML5 [2] que es la última versión de HTML disponible y que tiene como principal incorporación XHTML, una nueva variante de sintaxis para HTML siendo la primera vez que HTML y XHTML se han desarrollado en paralelo, aunque cabe mencionar que no haremos uso de esta característica durante el desarrollo del trabajo, ofrece potencial para la futura evolución de la página web.

2.1.2 CSS

CSS (Cascading Style Sheets) [3][4] es el código que se usa para agregar estilo a las páginas webs. No se trata realmente de un lenguaje de programación sino de un lenguaje de hojas de estilo que nos permite aplicar estilos de forma selectiva a los elementos que hemos definido previamente en HTML.

La especificación CSS es mantenida por el World Wide Web Consortium (W3C). El MIME type text/css está registrado para su uso por CSS descrito en el RFC 2318.5.

En este caso se ha hecho uso de CSS3 [5] que como en el caso de HTML5 es la última versión disponible de CSS y la cual nos trae una nueva serie de propiedades que podemos aplicar como pueden ser border-color, background-size, text-overflow, box-sizing …

2.1.3 JavaScript

JavaScript [6] es un lenguaje de programación que nos permite el script de eventos, clases y acciones para el desarrollo de aplicaciones web entre cliente y usuario. Nos permite dotar de un carácter interactivo a las páginas web mostrando actualizaciones de contenido, mapas interactivos, animación de gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo…

Los navegadores interpretan las sentencias de JavaScript incluidas directamente en una página HTML permitiendo la creación de aplicaciones similares a los CGI. El código de los scripts se introduce entre las etiquetas script de HTML.

Otro aspecto importante de este lenguaje es la anotación de objetos que define conocida como JSON, que se trata de un formato de intercambio de datos. Se utiliza para transmitir datos en aplicaciones web. JSON será el formato que utilizaremos más adelante para crear y recibir las respuestas de la API REST.

2.1.4 PHP

PHP (Hypertext Preprocessor) [7] es un lenguaje de código abierto muy popular y que puede ser incrustado en HTML. Se trata de un lenguaje que se ejecuta en el lado del servidor, generando HTML y enviándoselo al cliente que recibe el resultado de ejecutar el script desconociendo cual era el código subyacente que lo generó.

Una de las mejores ventajas [8] que presenta es el gran parecido que posee con los lenguajes más comunes de programación estructurada, como C y Perl, que permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta.

Se trata de unas de las tecnologías más utilizadas en el despliegue de sitios webs [9], aunque en la actualidad otro tipo de tecnologías en auge como pueden ser Node.js, Golang o ASP.NET están haciendo que el número disminuya. Sin embargo, cabe mencionar que un gran número de páginas relevantes como Zoom, Wikipedia o Facebook hacen uso de PHP.

Ha sido la tecnología elegida para el desarrollo del proyecto debido a que se trata de una alternativa de código abierto que cuenta con un gran apoyo por parte de la comunidad recibiendo actualizaciones de forma constante, la facilidad y rapidez con la cual se puede desarrollar además de la estabilidad, flexibilidad y velocidad que proporciona.

2.1.5 Java

Se trata de un lenguaje de programación originado por Sun Microsystems en 1995 cuya sintaxis deriva en gran parte de C y C++ que tiene como principal característica que se trata de un lenguaje orientado a objetos.

En este proyecto se hará uso de él para el desarrollo de la aplicación móvil dado que ya se ha trabajado con él, es más seguro y fácil de entender, permite la reutilización de código, cuenta con una amplia oferta de librerías aparte de permitir obtener una alta eficiencia.

2.1.6 BootStrap

Se trata de una biblioteca multiplataforma [10] de código abierto para el diseño de sitios webs y aplicaciones webs que contiene plantillas de diseño de tipografía, formularios, cuadros y cualquier otro tipo de elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales que solo se ocupa del desarrollo front-end.

Una de las principales ventajas de Bootstrap es su facilidad de uso, para acceder a sus funcionalidades simplemente tenemos que importarla como hoja de estilos en el HTML que deseemos y ya podemos hacer uso de su basta colección de clases.

2.1.7 MySQL

Se trata del sistema gestor de base de datos relacionales más popular de Internet, cuyas principales ventajas son su potencia y versatilidad con la cual es capaz de satisfacer las necesidades de la mayoría de los proyectos en la web además de su disponibilidad en la mayoría de los servidores.

2.2 Herramientas utilizadas

2.2.1 Visual Studio Code

Se trata de un editor de texto eficaz y ligero desarrollado por Microsoft que tiene como principales ventajas la capacidad de proporcionar emuladores locales para la compilación y ejecución de aplicaciones, capacidad de administrar bases de datos locales o remotas, soporte para casi todos los lenguajes y tipos de aplicaciones, además de contar con un gran apoyo por parte de la comunidad lo que le permite contar con una enorme cantidad de extensiones que proporcionan funcionalidades añadidas.

2.2.2 Adobe XD

Es un editor de gráficos vectoriales [11] desarrollado y publicado por Adobe Inc para diseñar y crear un prototipo de la experiencia del usuario para páginas web y aplicaciones móviles.

2.2.3 Postman

Es una herramienta que se utiliza, sobre todo, para el "testing" de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba la comprobación de este tipo de sistemas, genera una documentación bastante interesante y atractiva, con ejemplos y "snippets" de código, de forma que hace que sea muy fácil de entender cómo funciona una API determinada.

2.2.4 Android Studio

Android Studio es un nuevo entorno de desarrollo integrado para el sistema operativo Android lanzado por Google, diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones y alternativa al entorno Eclipse, hasta ahora el IDE más utilizado.

Al crear un nuevo proyecto en Android Studio, la estructura del proyecto aparece con casi todos los archivos dentro del directorio SRC, un cambio a un sistema de generación basado Gradle que proporcionará una mayor flexibilidad para el proceso de construcción. Además, gracias a su sistema de emulación integrado, Android Studio permite ver los cambios que realizamos en nuestra aplicación en tiempo real, pudiendo además comprobar cómo se visualiza en diferentes dispositivos Android con distintas configuraciones y resoluciones de forma simultánea.

2.2.5 Filezilla

Es una aplicación FTP libre y de código abierto que consta de un cliente y un servidor multiplataforma rápido y confiable con una gran cantidad de funcionalidades y una interfaz gráfica que soporta los protocolos FTP, SFTP y FTP sobre SSL/TLS.

2.3 Servicios

2.3.1 STIC

Se trata del Servicio de las Tecnologías de la Información y las Comunicaciones de la UAL que tiene como misión la innovación y organización eficiente de los sistemas de información y comunicaciones para el apoyo de las tareas de docencia, investigación y gestión de la Universidad.

Será un eje fundamental del proyecto puesto que será el encargado de proveernos con un servidor FTP en el cual subir los archivos del proyecto además de proporcionar un dominio institucional para la página web.

3 Desarrollo del Proyecto

En este apartado se hará una explicación detallada de cada uno de los pasos que se han seguido desde el diseño hasta la implementación de la página y la aplicación. Se encontrará subdividido en los puntos 3.1 y 3.2., siendo el primero dedicado en su plenitud al desarrollo de la página web y el último a la aplicación Android, aparte del 3.3 y 3.4 que explicarán la creación de la newsletter y la publicación de la página respectivamente.

3.1 Página web

En este subapartado se cubre como ha ido tomando forma la implementación de la página web desde la elección de las tecnologías que se usan, diseño, ejecución, despliegue y el mantenimiento de la misma.

3.1.1 Bocetos de diseño

Desde un primer momento se realizó una serie de “mockups” mediante el uso de la herramienta Adobe XD en los cuales quedaría determinada cual sería la estructura preliminar de la página para así tener una idea clara y definida de lo que se pretendía obtener durante la fase de ejecución.

Cabe mencionar que en este apartado se describen los bocetos sobre los cuales se realizó una implementación para tener una base con la cual trabajar con el director de la cátedra, una vez se alcanzó un prototipo funcional realizado con HTML y CSS, se trabajó sobre el mismo para realizar las modificaciones que él consideró oportunas sobre el prototipo.

Para hacerse una idea de cual era un buen diseño se consultaron las páginas de las cátedras de agua de Murcia y de Alicante debido a la proximidad de ambas, se podía deducir que, aunque el enfoque de esta cátedra está más orientado al uso del agua aplicado al regadío se compartiría una estructura similar.

Como se puede apreciar en la figura 3, la página de inicio de la Cátedra del Agua y Sostenibilidad de la Universidad de Murcia cuenta con un texto de presentación, el logo de esta, una sección de información en la cual se muestra la localización de la misma, el teléfono y email de contacto además de las redes sociales en las cuales tienen presentación, en este caso Twitter, Instagram y Facebook.

También se puede observar que cuentan con una sección de destacados en la cual tienen elementos representativos de la sección de premios, actividades y concursos. Las distintas secciones de las que está compuesta la cátedra se disponen en un menú de navegación vertical a la izquierda. Dichas secciones son presentación, integrantes, objeto y fin, actividades, convocatorias, concursos, premios, publicaciones y artículos de prensa.

Fig. 3 Página web de la Cátedra del Agua y Sostenibilidad

Por otro lado, en la figura 4 se puede observar la página de introducción de la Cátedra del Agua de la Universidad de Alicante, en la cual podemos ver a primera vista el logo de la misma, la información de contacto, una barra de navegación horizontal, una imagen y un enlace a la presentación de la cátedra.

Si se sigue y hace scroll hacia abajo como se muestra en la figura 5 se observa el texto de presentación de la cátedra y una selección de objetivos. Las secciones en las que se divide son el inicio, información, eventos, noticias, publicaciones y contacto.

Fig. 4 Página web de la Cátedra del Agua de la Universidad de Alicante

Fig. 5 Página web de la Cátedra del Agua de la Universidad de Alicante (2)

Se pueden observar patrones similares en la Cátedra AQUAE de Economía del Agua representada en la figura 6 y la Catedra del Agua EMASESA en la figura 7.

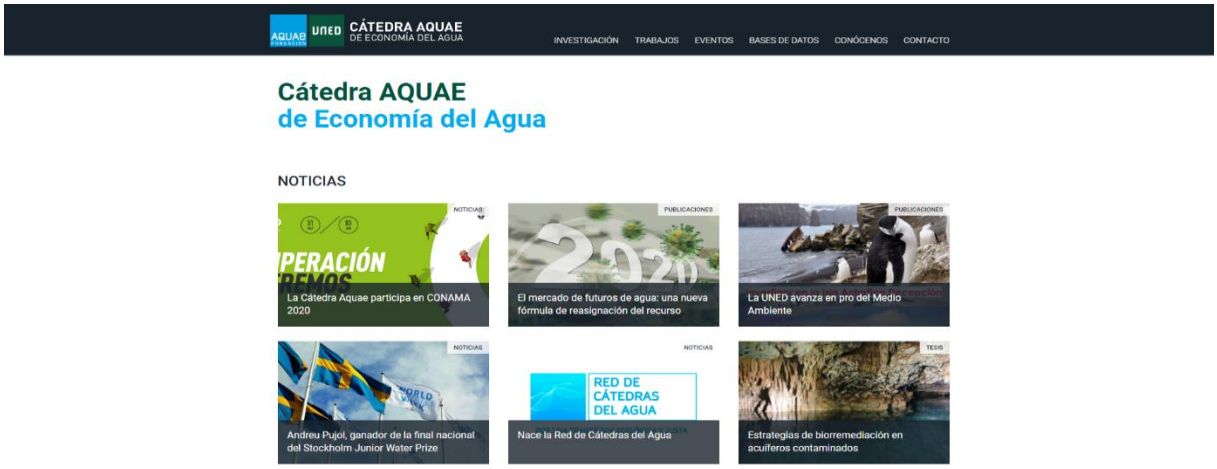

Fig. 6 Página web de la Cátedra AQUAE de Economía del Agua

Fig. 7 Página web de la Cátedra del Agua EMASESA

Con todo esto podemos hacernos una idea de cuáles son las ventanas básicas de cualquier página que intente presentar una cátedra del agua, estas son una página de inicio que puede contar o no con una serie de fotos que la presenten, una página de información en la que se comenten objetivos e integrantes, noticias relacionadas con la cátedra, actividades organizadas por la misma, publicaciones y una sección “contacta”.

Una vez se tiene una idea acerca de cuál es la idea básica que tenemos acerca de las características de la página se pudo empezar a desarrollar los diseños en los que se van a presentar las diferentes vistas de la página.

Antes de pasar a la creación de los diseños, se hará una breve explicación de cómo se realizan los diseños con Adobe XD, aunque esta herramienta presenta una serie de funcionalidades extra que la hace ideales para la creación de prototipos como puede ser el hecho de añadir navegación entre las distintas ventanas, se basa en la capacidad de añadir componentes como pueden ser cuadrados los cuales se pueden ajustar de la forma que desee el usuario y determinar detalles de su estilo como pueden ser el color u opacidad, también nos permite añadir otros componentes como imágenes y texto.

Dicho esto, se puede pasar al diseño de la página, esta cuenta con una barra de navegación horizontal situada en la parte posterior desde donde se podrá acceder a todas las secciones que son la de inicio, presentación, noticias, actividades y contacto, dado que son las que más se repiten por lo general.

También contara con un título a la izquierda de la barra que servirá como botón para poder acceder a la página de inicio desde cualquier otra ventana, además del logo de la UAL y de las redes sociales en las que se espera que esté presente la cátedra y que servirán como enlace para acceder a ellas. Cabe mencionar que dicha barra estará presente en todas las diferentes ventanas de la página web.

La página de inicio que se puede apreciar en la figura 8 contará con una serie de imágenes que ocuparán el resto de la vista que se irán sucediendo como en una presentación y en el centro el nombre de la cátedra.

Fig. 8 Diseño de la página de inicio

La página de presentación que se puede apreciar en la figura 9 estará formada en un principio por dos textos diferentes, uno que explique acerca de la cátedra y cuáles son sus funciones, objetivos, integrantes y otro sobre el origen, concepción y justificación de cuáles son las causas de su creación.

Fig. 9 Diseño de la página de presentación

La página de noticias que se puede apreciar en la figura 10 cuenta con una serie de noticias, cada una de ellas contenida en una tarjeta en la cual se podrá apreciar la imagen de la noticia, el título de la misma el cual será “clickable” y permitirá redireccionar a la noticia en cuestión y la fecha, estas tarjetas se organizan en filas de 3, se mostrarán todas las noticias almacenadas lo que hará que la altura de la página crezca y haya que hacer uso del “scroll” para poder visualizar las noticias más antiguas.

Fig. 10 Diseño de la ventana de noticias

La página de actividades que se puede apreciar en la figura 11 contará con la lista de actividades organizadas, cada una de las cuales incluirá un título, descripción y fecha de realización de la actividad, el estilo de disposición será el mismo que el de las noticias, pero en este caso eliminaremos las actividades que ya hayan tenido lugar ocultándolas, aunque se dará la opción de poder recuperarlas si así el usuario lo desea.

La página de “contacta” que se puede apreciar en la figura 12 cuenta con un formulario de contacto en el cual los usuarios podrán interactuar directamente con el director dejando mensajes, en este formulario habrá campos para el nombre del usuario, su email con el cual se podrá contactar con el usuario después y el contenido del mensaje.

También hay otra sección en la cual se mostrará toda la información de la catedra para contactar con ella, que será el número de teléfono, la dirección de email, sus redes sociales y la ubicación en la que se encuentra la cátedra.

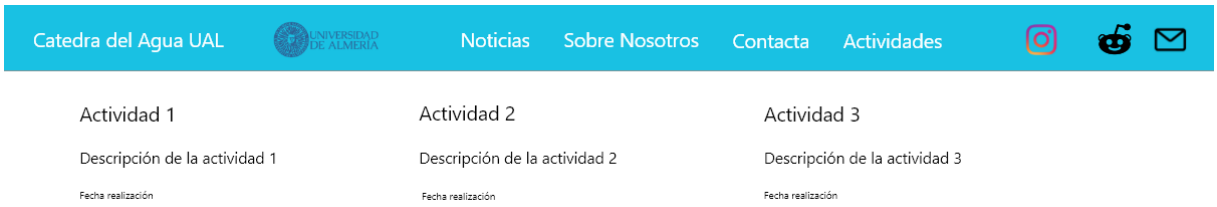

Fig. 11 Diseño de la ventana de actividades

Fig. 12 Diseño de la pagina de contacta

3.1.2 Implementación del diseño

En este apartado se cubrirá como se han llevado a código los bocetos que se han creado en el punto anterior, dicha implementación solo cubrirá el apartado de “front-end”, con lo cual nos centraremos en el uso de HTML y CSS, además de BootStrap.

Para implementar el diseño plasmado en Adobe XD haremos uso de Visual Studio Code que es un editor de texto que ofrece soporte tanto de HTML como de CSS, dado que los archivos que generamos simplemente se pueden abrir con un navegador para comprobar cómo van tomando forma, en este paso no necesitaremos de más herramientas.

Como se ha mencionado antes, en este proceso nos ayudaremos de BootStrap que es un framework el cual para utilizarlo, simplemente tenemos que importarlo como hoja de estilos en cada archivo HTML que necesitemos, nos ayudará a desarrollar de forma más fácil y rápida puesto que incluye una gran serie de clases que al usarlas nos permitirá definir con un término el color del texto, del fondo, el tamaño de la letra, el “padding”, margen o borde de cualquier elemento además de que en su página se muestra una serie de ejemplos para el desarrollo de componentes tales como barras de navegación, botones, alertas…

Mediante el uso de HTML, CSS y BootStrap se consiguió modelar el diseño en las siguientes páginas, por ejemplo, para realizar la barra de navegación de la página nos apoyaremos en uno de los ejemplos que nos provee BootStrap el cual modificaremos añadiendo las secciones que hemos definido en la etapa de diseño. También se decidió eliminar las redes sociales y el logo de la UAL que irán incorporados en un nuevo componente.

Como se puede apreciar en la figura 13 el inicio es bastante parecido al diseño que se planeó en un primer momento, la única diferencia que se puede observar se encuentra en la barra de navegación con las diferencias que se han comentado anteriormente.

Fig. 13 Implementacion de la página de inicio

En la figura 14 se puede apreciar la página de información en la cual se hace una presentación de la fundación y objetivos de la cátedra, en esta ventana se puede empezar a ver modificaciones, en primer lugar, se ha modificado el color de fondo sustituyendo el blanco por un gradiente de azul a purpura, también se ha estilizado un poco ambos párrafos centrándolos y añadiendo un botón para cada párrafo que permite enlazar las páginas de las asociaciones implicadas, también se ha añadido una parte inferior la cual se abordará más adelante.

Fig. 14 Implementación de la página de información

En la figura 15 se puede observar la sección de noticias la cual ha seguido el mismo patrón que el diseño únicamente colocando un título encima a modo de aclaración y en la que se ha hecho uso de los componentes definidos por BootStrap.

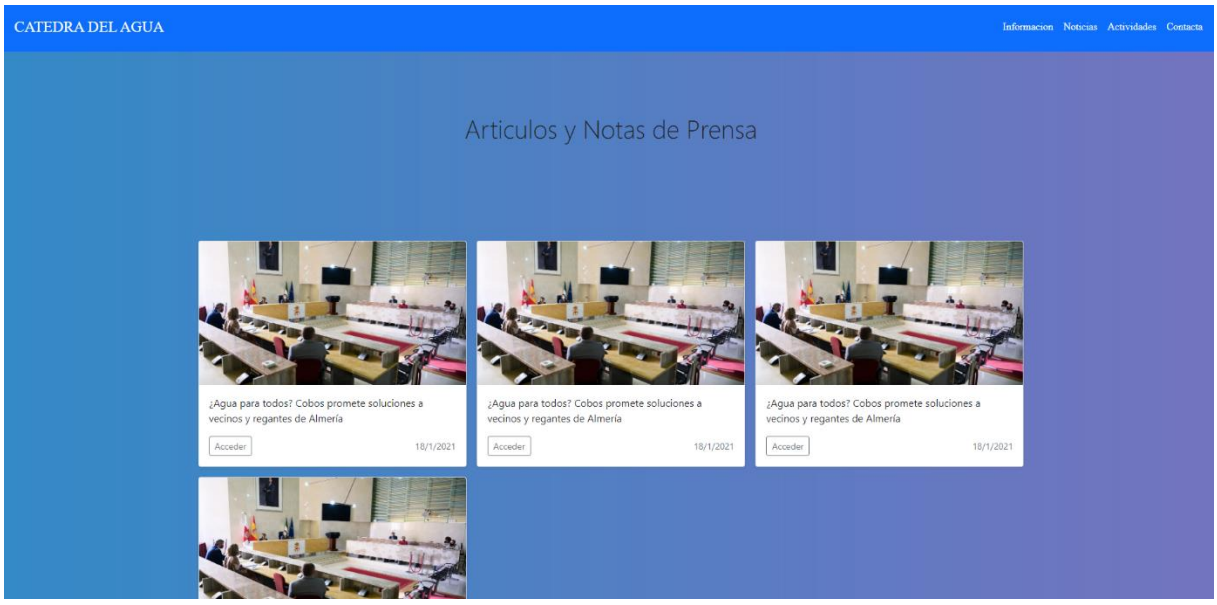

Fig. 15 Implementación de la página de noticias

En la figura 16 se puede ver como se disponen las actividades de la cátedra, la única diferencia es que se han creado 3 columnas que podrán variar de número para organizar los diferentes tipos de actividades que se realicen. También está presente en esta ventana la parte inferior (“footer”) que ahora se puede ver de forma completa en la cual se mostrarán los cronogramas de las diferentes organizaciones que participan en la cátedra, una serie de enlaces de interés como pueden ser el ministerio de agricultura, también se muestran la lista de enlaces a las redes sociales y la información de contacto.

Fig. 16 Implementación de la página de actividades

En la figura 17 se puede ver la sección de contacta donde se ha suprimido la parte de redes sociales dado que ahora se encuentra incorporada en la parte inferior.

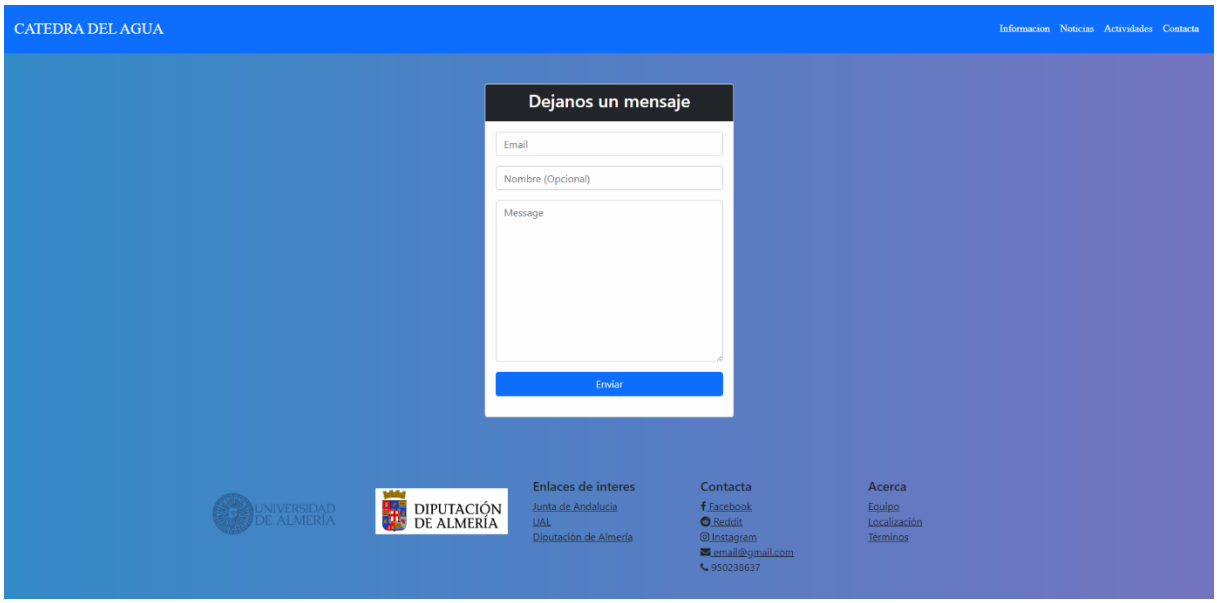

Fig. 17 Implementación de la página de contacta

Como ejemplo de cómo se ha creado el código de la página tenemos la figura 18, en la cual podemos observar una parte del código de cómo se estructura una de las columnas, en la cual mediante el uso de las clases que nos presta BootStrap creamos una columna cuya anchura ocupe un tercio del total, puesto que se trabaja sobre un máximo de 12, con p-3 le asignamos un padding en todos los lados de 3, lo que equivaldría a 1 rem, un tipo de medida relativa de CSS, con text-left se hace que el texto se justifique a la izquierda del elemento. De esta forma se pueden realizar cambios rápidos a los elementos sin tener que crear una clase específica para ello y especificar los detalles de esta en el archivo de CSS.

```
<div class="col-sm-4 p-3 "><h1>Fotografia</h1>
    <div class="evento p-3">
        <a href class="" style="text-decoration:none;"><p class="text-left text-dark mb-0">Webinar ODS6 Agua Limpia y Saneamiento.</p></a>
        <p class="text-left text-dark">Descripcion de la actividad <br> 14 de Febrero de 2015</p>
    </div>
    <div class="evento p-3">
        <a href class="" style="text-decoration:none;"><p class="text-left text-dark mb-0">Webinar ODS6 Agua Limpia y Saneamiento.</p></a>
        <p class="text-left text-dark">Descripcion de la actividad <br> 14 de Febrero de 2015</p>
    </div>
</div>
```

Fig. 18 Parte del código HTML de la página de actividades

Con todo lo realizado ya tenemos completo un modelo base de la página web a partir del cual se puede trabajar con el director para realizar las modificaciones necesarias.

3.1.3 Diseño e implementación de la base de datos

En este apartado se abordará como se encuentra estructurada la base de datos con la cual se ha trabajado durante el desarrollo de lo que resta de proyecto, aunque puede darse el caso de que se produzcan modificaciones en ella si el director desea añadir un nuevo apartado o una nueva funcionalidad.

Una vez se ha mostrado el diseño preliminar de la página al director se han llegado a una serie de conclusiones tanto sobre el diseño como sobre la implementación, las diferentes modificaciones que se aplicarán a la interfaz y la funcionalidad de la página se abordarán más adelante en el punto 3.1.8.

Con todo lo que se ha concluido con estas reuniones se ha alcanzado una idea acerca de que clases de tablas va a mantener la base de datos, que servirá para que el contenido de la web sea dinámico y modificable a través de la aplicación móvil.

Para hacer un diseño de la base de datos preliminar se ha utilizado la herramienta de Visual Paradigm con la cual se puede hacer uso de un diagrama de clases, cada clase representará a una tabla, tendrá el título de la misma y sus atributos, además de que permite plasmar las relaciones que existen entre las tablas en caso de que existieran.

Debido a todo lo anteriormente dicho se llegó a la conclusión de que la base de datos debía guardar todo lo relacionado a las actividades, publicaciones, noticias, los mensajes de los usuarios de la sección contacta y los emails de la “newsletter” que se ha decidido crear.

Como se puede apreciar la mayoría de ideas que se plasmaron se han mantenido como son las noticias, actividades y contacta, a las cuales se le suman una nueva tabla para las publicaciones y la newsletter, ambas fueron recomendaciones como características que le otorgarán un valor añadido a la página y con las cuales el director estuvo de acuerdo, también se planteó crear unas tablas adicionales para los componentes de la cátedra y para el texto de presentación pero conforme avanzó el desarrollo se descartó debido a la naturaleza estática de las mismas.

En la figura 19 se puede apreciar el diagrama de la base de datos en el que se encuentran las tablas que se han mencionado previamente, como podemos apreciar no existe ningún tipo de relación entre las tablas dado que todas son independientes, en el contacta se guardará el mensaje con el nombre, email del autor y el contenido, las actividades podrán relacionarse con un tipo de actividad, y tendrán título, fecha, un enlace al cual redireccionar y una descripción, las publicaciones contarán con título, descripción, fecha y enlace, las noticias con un título, fecha, enlace de la imagen, enlace de la noticia y finalmente la newsletter solo almacenará los emails de las personas que se suscriban a ella.

Cabe mencionar que estos solo son los elementos específicos de cada tabla, cada una tendrá el atributo “id” que será un número de tipo entero el cual nos permita identificar cada registro y realizar operaciones de actualización y borrado sobre los datos existentes.

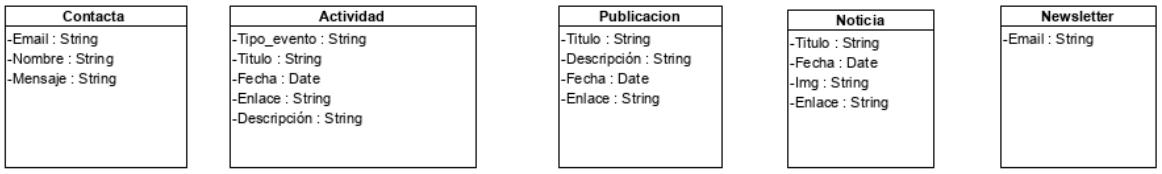

Fig. 19 Diagrama de clases de la base de datos

Una vez tenemos el diseño de la base de datos listo solo resta la creación de la misma, pero para ello tenemos que tener antes un servidor de MySQL funcionando que nos permita almacenar los datos, para ello haremos uso de XAMPP, el cual ofrece un entorno local que permite tener PHP, PERL y MariaDB, una base de datos open source derivada de MySQL. Se puede ver la interfaz de dicho programa en la figura 20 donde se observan los diferentes módulos de los que consta actualmente la aplicación, para hacer uso de ella simplemente nos tenemos que ir a su página descargar el ejecutable y ejecutarlo.

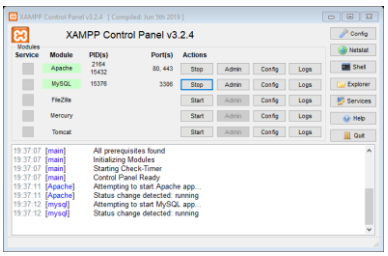

Fig. 20 Interfaz del programa XAMPP

Una vez tenemos el servicio de MySQL funcionando nos podemos centrar en la creación de la base de datos, para ello nos podemos dirigir al phpMyAdmin en nuestro localhost que se encontrará en la dirección http://localhost/phpmyadmin/index.php y que podemos observar en la figura 21.

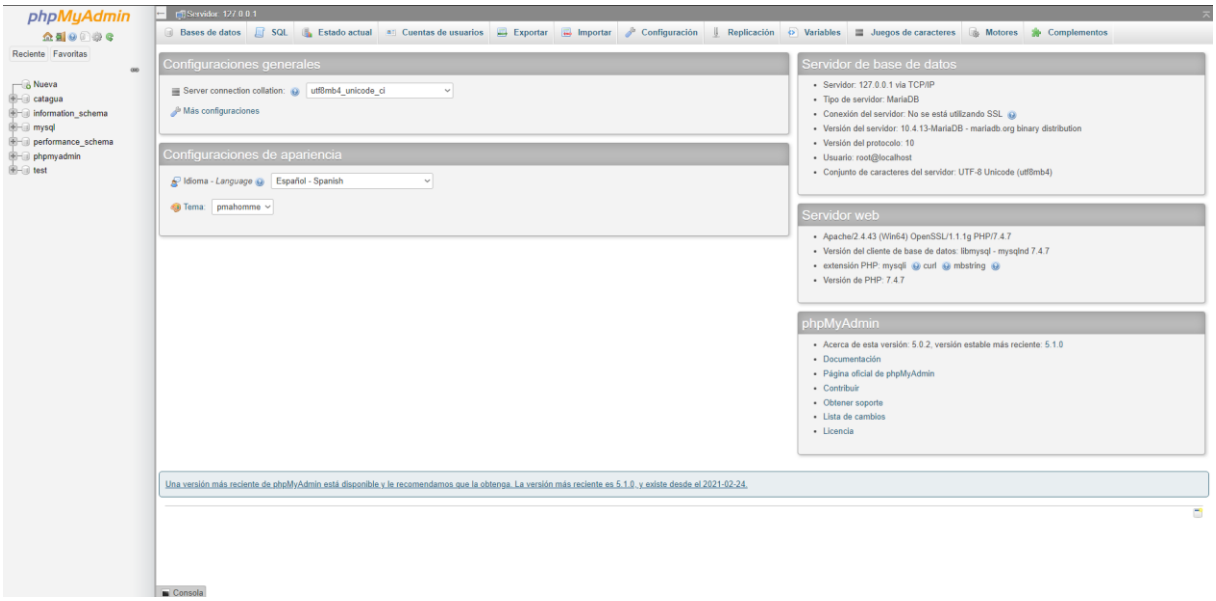

Fig. 21 Interfaz de phpMyAdmin

En dicho programa se puede crear una nueva base de datos y especificar dentro de ella cada una de las tablas que se han diseñado, para ello simplemente tendremos que darle a crear una nueva tabla y especificar los atributos de la misma con lo cual tendremos una tabla como la de la figura 22 en la cual se define el id como la clave principal de la tabla.

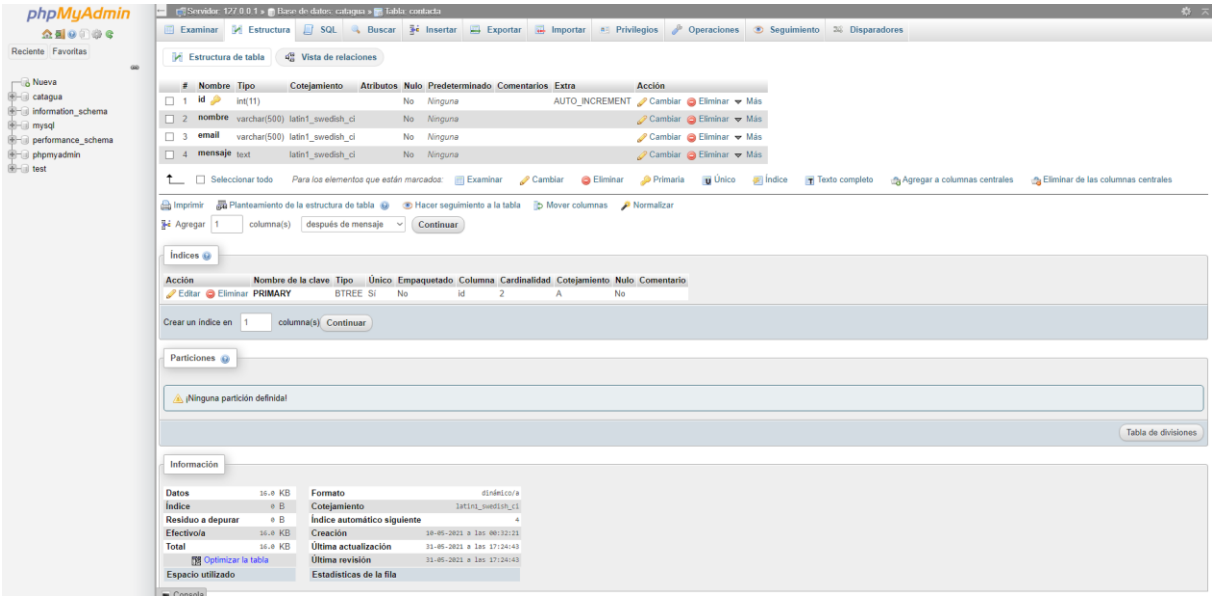

Fig. 22 Vista de la tabla de contacta en phpMyAdmin

3.1.4 Implementación del código PHP

En este apartado nos dedicaremos a transformar la página que anteriormente se ha creado en HTML y CSS y es estática en otra que haga uso de la base de datos para lo cual se utilizará PHP.

PHP es un lenguaje que permite incrustar código HTML en sus archivos y añadirles funcionalidades, para desarrollar en PHP haremos uso de Visual Studio Code y de XAMPP que nos provee de un directorio en el cual se pueden ejecutar archivos PHP, y que por defecto se encuentra dentro de la carpeta de xampp y se denomina “htdocs”.

Lo primero que se hará será estructurar el código, dado que PHP nos da la posibilidad de incluir código de otros archivos dividiremos el código de la página en diferentes componentes, uno para la barra de navegación, otro que se encargue de importar las diferentes hojas de estilo que se utilizan, otra para la parte inferior de la página (footer) en la cual se muestran las organizaciones involucradas y otro para conectarse a la base de datos.

Estos componentes se organizan en una carpeta denominada “partials”, se puede ver el contenido de dicho directorio en la figura 23, en dicha figura se aprecia los componentes de los que se han hablado anteriormente aparte de dos archivos más, addNewsletter y addEscribenos que se encargaran de gestionar los formularios de la suscripción de una persona a la newsletter y de los mensajes escritos por los usuarios.

También se puede observar cómo se estructurará el resto de los archivos, en la carpeta public se almacenarán tanto los archivos que se quieran mantener en el servidor para que sean accesibles, las imágenes que se utilizan en la página y los archivos CSS en los cuales se han creado los estilos personalizados. También existe una carpeta para toda la lógica detrás de la API que abordaremos más adelante y los archivos que componen las diferentes ventanas de la página y que se encuentran en la raíz del directorio.

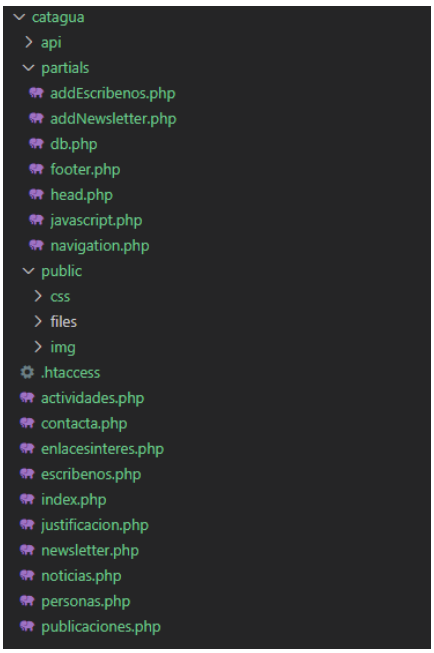

Fig. 23 Contenido del directorio partials

En la figura 24 se puede ver un ejemplo de cómo se incorpora el código de los diferentes componentes mediante el uso del comando “include”, en el cual se especifica la ruta del archivo que queremos incorporar.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Noticias</title>
    <?php include 'partials/head.php'; ?>
    <link href="/css/noticias.css" rel="stylesheet">
</head>

<body>
    <?php include 'partials/navigation.php'; ?>
      <main>
        <section class="py-5 text-center container">
          <div class="row py-lg-5">
            <div class="col-lg-6 col-md-8 mx-auto">
              <h1 class="fw-light ">Articulos y Notas de Prensa</h1>
            </div>
          </div>
        </section>
```

Fig. 24 Parte del código PHP del archivo de la ventana de noticias

De esta forma en caso de que el director quiera realizar un cambio en específico en algunos de los componentes como incorporar una nueva sección en la barra de navegación, solo se tiene que modificar el código de un archivo y las modificaciones se aplicarán a todos los que lo tengan importado facilitando y agilizando de esta manera la implementación de cambios.

Ahora que se sabe cómo estructurar el trabajo podemos adentrarnos dentro de la forma con la cual se trabaja con PHP.

En los archivos que representen ventanas como pueden ser las de inicio cuyo contenido es estático y no se presenta ningún tipo de interacción con la base de datos lo que haremos será redefinir la extensión del archivo de .html a .php dado que PHP nos permite tener incrustado código HTML sin ningún tipo de problema.

En los archivos que tengan que recurrir a elementos dinámicos, es decir, a la base datos en primer lugar se tendrá que importar otro archivo de partials denominado db.php en el cual se establecerá la conexión a la base de datos, en dicho archivo que se puede apreciar en la figura 25 se puede observar cómo se conecta a la base de datos a través del nombre de la base de datos que se desea usar, el usuario, la contraseña y el nombre del servidor, cabe mencionar que dado que se está trabajando en local estos datos tendrán que modificarse una vez se publique la página.

La conexión se establecerá a través de mysqli_connect [12], que es la función que se suele utilizar con las bases de datos MySQL cuya versión es superior a la 4.1.3, mientras que con las anteriores se utiliza mysql_connect.

Este archivo cuenta con dos funciones principales, la primera denominada “NonQuery” que se encargará de ejecutar las sentencias que no impliquen realizar una consulta que devuelva datos, es decir, se encargará de las operaciones de modificación, creación y eliminación.

El método recibirá por parámetro la sentencia, la ejecutará y en caso de que se haya ejecutado de forma satisfactoria devolverá el mensaje correspondiente, en caso contario informará del error ocurrido. Esta función también cuenta con un segundo parámetro que es la conexión, lo cual nos permitiría hacer consultas con otras bases de datos, pero dado que no trabajaremos con ello de momento lo que haremos será si dicho parámetro es nulo utilizará la conexión que se ha establecido previamente.

La segunda función denominada “Query” se encargará de las consultas que soliciten datos, contará con los mismos parámetros que la anterior función que son la conexión y la consulta en sí y devolverá un array con los resultados obtenidos.

También hay que mencionar que la conexión que se establece con la base de datos debe de poder proveer respuestas cuyo contenido se encuentre en UTF-8 dado que se debe poder hacer uso de caracteres especiales como la ñ o el acento, por ello estableceremos que el charset de la conexión sea UTF-8.

Este archivo será el pilar principal en el cual se apoyarán los demás archivos que tengan que hacer uso de alguna manera u otra de la base de datos.

```php
<?php
$servername = "localhost";
$database = "catagua";
$username = "root";
$password = "";

$connected = true;

// Create connection
$conn = mysqli_connect($servername, $username, $password, $database);
$conn->set_charset("utf8");
// Check connection
if ($conn->connect_error) {
    $connected = false;
}

function NonQuery($sqlstr, &$conn = null){
    if(!$conn){
        global $conn;
    }
    if ($conn->query($sqlstr) === TRUE) {
        return "New record created successfully";
      } else {
        return "Error: " . $sqlstr . "<br>" . $conn->error;
      }

}

function Query($sqlstr, &$conn = null){
    if(!$conn){
        global $conn;
    }
    $result = $conn->query($sqlstr);
    $array = array();
    if (is_array($result) || is_object($result)){
        foreach($result as $registros){
            $array[] = $registros;
        }
    }
    return $array;
}
```

Fig. 25 Código del archivo db.php

Una vez se sabe la manera en la cual se puede acceder a la base de datos de la aplicación podemos comenzar con la extracción de datos de la misma y la forma en la que se van a mostrar estos por pantalla.

Las consultas que se deben ejecutar deben estar escritas en SQL, que se trata de un lenguaje de alto nivel a partir del cual interaccionaremos con la base de datos, las operaciones más comunes relacionadas con una base de datos como se puede observar en la figura 26 son las CRUD, es decir, las tareas de crear, leer, actualizar y eliminar datos.

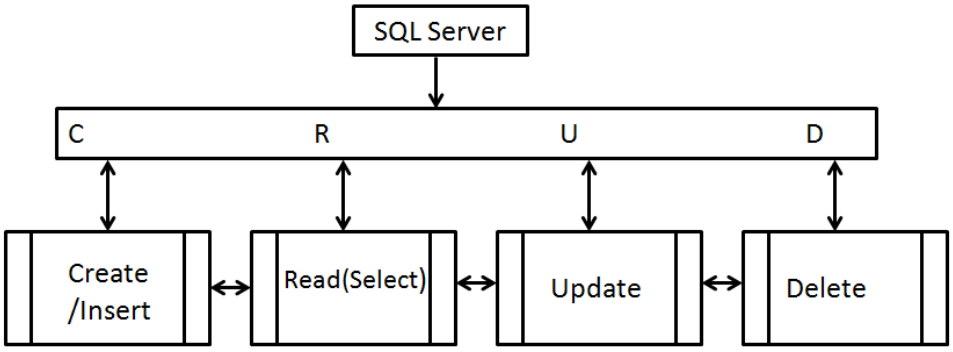

Fig. 26 Esquema de las operaciones CRUD

Las operaciones de lectura en SQL se realizan con el operando SELECT que se encarga de seleccionar las columnas de las que se van a extraer los datos, que se pueden identificar con los nombres de las mismas o en caso de que se quieran seleccionar todas las de la tabla se utiliza el carácter *, a continuación utilizamos FROM que indica la tablas a las cuales van a pertenecer las columnas anteriores, aparte de estos dos operandos que son la base de los sentencias de consulta de datos se pueden usar otros operandos como WHERE que nos permite por ejemplo incluir condicionales como seleccionar solo las noticias a partir de una fecha, actividades que todavía no han tenido lugar o relacionar dos tablas que tengan en común una columna como puede ser un id y ORDER BY que nos permite elegir el criterio con el cual se ordenaran los resultados. Podemos ver un ejemplo de ello en la figura 27.

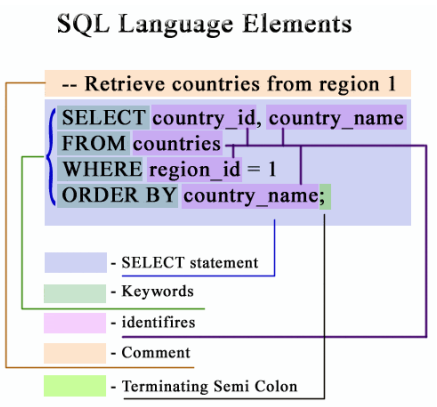

Fig. 27 Resumen de las consultas SELECT

Una vez sabemos cómo se van a producir las consultas de datos con las que se trabaja se puede empezar a ver como se realizan en PHP y como trabajar con los datos que nos proporcionan.

Tanto en la figura 28 como en las anteriores se puede apreciar que en primer lugar para incluir código PHP debemos incluir las etiquetas de comienzo y final de PHP, después importamos el código del archivo que nos conecta con la base de datos, declaramos cual será la consulta que se ejecutará, en este caso, extraeremos toda la información relacionada con la tabla Actividad, esta información estará ordenada a través de los valores de "tipo_evento" agrupando de esta manera las actividades según de que tipo son y finalmente la ejecutamos con el método query, dado que no se modifican los datos y asignamos el resultado a una variable denominada result.

```php
<?php
include "partials/db.php";
$sql = "SELECT * FROM actividad ORDER BY tipo_evento";
$result = $conn->query($sql);
?>
```

Fig. 28 Parte del código PHP del archivo de la ventana de actividades

Una vez se ha recopilado la información sobre las actividades solo falta hacer uso de ella, para ello nos podemos apoyar en la figura 29, en ella se puede ver como se procede, en primer lugar, volvemos a abrir las etiquetas para incrustar código PHP, dado que la tabla de actividades cuenta con una columna denominada tipo_evento, se creará una variable auxiliar denominada tipo cuyo uso se verá más adelante. También inicializaremos una variable denominada i con valor 0 que utilizaremos para iterar en el bucle.

En primer lugar, comprobaremos si el número de filas que hemos extraído, correspondiendo cada fila a una actividad, es mayor de 0, si no es así se mostrará un mensaje que comunique que actualmente no se encuentra disponible ninguna actividad. En caso contrario, iteraremos sobre las filas haciendo uso del método fetch_assoc() que cada vez que es llamado recupera una fila del array hasta que llegue a la última, guardaremos el valor de cada fila en una variable denominada row.

En la primera actividad que se recupera lo que haremos será mostrar un título con el tipo de actividad del que se trata y asignaremos el valor de este a la variable tipo, en caso de que no sea la primera actividad comprobaremos si el tipo de la actividad sobre la que se está iterando se ha mostrado ya, si no es así se mostrará y le asignaremos el valor a la variable tipo_evento. Dado que las actividades están agrupadas según su tipo no se correrá el riesgo de repetir el título de alguna actividad.

En cada iteración del bucle se creará una actividad a través de la estructura común de código que tienen todas, modificando los detalles específicos a través de los valores de cada uno de los atributos de la actividad, a los que accederemos refiriéndonos a la variable "row" seguida del nombre del atributo entre corchetes, que es la manera de acceder a los valores de la columna, para mostrarlos en la página haremos uso de echo que nos permite mostrar código HTML dentro de código PHP y se concatenará con los valores de la actividad con el operando '.'.

```php
<?php
    $tipo = "";
    $i = 0;
    if ($result->num_rows > 0) {
      // output data of each row
      while($row = $result->fetch_assoc()) {
          if($i==0){
              echo '<div class="col-sm-6 p-3 "><h1>' . $row["tipo_evento"] . '</h1>';
              $tipo = $row["tipo_evento"];
          }
          if($tipo != $row["tipo_evento"]){
              $tipo = $row["tipo_evento"];
              echo '</div>
                      <div class="col-sm-6 p-3 "><h1>' . $tipo . '</h1>';

          }
          echo '<div class="evento p-3">
                  <a href = "/actividad/' . $row["id"] . '" style="text-decoration:none;"><p class="text-left text-dark mb-0">' . $row["titulo"] . '</p></a>
                  <p class="text-left text-dark mb-0">' . $row["descripcion"] . '</p>
                  <p class="text-left text-dark">' . $row["fecha"] . '</p>
                </div>';
          $i++;
      }
    } else {
      echo "<h1>No hay actividades</h1>";
    }
  ?>
```

Fig. 29 Creación de las actividades a partir de los datos recuperados

Una vez se ha mostrado como extraer valores de una tabla y mostrarlos por pantalla, podemos hacernos una idea de cómo será el proceso de realizarlo con el resto de ventanas de las que está compuesta la página, dado que la base sigue siendo la misma, extraer los valores de la tabla, dado que no existen relaciones entre dos o más tablas todas las consultas realizadas seguirán el mismo patrón que el mostrado anteriormente, extraer la estructura común de los elementos como pueden ser las noticias, publicaciones e iterar sobre los resultados haciendo uso del código común y rellenando los detalles específicos de cada elemento con los atributos del elemento de la iteración actual.

3.1.5 Implementación de los formularios con PHP

En este apartado se abordará los formularios presentes en las ventanas de “contacta” y de la “newsletter”, en estas los usuarios podrán escribir mensajes y suscribirse a la newsletter respectivamente. Cada formulario tendrá un botón que cuando sea pulsado lo que hará será enviar los datos recogidos en sus campos de texto e introducirlos en la base de datos.

Para insertar datos tendremos que hacer uso de la orden INSERT, donde la palabra clave INTO indicará la tabla a la cual añadiremos los datos y después entre paréntesis las columnas en las que introduciremos los datos, seguido de ello la palabra VALUES después de la cual se pondrá entre paréntesis los valores de cada una de las columnas en el mismo orden que se han expuesto anteriormente, como se puede apreciar en la figura 30.

Fig. 30 Ejemplo de uso del comando INSERT

Para realizar este tipo de acciones nos apoyaremos tanto en PHP como en HTML, en la ventana de la newsletter crearemos un formulario con un input en el cual los usuarios introducirán su email y cuando pulsen el botón de suscribirse se enviará el valor de dicho campo al archivo addNewsletter mediante el método POST como se aprecia en la figura 31.

```
<form action="partials/addNewsletter.php" method="POST">
    <div class="input-group">
        <input type="email" name="email" class="form-control" placeholder="Introduce tu email">
        <span class="input-group-btn">
            <button class="btn" type="submit">Suscribete Ahora</button>
        </span>
    </div>
</form>
```

Fig. 31 Formulario de la ventana de la newsletter

En la figura 32 podemos ver el código del archivo addNewsletter, en el cual asignamos a la variable email el contenido que se ha recibido a tráves del método POST, preparamos la consulta SQL con la función “prepare”, la cual nos permite usar el carácter ? para sustituirlo posteriormente por los valores reales a traves de “bind_param”.

Esto nos permitira añadir una capa de seguridad a la página puesto que si en este caso hacemos uso del método NoQuery y creamos una cadena en la cual usamos la variable email, se puede correr el riesgo de que la variable tenga comillas que cierren la consulta y ejecuten otras ordenes con lo cual tendríamos una inyección de código SQL que permitiría una modificación de los datos.

Una vez hemos sustituido los valores de la consulta y esta lista podemos ejecutarla, una vez ejecutada podemos cerrar este proceso y redirigir la vista a la localización de la ventana de newsletter de nuevo.

```
<?php
include 'db.php';
$email = $_POST["email"];
$stmt = $conn->prepare("INSERT INTO newsletter (email) VALUES (?)");
$stmt->bind_param("s", $email);
$stmt->execute();
$stmt->close();
header("Location: ../newsletter");
?>
```

Fig. 32 Contenido del archivo addNewsletter

También podemos observar cómo realizamos un proceso similar al anterior con los mensajes que se pueden crear en la ventana de contacta, en ella crearemos otro formulario, en este caso, compuesto de 3 campos, el nombre, email y contenido del mensaje del usuario. Como se puede observar en la figura 33 creamos un formulario con 3 inputs, uno para cada campo y cuya acción apunta al archivo addEscribenos el cual recibirá el valor de estos campos mediante POST.

```
<div class="card-body">
  <form action="partials/addEscribenos.php" method="POST">
    <div class="form-group">
      <input class="form-control" name="email" type="text" placeholder="Email">
    </div>
    <div class="form-group">
      <input class="form-control" name="nombre" type="text" placeholder="Nombre (Opcional)">
    </div>
    <div class="form-group">
      <textarea class="form-control" name="mensaje" cols="30" rows="10" placeholder="Message"></textarea>
    </div>
    <div class="form-group">
      <button class="btn btn-primary btn-block" type="submit">
        Enviar
      </button>
    </div>
  </form>
</div>
```

Fig. 33 Formulario de la ventana de contacta

En la figura 34 se puede ver las acciones que realiza addEscribenos y que son las misma que el archivo anterior, importar el archivo de la base de datos, asignar el valor de las variables obtenidas a través de POST, preparar la sentencia de SQL esta vez sustituyendo las interrogantes por los tres valores recibidos, ejecutar la sentencia y finalmente se redirige la vista a la ventana de contacta.

```
<?php
include 'db.php';
$email = $_POST["email"];
$nombre = $_POST["nombre"];
$mensaje = $_POST["mensaje"];
$stmt = $conn->prepare("INSERT INTO contacta (email,nombre,mensaje) VALUES (?,?,?)");
$stmt->bind_param("sss", $email, $nombre, $mensaje);
$stmt->execute();
$stmt->close();
header("Location: ../contacta");
?>
```

Fig. 34 Contenido del archivo addEscribenos

Una vez hemos finalizado tanto la implementación de las funcionalidades que nos permiten extraer los datos de la base de datos y mostrarlos por pantalla como las que permite a los usuarios interactuar con el sistema suscribiéndose a la newsletter y escribiendo mensajes podemos dar por concluida la parte en la cual hemos creado el código PHP que nos permite tener ventanas que muestren un contenido dinámico.

3.1.6 Implementación del .htacces

En este apartado se hará una explicación de que es el archivo .htaccess y como se utiliza para mejorar la experiencia en la página web.

En primer lugar, tendremos que hacer una breve presentación de en qué consiste este archivo, los htacess [13] cuyo nombre deriva de Hypertext Access se tratan de archivos de configuración a nivel de directorio que son soportados por una variedad de servidores de páginas webs usados para modificar los problemas de acceso a la página como pueden ser acortamientos de urls o redireccionamientos.

Estos archivos permiten crear un nuevo tipo de configuración que se aplica únicamente en el directorio en el que se encuentran o en el resto de los subdirectorios, por lo tanto, puede haber más de uno en la misma página, por lo cual también se les conoce como archivos de configuración distribuidos.

El uso que se le quiere dar es el de mejorar el SEO de la página y para ello lo que se hará es crear urls que sean más amigables y que tanto los motores de búsqueda como los usuarios puedan entender, por ejemplo, cuando un usuario entre en la página de contacta en vez de ver que se encuentra en el archivo contacta.php simplemente vea que se encuentra en la sección contacta sin la extensión del archivo presente.

Ahora se comenzará con la creación del archivo, en primer lugar, lo que tendremos que hacer es usar la directiva “RewriteEngine on” con lo cual nos aseguraremos de que los siguientes cambios tengan efecto al haber activado el motor de sobreescritura, después especificaremos dos condiciones.

La primera condición se encargará de comprobar de si la dirección a la cual se está intentando acceder corresponde a un directorio, si es así no se seguirá ejecutando el código.

La segunda condición se encargará de comprobar si existe un archivo PHP cuyo nombre coincida con la dirección, si no es así no se seguirá ejecutando el código.

Las siguientes reglas que se crean se encargarán de sobrescribir las urls, en cada una de ellas la estructura será la misma, si la dirección corresponde con el nombre de alguno de los archivos que hemos creado previamente le indicaremos al compilador de que cargue ese mismo archivo, así por ejemplo si se accede a la sección información de nuestra página web, se comprobará si existe tal directorio, como no es así continuara a la siguiente condición que comprobará que existe un archivo php cuyo nombre coincide y después se encontrará ante la regla que sobrescribirá información por información.php accediendo así al archivo.

Se puede comprobar el código del archivo .htaccess en la figura 35.

```
RewriteEngine on

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_FILENAME}\.php -f

RewriteRule ^informacion  informacion.php

RewriteRule ^actividades  actividades.php

RewriteRule ^noticias  noticias.php

RewriteRule ^publicaciones  publicaciones.php

RewriteRule ^newsletter  newsletter.php

RewriteRule ^escribenos  escribenos.php

RewriteRule ^personas  personas.php

RewriteRule ^justificacion-y-objetivos  justificacion.php

RewriteRule ^contacta  contacta.php

RewriteRule ^enlaces-de-interes  enlacesinteres.php

RewriteRule ^index  index.php
```

Fig. 35 Contenido del archivo .htaccess

3.1.7 Creación de una API REST

Dado que la página web va a acceder a los datos que son mantenidos por una base de datos que se encuentra alojada en el mismo servidor que la página, el cual nos provee el STIC y nuestra aplicación móvil tiene que acceder a estos datos y ser capaz de modificarlos surge la cuestión de como poder compartir los datos entre la aplicación móvil y la web.

Una de las herramientas que nos permite compartir información entre dos aplicaciones diferentes son las APIs (Interfaz de programación de aplicaciones), que son un conjunto de funciones y métodos que nos ofrece una cierta biblioteca para ser utilizada por otro software representando las capacidades de comunicación entre componentes software.

Las API generalmente se refieren a ambos lados de los sistemas informáticos que se comunican a través de una red, tanto los servicios API ofrecidos por el servidor como la API ofrecida por el cliente como puede ser un navegador web.

Este será el tipo de comunicación elegido con el cual se podrá hacer que la aplicación móvil se comunique con la base de datos de forma directa. Sin embargo, existen diferentes tipos de APIs entre las cuales elegir por lo que a continuación se hará una breve explicación de estas.

Antes de nada, hay que explicar que la parte de la API soportada por el servidor es una interfaz programada para definir un sistema de mensajes respuesta-petición y la cual usualmente se define como un servicio web, existen diferentes tipos de servicios web, aunque los dos más dominantes [14] son la SOAP y REST.

La API SOAP (Protocolo de acceso a objetos simples) es el modelo más conocido, cada operación que es proveída por el servicio esta explícitamente definida además de la estructura XML de la petición y de la respuesta para esa operación. Cada parámetro de entrada se encuentra definido y ligado a un tipo de dato.

Todos los intercambios están codificados en WSDL (Lenguaje de descripción de los servicios web) que se puede entender como un contrato entre el proveedor y el consumidor del servicio.

Sin embargo, esto provoca que, si se hace necesario hacer cambios a la API, aunque sean tan pequeños como añadir un nuevo parámetro el WSDL debe cambiar, y cuando el WSDL cambia también cambian los clientes, con lo que todos los consumidores del servicio deben recompilar la aplicación de cliente con el nuevo WSDL.

Una de las ventajas que ofrece este tipo de API es la capacidad de ser transmitida a través de cualquier protocolo, aunque no supone una ventaja real dado que la mayoría de los servicios se ejecutan a través del protocolo HTTP.

Mientras que el otro tipo está formado por las REST API [17] (Transferencia de estado representacional) que se están convirtiendo en el tipo preferido para las APIs públicas, se basa en protocolos de comunicación sin estado, especialmente HTTP. Usa estructuras de datos como XML, YAML o cualquier otro tipo de formato legible por las maquinas, aunque JSON es el más ampliamente usado. Se les conoce también como API Restful. No existe un estándar para el formato de descripción.

En este caso se hará uso de una REST API debido a que son las más populares actualmente y son las que nos ofrecen un mayor número de posibilidades, la API se conectará a la base de datos de tipo MySQL y servirá de interfaz para que nuestra aplicación pueda interactuar con la base de datos como se muestra en la figura 36.

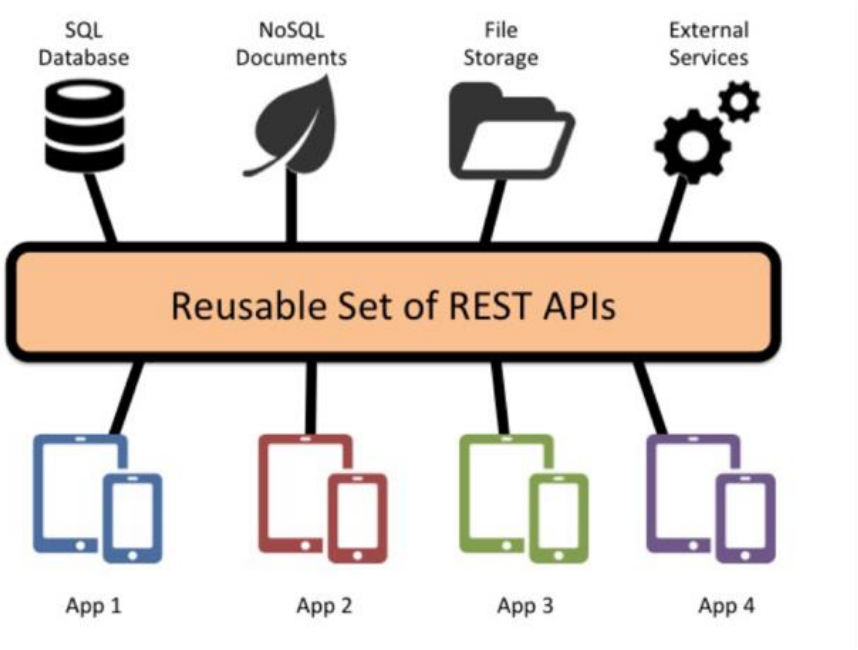

Fig. 36 Esquema de tipos de REST APIs

A continuación, se comenzará a explicar los métodos básicos de los que dispone una REST API, sin embargo, para entender cuál es el origen de estos antes se debe de explicar que estos métodos se basan en las peticiones HTTP.

HTTP [15] [16] es un protocolo el cual nos permite realizar una petición de datos y recursos como pueden ser documentos HTML y que supone la base de cualquier intercambio de datos en la Web. Define los pasos a seguir para cualquier interacción entre un servidor y un cliente.

En primer lugar, se abre una conexión TCP que se usará para enviar las peticiones y recibir las respuestas, el cliente abre una nueva o reúsa una existente para comunicarse con el servidor, después realiza una petición HTTP que son legibles en texto plano, se envía la petición al servidor, este la lee y envía una repuesta, el cliente la recibe y finalmente se cierra o se reúsa la conexión ya establecida para futuras peticiones.

Los mensajes en HTTP se clasifican en peticiones o respuestas, las peticiones como se puede observar en la figura 37 están compuestas por un método HTTP que define la acción que se desea realizar, la dirección del recurso pedido, la versión del protocolo HTTP, las cabeceras HTTP opcionales y el cuerpo del mensaje, necesario en algunos métodos.

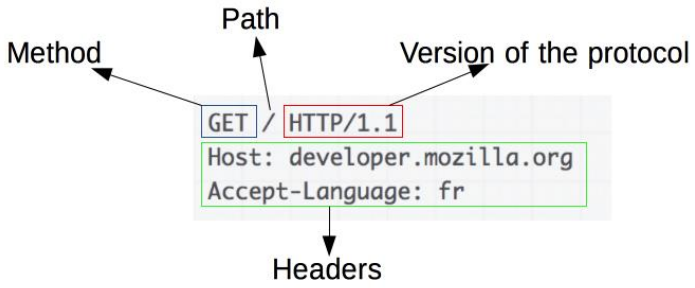

Fig. 37 Estructura de una petición HTTP

Las respuestas por otra parte están formadas por la versión del protocolo HTTP, el código de estado, un mensaje de estado, las cabeceras HTTP y opcionalmente el recurso que se ha solicitado. Se puede ver un ejemplo en la figura 38.

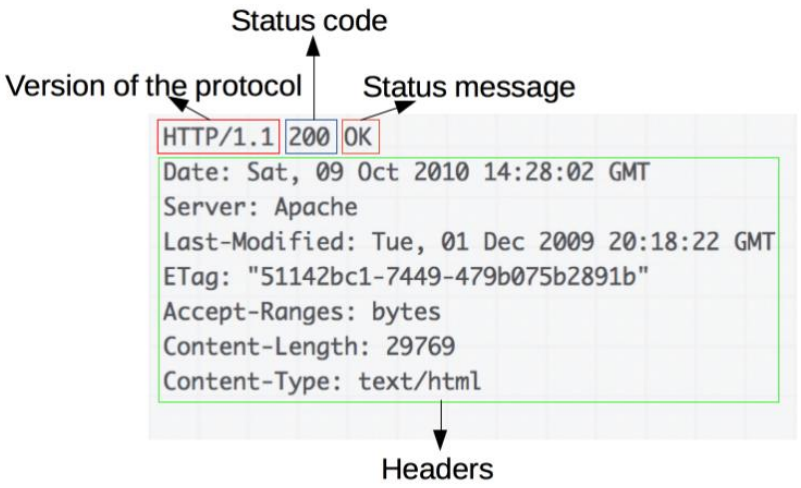

Fig. 38 Estructura de una respuesta HTTP

Estas serán las bases de nuestra API, en función de la dirección a la cual se dirija la petición, el método que se utilice y el mensaje que se envíe se creará una respuesta personalizada.

Una REST API [18] define principalmente 5 operaciones correspondientes a algunos de los métodos definidos en HTTP:

- Get: Se utiliza para únicamente recuperar la representación o información de recursos y no para modificarlas de ninguna manera, dado que no cambian el estado del recurso se dice que son métodos seguros, es el método que se utiliza cuando se visita una página web.
- Post: Se utiliza para crear nuevos recursos subordinados, en términos de una REST API se utilizan para crear un nuevo recurso en una colección, en nuestro caso sería añadir un nuevo objeto en nuestra base de datos. Si se crea se debería devolver un código de respuesta 201 con el nuevo recurso creado y su ubicación.
- Put: Se usan para actualizar un recurso existente, en caso de que no exista la API puede decidir entre crearlo o no, si se crea se envía un código 201, y en caso de que se haya modificado 200.
- Delete: Se utilizan para eliminar recursos, si se ha eliminado se devuelve un código 200, 202 si ha se puesto en cola o 204 si no se ha encontrado al recurso especificado.
- Patch: Se usan para actualizar de forma parcial un recurso, mientras que PUT se tiende a usar para reemplazar el recurso de forma completa. Sin embargo, el soporte para esta operación no es universal, PHP, Tomcat o Django entre otras no soportan esta operación.

Dado que se trabaja en un servidor Apache con PHP solo se usa las 4 primeras operaciones. Como formato de representación de datos se escogerá JSON puesto que es el más usado y ofrece la información de forma más visual, con lo cual creamos una REST API con una estructura igual a la representada en la figura 39.

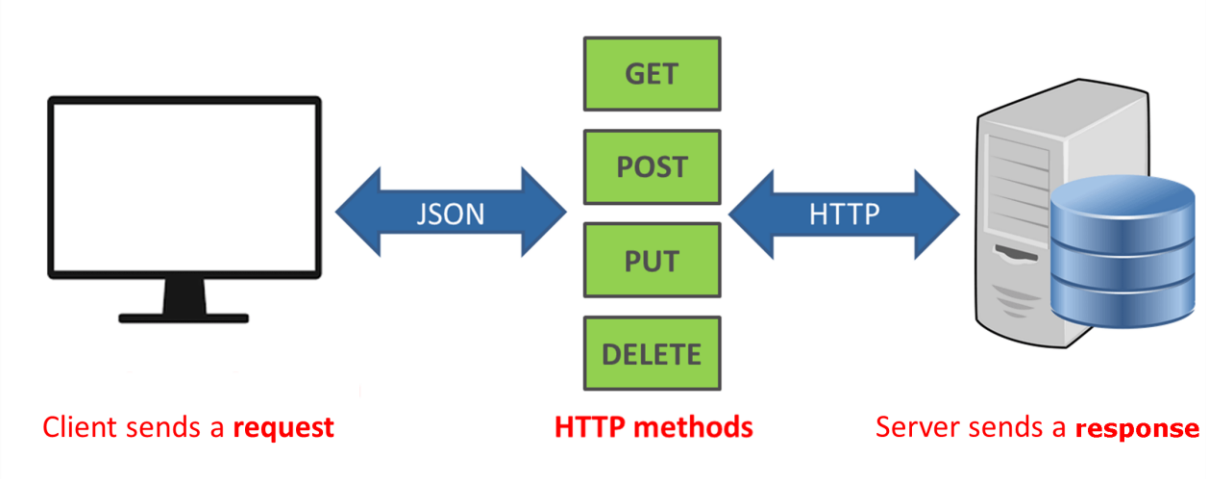

Fig. 39 Estructura de la REST API

Una vez tenemos definido el funcionamiento de una REST API podemos definir que es JSON y como se representa. Se trata de un formato de datos basado en texto que sigue la sintaxis de objetos de JavaScript en el cual se pueden incluir los mismos tipos de datos básicos que en un objeto estándar de JavaScript (cadenas de caracteres, números, booleanos…)

Los objetos se encuentran definidos entre llaves “{}” en las que se sitúan las propiedades y los valores de estas mismas que están separadas por los dos puntos, tanto para los nombres de las propiedades como para las cadenas se requiere el uso de comillas dobles. Las propiedades de un objeto como los elementos de un array se separan a través de comas. En la figura 40 se puede ver un array formado por dos objetos JSON cada uno de los cuales representa una noticia y que están constituidos por un id, titulo, fecha, img y enlace.


```
[
  {
    "id":"1",
    "titulo":"¿Agua para todos? Cobos promete soluciones a vecinos y regantes de Almería",
    "fecha":"15-02-2021",
    "img":"https:\/\/lh3.googleusercontent.com\/dJ0YQf8c4r4Dpqh5eELIJ1h8LPefhLtd1DNgG9bSJjT_h55SWtRq-CxbryH2NVXUJWhyBL-LYmK3wdBJJmygsUQ3kVudCgUMvF4=w840-h473-n-rw-e365",
    "enlace":"https:\/\/www.lavozdealmeria.com\/noticia\/12\/almeria\/201885\/agua-para-todos-cobos-promete-soluciones-a-vecinos-y-regantes-de-almeria"},
  {
    "id":"2",
    "titulo":"La catedra del agua en agricultura, regadio y agroalimentacion de la UAL comienza a fluir según sus objetivos",
    "fecha":"30-10-2020",
    "img":"https:\/\/news.ual.es\/wp-content\/uploads\/2020\/10\/FOTO-1-OCTUBRE-30-2020-REUNI%C3%93N-C%C3%81TEDRA-AGUA-Y-REGAD%C3%8DO-696x464.jpg",
    "enlace":"https:\/\/news.ual.es\/formacion\/la-catedra-del-agua-en-agricultura-regadio-y-agroalimentacion-de-la-ual-comienza-a-fluir-segun-sus-objetivos\/"
  }
]
```


Fig. 40 Ejemplo de un array de dos elementos en formato JSON

Para acceder a los valores de un objeto JSON simplemente se tiene que colocar detrás del nombre de la variable que almacena el objeto JSON el nombre del atributo cuyo valor queremos conocer entre comillas simples dentro de un corchete. Por ejemplo, si tuviéramos una variable noticia que representara un objeto como los definidos en la figura 40 para acceder al valor del atributo del id solo tendríamos que usar noticia[‘id’].

Una vez tenemos definido tanto el funcionamiento de la API REST como el formato de los datos con el que se trabaja podemos empezar con la implementación de esta.

Se creará una ubicación específica para cada tabla de la base de datos que tenga definido los 4 métodos (PUT, POST, DELETE, GET) excepto para la tabla de los mensajes de la sección contacta y los emails de la newsletter, dado que los datos surgen de la interacción de los usuarios y se pueden crear nuevos registros mediante el uso de los formularios las operaciones de POST y de PUT no se contemplan con estos datos.

En primer lugar, se crea una nueva carpeta en la raíz de nuestro directorio denominada “api” donde se guarden todos los archivos relacionados con la API REST, constará de un archivo por cada tabla y otro adicional denominado utilidades en el cual se realicen todas las peticiones a la base de datos.

Este último archivo albergará todos los métodos que realicen las consultas a la base de datos, se puede observar en la figura 41 los métodos relacionados con la tabla de actividades.

```php
<?php

require '../partials/db.php';

function getTodasLasActividades(){
    $query = 'SELECT * from actividad';
    return json_encode(Query($query), JSON_UNESCAPED_UNICODE);
}

function getActividad($id){
    $query = 'SELECT * from actividad where id = ' . $id;
    return json_encode(Query($query), JSON_UNESCAPED_UNICODE);
}

function deleteActividad($id){
    $query = 'DELETE from actividad where id = ' . $id;
    NonQuery($query);
}

function postActividad($json){
    $query = "INSERT into actividad (tipo_evento,titulo,fecha,descripcion,enlace_pdf,programa) VALUES ('" . $json['tipo_evento'] . "','"  . $json['titulo'] . "','"  . $json['fecha'] . "','"  . $json['descripcion'] .
    return NonQuery($query);
}

function putActividad($id,$json){
    $query = "UPDATE actividad SET tipo_evento = '" . $json['tipo_evento'] . "', titulo = '" . $json['titulo'] . "' , fecha = '" . $json['fecha'] . "' ,descripcion = '"  . $json['descripcion'] . "' ,enlace_pdf = '"
    return NonQuery($query);
}

function getTodasLasNoticias(){
    $query = 'SELECT * from noticia';
    return json_encode(Query($query), JSON_UNESCAPED_UNICODE);
}
```

Fig. 41 Parte del código del archivo utilidades

En este archivo se encuentra en primer lugar el método getTodasLasActividades() que hará una consulta que devuelva todas las actividades que almacena la base de datos y devolverá el resultado de esta consulta, este resultado lo codificaremos en formato JSON a través del método json_encode al cual le añadiremos el parámetro JSON_UNESCAPED_UNICODE para que el resultado este expresado en UTF-8. Esta será la función que se llamará cuando el usuario realice una petición GET para recuperar los datos de todas las actvidades.

El método getActividad() cuenta con un parámetro id que representa el id de la actividad cuyos datos se quieren recuperar y que serán devueltos de la misma forma que en el método anterior.

El método deleteActividad realizará la eliminación de la actividad cuyo id se le ha pasado por parámetro, para ello primero creará la consulta y después la ejecutará con el método NoQuery definido anteriormente.

El método postActividad se encargará de introducir en la base de datos una nueva actividad a partir de un objeto que se la pasa por parámetro, el método rellenará los valores del nuevo registro a partir de acceder a las diferentes propiedades del objeto y ejecutando finalmente la consulta con el método NoQuery.

El método putActividad realizará la actualización de la actividad cuyo id se le pasa por parámetro y para ello utilizará los nuevos valores que se le han pasado a través del objeto.

Repetiremos este proceso creando estos 5 métodos para todas las tablas excepto para las de newsletter y contacta, las cuales solo contarán con los relacionados con DELETE y GET, de esta forma tendremos en un único archivo todos los métodos que nos permitirán realizar todas las operaciones CRUD sobre la base de datos.

Ahora se empezará con la creación de los archivos que se encargarán de gestionar los diferentes tipos de peticiones para cada tabla de la base de datos.

Como ejemplo tomaremos el archivo dedicado a la tabla actividades que se puede visualizar en la figura 42, en el que se puede observar que en primer lugar se importa el archivo utilidades en el cual se encuentran definidos todos los métodos de acceso a la base de datos.

A continuación, crearemos una estructura de tipo switch en el cual el objeto cuyo valor observaremos será el método de petición, para ello haremos uso del método _SERVER que se trata de un array que contiene una serie de información relativa a la petición como pueden ser los headers, la dirección o las localizaciones de los scripts.

Toda la información de este array es producida por el servidor web con lo cual no hay garantía de que todos estén definidos, aunque la mayoría de estas variables como se encuentran definidas por la especificación CGI/1.1 estarán disponibles siempre.

Como lo que nos interesa para definir la respuesta del servidor es el método que se utiliza en la petición lo que haremos será acceder al valor de este que se denomina REQUEST_METHOD.

En caso de que el valor de esta variable sea POST lo que haremos será leer el cuerpo de la petición a través del método de file_get_contents que leerá los contenidos de lo que se le especifique como parámetro y que en este caso será el cuerpo de la petición como se ha mencionado previamente. En este caso haremos uso del método json_decode para convertir el objeto JSON que leemos en un array, llamaremos al método postActividad del archivo utilidades y devolveremos como respuesta el resultado que nos devuelve el método, para lo cual haremos uso de print_r.

Si se trata del método GET, en primer lugar, comprobaremos si la en la dirección en la que se realiza la petición se especifica el valor de una id, si es así mostraremos la respuesta del método getActividad con el valor de la id como parámetro, si no es así haremos uso del método getActividades.

En el caso de PUT, dado que es necesario que se especifique el id de la actividad a actualizar comprobaremos que esté presente, si no es así enviaremos una respuesta indicando el error, si no, se llamará al método putActividad con el id que se ha especificado en la dirección y el resultado de decodificar el contenido del body del formato JSON a un array.

En caso de que sea DELETE, procederemos de la misma forma que en el caso anterior, comprobaremos que se ha especificado en la dirección el id de la actividad a eliminar, si no se avisará del error en la respuesta del servidor, en caso afirmativo llamaremos al método deleteActividad y se enviará como body de la respuesta el resultado de la función.

Como podemos observar, sea cual sea el caso del método con el cual creamos la petición, siempre y cuando esté dentro de los que están definidos en nuestra REST API devolveremos una respuesta, ya sea mostrando el resultado de la consulta que se ha ejecutado o el error que se ha cometido a la hora de realizar la petición y que ha provocado que la consulta no se ejecute.

Todo el contenido del archivo se puede mostrar con el código de la figura 42.

Tendremos que repetir este mismo proceso creando un nuevo archivo para cada una de las diferentes tablas en las cuales se modelen los comportamientos de la API REST dependiendo de a qué dirección se realice la petición y el método el cual se utilice.

Cabe mencionar que dado que se estaba trabajando desde un entorno local todo lo que se ha realizado está protegido por el hecho de que nadie puede acceder a la API, en caso de que esta se encuentre desplegada en un servidor real que esté funcionando y que sea accesible por cualquiera correremos el riesgo de que una persona cualquiera sea capaz de modificar los datos que están almacenados y así modificar el comportamiento de la página.

```php
<?php
include 'utilidades.php';

switch($_SERVER['REQUEST_METHOD']){

    case 'POST':
        $body = json_decode(file_get_contents('php://input'),true);
        print_r(postActividad($body));
    break;

    case 'GET':
        if(isset($_GET['id'])){
            print_r(getActividad($_GET['id']));
        }
        else{
            print_r(getTodasLasActividades());
        }

    break;

    case 'PUT':
        if(isset($_GET['id'])){
            $body = json_decode(file_get_contents('php://input'),true);
            print_r(putActividad($_GET['id'],$body));
        }
        else{
            echo 'No ID especificado';
        }
    break;

    case 'DELETE':
        if(isset($_GET['id'])){
            print_r(deleteActividad($_GET['id']));
        }
        else{
            echo 'No ID especificado';
        }
    break;
    default;
}
?>
```

Fig. 42 Contenido del archivo correspondiente a las respuestas de la API para la tabla actividades

Para evitar el problema de seguridad tendremos que encontrar una manera de asegurar las operaciones de actualizar, crear y borrar puesto que los resultados de realizar una petición a la API de tipo GET mostrará la misma información que la que se muestra en la página web, aunque las tablas de newsletter y contacta sí que tendrán que estar protegidas de estas peticiones también, puesto que se almacena información sensible como pueden ser nombres reales y correos electrónicos de los usuarios.

Para proteger nuestra REST API existe una numerosa serie de formas de conseguirlo [19], a continuación, se repasará alguna de ellas y la solución que finalmente se ha decidido adoptar.

La primera que se mostrará será una clave API, que se trata de probablemente la forma más simple y rápida, se utiliza básicamente en APIs cuyo acceso es limitado y no público, se basa en validar la clave que se puede encontrar tanto como parámetro como en el header de la petición.

Se comprobará que la clave esté definida en el header o como parámetro del body y que el valor de esta coincida con el esperado, en la figura 43 podemos observar como para hacer uso de una de las APIs ofrecidas por Google Cloud Platform se provee al usuario de una clave API especial para él.

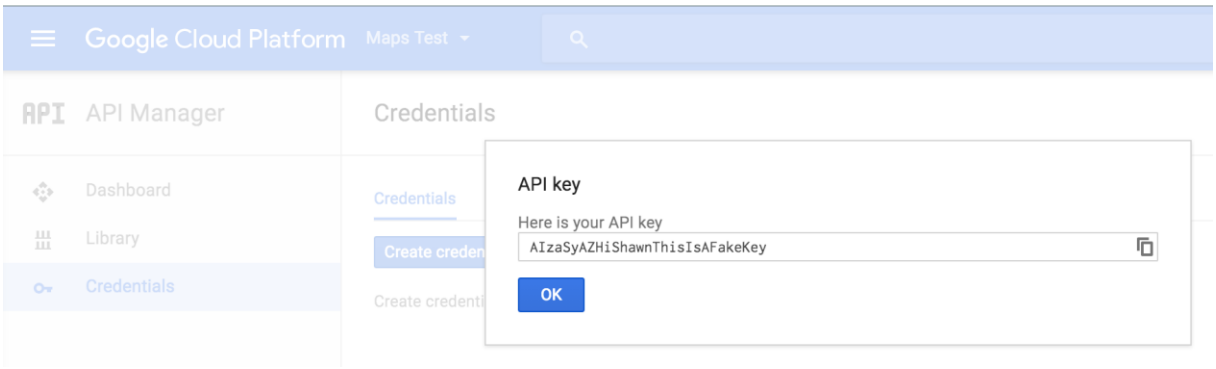

Fig. 43 Ejemplo de clave API ofrecida por Google Cloud Platform

Otra también de las más usadas es la autentificación básica, cuyo uso es muy similar al anterior, se basa en el uso del campo de autorización del header, el cual está formado por el resultado de codificar el usuario y la contraseña con la que se pretende acceder separados ambos por el carácter de los dos puntos, esto se codificará utilizando una variante de Base64, este resultado va después de la palabra Basic separada por un espacio. Podemos ver un ejemplo en la figura 44.

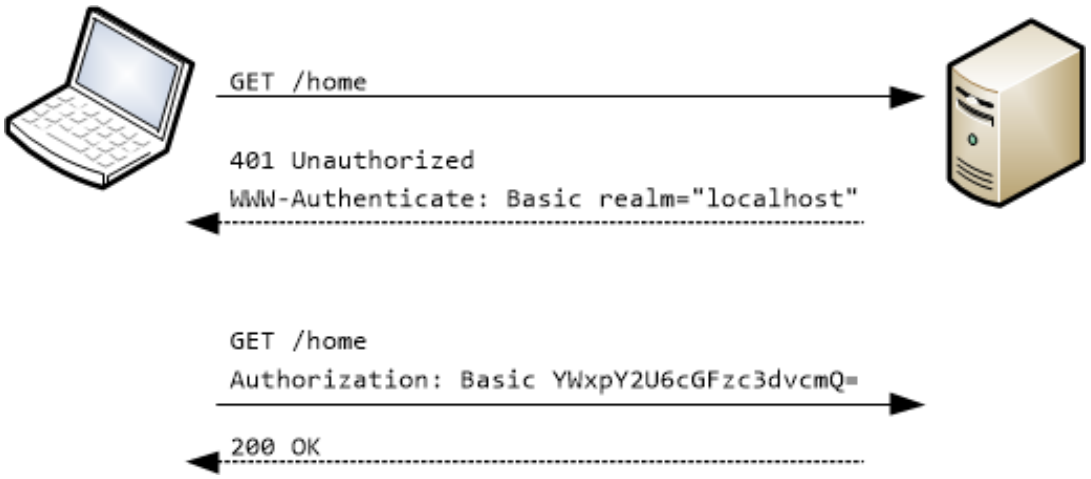

Fig. 44 Esquema de la autentificación básica

Otro método es Oauth2 [20] que nos permite compartir información entre sitios sin tener que compartir la identidad mediante el uso de tokens.

Una variación de la anterior es Oauth2 + JWT, en la cual cuando el usuario inicia sesión por primera vez se le devuelve un token de acceso que representa un mapa JSON que contiene toda la información del usuario y que esta codificada mediante el uso de Base64 y firmada por una clave privada.

Estas dos formas están más enfocadas en permitir que terceras partes como pueden ser aplicaciones o páginas webs accedan a la información del usuario contenida en otras páginas, pero sin la necesidad de proporcionarles ni la contraseña ni el usuario.

En la figura 45 podemos ver un ejemplo de su funcionamiento, el dueño del recurso es el usuario que da autorización a una determinada aplicación para acceder a su cuenta, el cliente es la aplicación que desea acceder a esa cuenta de usuario, el servidor de autorización se encarga de verificar la identidad de los usuarios y emitir una serie de tokens de acceso a la aplicación cliente mientras que el servidor de recursos será la API propiamente, es decir, el servidor que aloja el recurso protegido al cual queremos acceder.

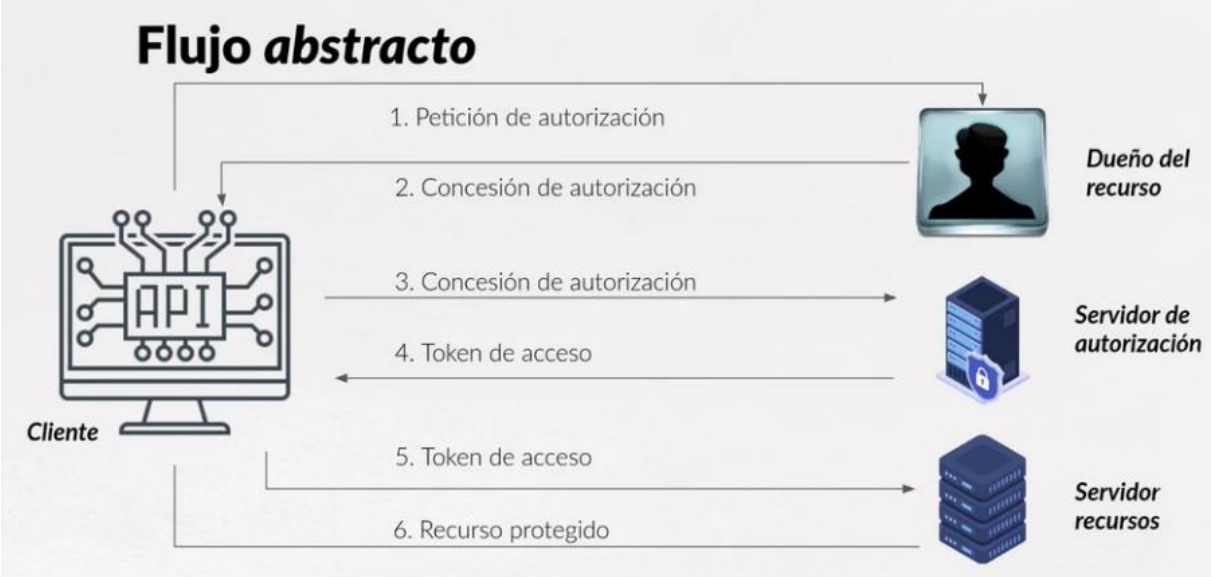

Fig. 45 Esquema de funcionamiento de OAuth2

Dado que la aplicación que se ha desarrollado está diseñada para que solo sea usada por el director de la cátedra y no se ha querido añadir más capas que impidan un fácil uso de la aplicación como puede ser añadir una ventana de inicio de sesión nos decantaremos por el uso de una clave API que ya va integrada en las peticiones que realizará la aplicación.

3.1.8 Modificación de la página web

Conforme iba avanzando tanto la implementación de la página como la de la aplicación móvil a través de las reuniones que tenían lugar con el director de la cátedra se iba obteniendo su opinión en el aspecto visual y funcional de la misma, en estas reuniones se realizaban una serie de modificaciones sobre las vistas existentes y la adición de nuevas vistas.

En este apartado repasaremos los cambios que han tenido lugar desde la implementación del primer diseño que se le mostró al director hasta la fecha actual mostrando cual es el aspecto de la página en este momento. Se ira repasando vista a vista cuales son las diferencias de cada una.

En cuanto a la vista general de la página se decidieron dos cambios en cuanto a los colores utilizados a lo largo de todas las ventanas , en primer lugar, el color de la barra de navegación pasa a ser el mismo que el utilizado en la página de la universidad para tener mayor consistencia dado que la página se desplegara en un subdominio de la UAL, y en segundo se decidió abandonar el gradiente de color de fondo y apostar por un color muy parecido al blanco pero menos brillante para hacer que no resalte tanto y que personas que estén acostumbradas a navegar en páginas con modo oscuro estén más cómodas con la experiencia.

En cuanto a la página de inicio, las imágenes que componen la galería se escogieron de entre las imágenes que se muestran en las páginas de las organizaciones que conforman la cátedra para evitar problemas legales a la hora del copyright. El resultado final se puede apreciar en la figura 46.

Fig. 46 Nueva implementación de la página de inicio

También se decidió incorporar a la ventana de inicio el menú de presentación con los dos párrafos hablando acerca de la cátedra y el footer con lo cual al navegar hacia abajo nos encontrábamos ante la figura 47, en ella podemos observar el nuevo color de fondo de la página además de que al footer ya se le han añadido una serie de enlaces de interés como pueden ser ministerios y consejerías además de la información relativa a la cátedra como puede ser el email o teléfono de contacto.

Fig. 47 Resultado de hacer scroll en la ventana de inicio

En esta página el director de la cátedra decidió eliminar la parte introductoria y crear una nueva vista con ella, además también expreso sus deseos de crear una serie de estructuras parecidas a gotas de agua, una para cada vértice en el cual se basa la cátedra, es decir, agua, agricultura, regadío y agroalimentación dispuestas en el orden en el que se encuentran en el título.

Cada gota tendría una imagen de fondo que indicará que apartado representan y serán “clickables”, al clicar sobre cada una abriremos un nuevo apartado en el que se mostrará información relacionada con esa parte. En la figura 48 se pueden observar los óvalos que se han creado y lo que sucede en caso de que por ejemplo, se clique en el apartado de regadío.

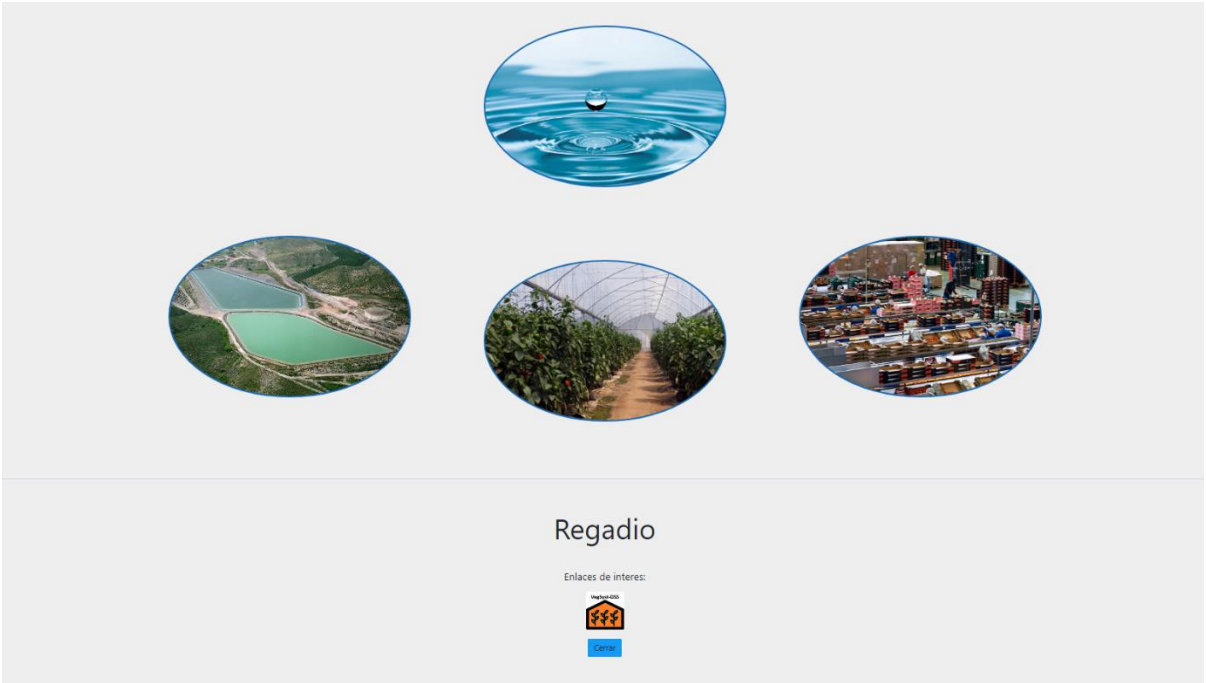

Fig. 48 Ventana de inicio al pulsar sobre el componente de regadío

Debido a que el director tras eliminar la parte de introducción quería que tanto la parte de la galería de imágenes como el footer se pudieran ver a primera vista en la ventana de inicio sin la necesidad de hacer scroll, se redujo la altura de las imágenes para que el footer se presentara inmediatamente después de las imágenes, también se quitó la parte de los enlaces de interés creando una nueva vista para ellos.

También se introdujeron otra serie de cambios, en primer lugar, en la barra de navegación se creó un menú desplegable para la información de la cátedra que está formada por la justificación y objetivos, los integrantes y los enlaces de interés. En la parte izquierda se sustituyó el nombre de la cátedra por el nuevo logo de la misma, y se añadió una nueva imagen en la galería correspondiente al cronograma de la cátedra que es la imagen que se aprecia en la figura 49, también se introdujo un título con el nombre de la cátedra que está presente en todas las imágenes excepto en la del cronograma.

Fig. 49 Resultado final de la ventana de inicio

En la figura 50 por otra parte podemos ver cual es el resultado de la página si hacemos scroll, en el se muestra como la animación de las imágenes se va produciendo, dicha animación se realiza a traves de la opción animation en CSS, en dicha animación podemos definir un tiempo total, en este caso de 20 segundos, que se trate de una animación infinita, es decir, al terminar vuelve a comenzar, para realizarla tenemos que especificar la anchura total del contenedor del componente con el que trabajamos, que en este caso será 500%, con lo cual cada imagen del total de 5 puede ocupar de forma completa la pantalla, y lo que haremos será indicar con porcentajes el momento en el cual movemos el elemento hacia la derecha un 100%, de esta manera se realizará la transición de una imagen a otra, en el caso del título haremos lo mismo solo que en este caso lo que haremos será manipular el atributo opacidad del titulo con lo cual cuando llega a la ultima imagen es de 0 y por tanto completamente transparente.

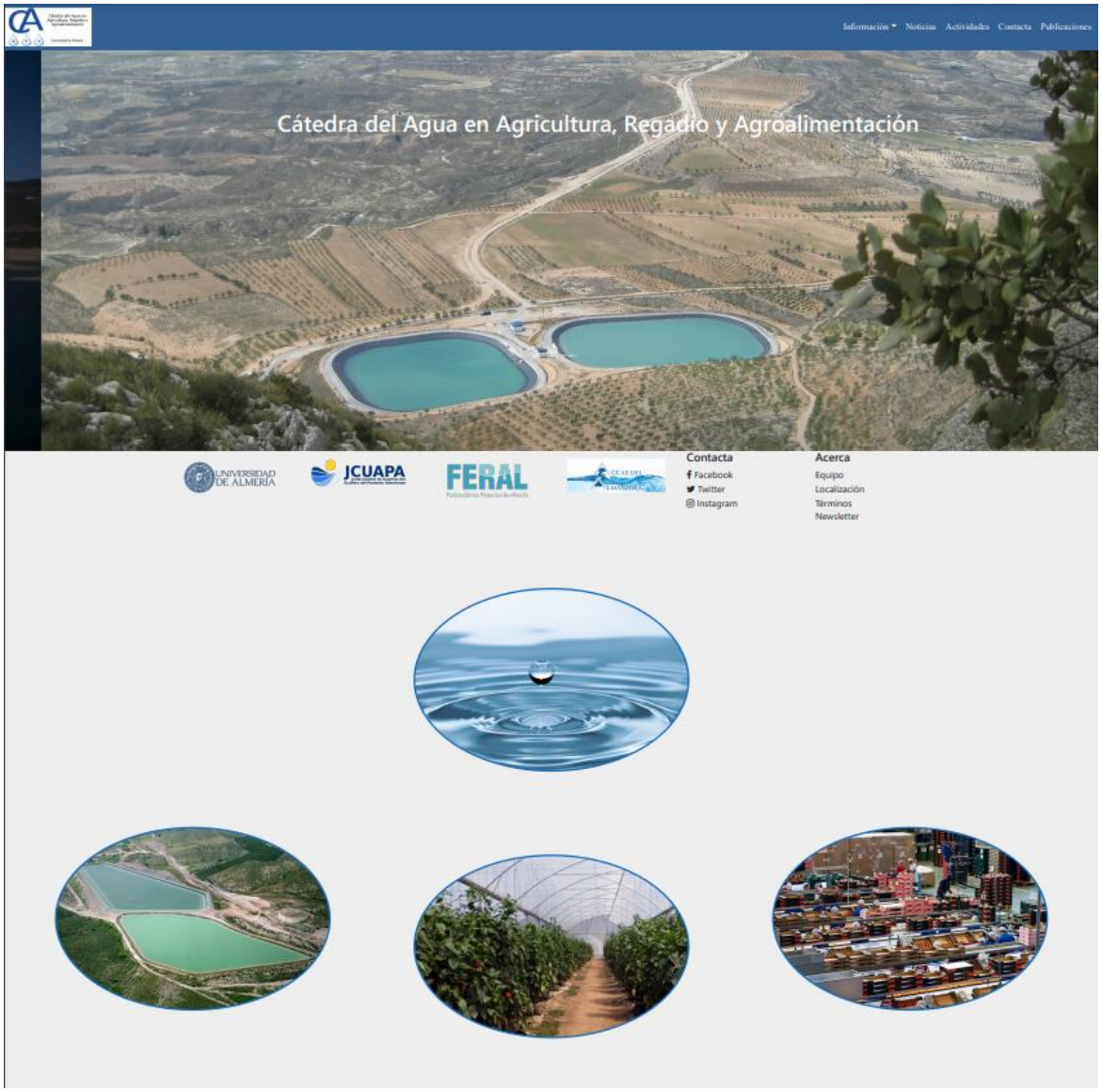

Fig. 50 Aspecto final de la ventana de inicio completa

Las ventanas de publicaciones y actividades son las mismas excepto el hecho de que ahora la barra de navegación y el color de fondo ha variado. Se muestran dichas ventanas en las figuras 51 y 52 cuando en la base de datos no hay ninguna actividad ni publicación disponibles (se encuentra en fase de pre-publicación).

Fig. 51 Resultado final de la ventana de publicaciones

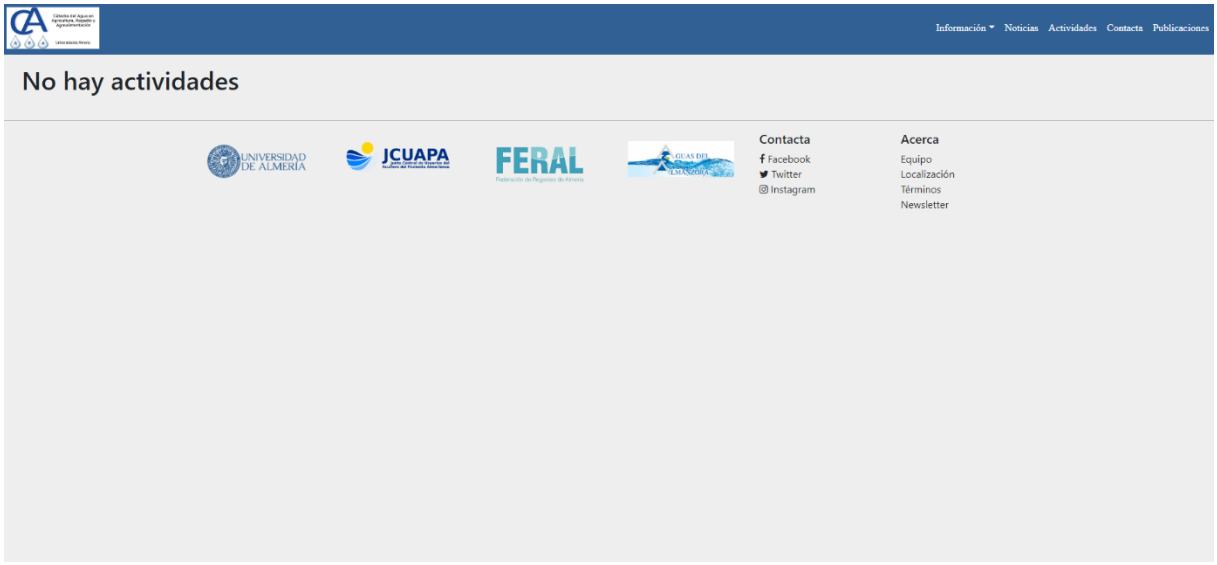

Fig. 52 Resultado final de la ventana de actividades

En cuanto a la ventana de contacta podemos ver como el formulario ha permanecido similar, simplemente se ha ampliado su anchura e incorporado a la izquierda la información de contacto fundamental de la cátedra. El resultado se puede visualizar en la figura 53.

En cuanto a la ventana de la subscripción de la newsletter ha permanecido igual como se puede apreciar en la figura 54.

En cuanto a la sección de información de la cátedra se han creado tres nuevas ventanas, una en la que se explica la justificación y objetivos de la misma, para la cual el director aportó un texto el cual se estructuro en tres columnas diferentes con el texto justificado intentando que la altura de todas las columnas sea la misma, el resultado se puede apreciar en la figura 55.

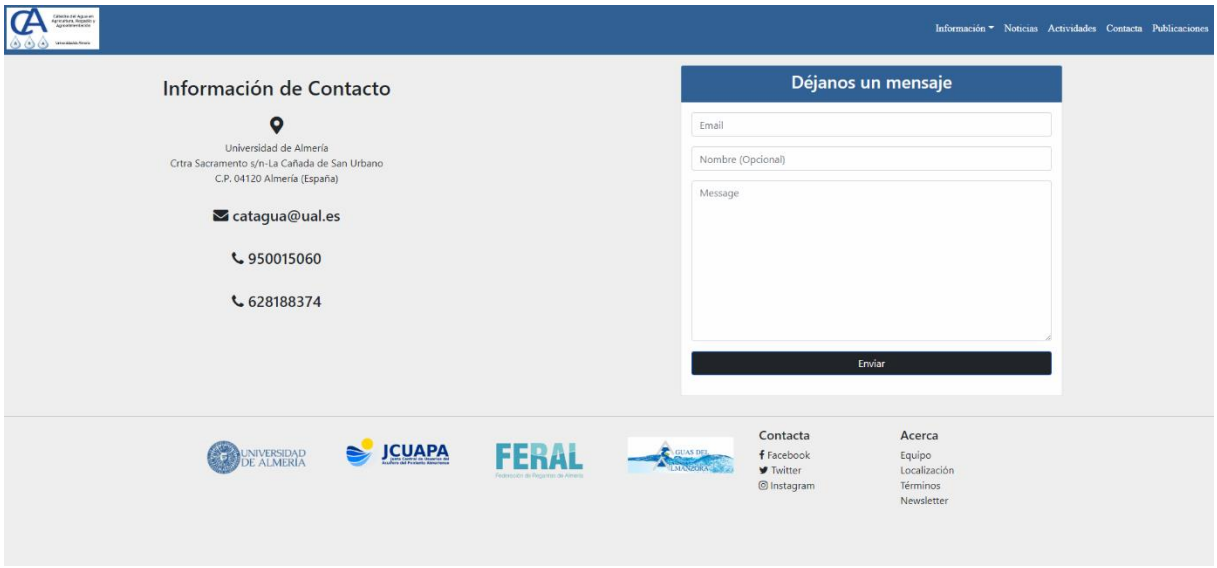

Fig. 53 Resultado final de la ventana contacta

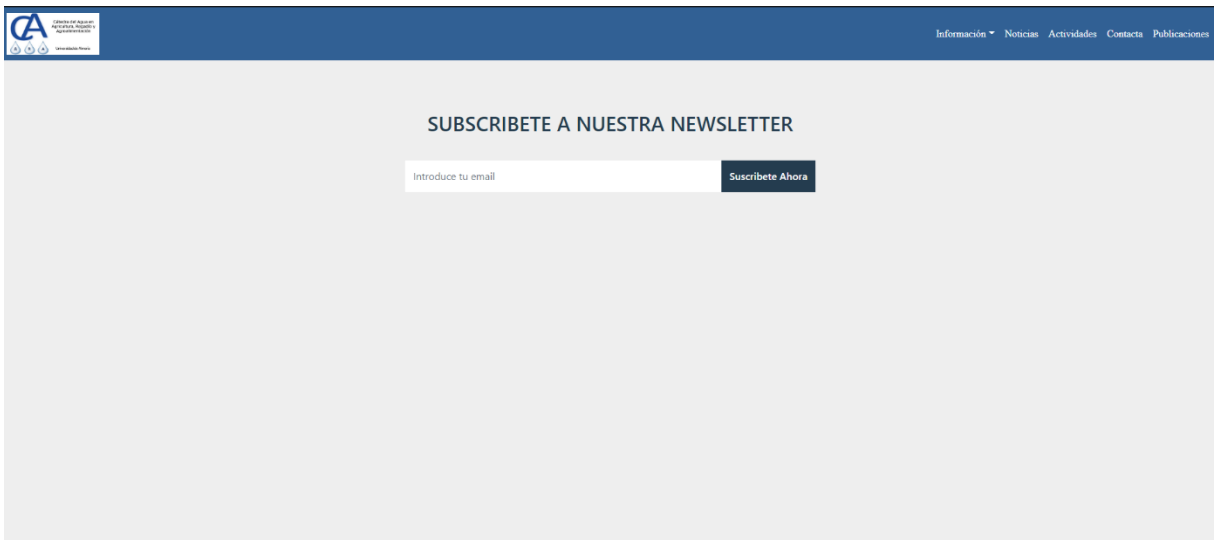

Fig. 54 Resultado final de la ventana de la newsletter

También se creó otra vista para los integrantes en el cual se muestra la posición que tiene cada uno con sus referencias, se puede ver en la figura 56.

En la figura 57 podemos observar los enlaces de interés, los cuales se disponen en una lista de enlaces que redireccionan a las correspondientes páginas.

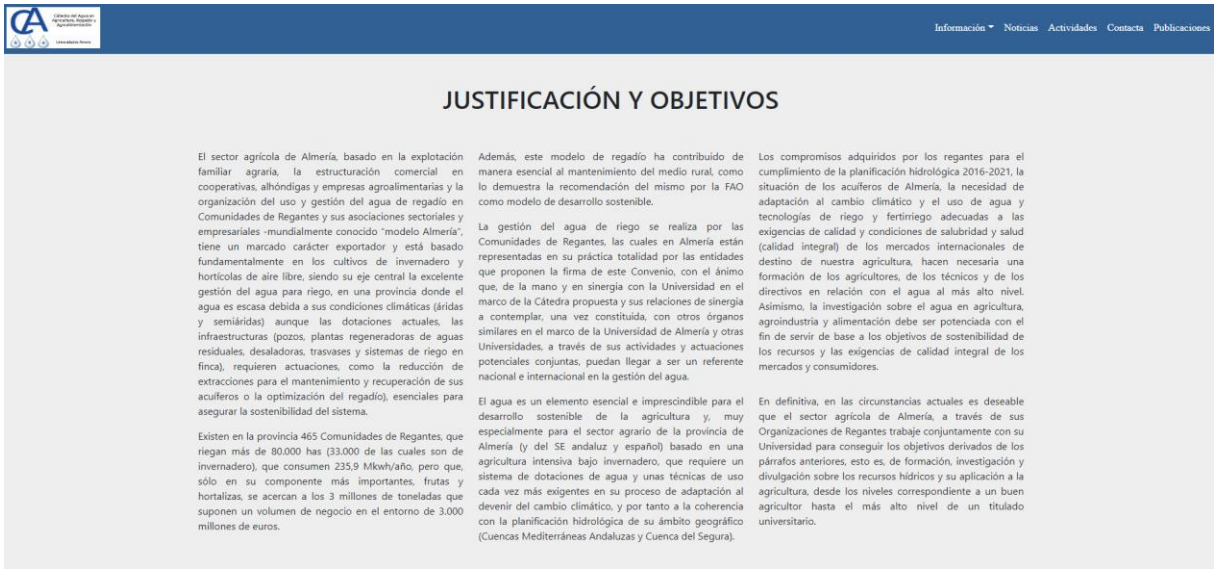

Fig. 55 Ventana de justificación y objetivos

Fig. 56 Ventana de integrantes de la cátedra

Fig. 57 Ventana de enlaces de interés

3.2 Aplicación Móvil

3.2.1 Diseño de la aplicación móvil

En este apartado como en el de diseño de la página web abordaremos el diseño de las ventanas y las distintas funcionalidades que tiene la aplicación móvil.

Dado que la idea básica de la aplicación es su utilización como panel de administrador de la página web a través de la interacción con la base de datos, tendrá que permitir ejecutar las funciones básicas CRUD sobre las tablas por las que está formada la base de datos. En este sentido la idea básica era la de presentar todos los elementos de los que consta una tabla en una ventana, poder seleccionar el elemento que el usuario desee, lo que le redireccionará a otra ventana donde poder modificarlo o eliminarlo, y otra ventana que funcionará a modo de formulario para la creación de un nuevo elemento.

Cabe mencionar que todo el desarrollo del prototipado del diseño se ha desarrollado como anteriormente mediante el uso de la herramienta Adobe XD escogiendo en este caso un Google Pixel 4 como modelo sobre el cual realizar el diseño.

La aplicación cuenta con una ventana de inicio en la cual podremos ver en la parte de arriba (toolbar) el nombre de la aplicación, Cátedra del Agua, y un botón a la izquierda para abrir la barra de navegación lateral. El toolbar tendrá un color azul claro mientras que el color del fondo de la pantalla será blanco, contando con un mensaje de inicio de momento. Podemos observar el diseño en la figura 58.

Fig. 58 Diseño de la ventana de inicio de la aplicación

En caso de que pulsemos sobre el botón y abramos la navegación lateral nos encontraremos ante enlaces para las otras 5 secciones de las que consta la aplicación, noticias, actividades, publicaciones, mensajes y newsletter. Podemos ver el resultado en la figura 59.

Fig. 59 Navegación lateral de la aplicación

En caso de que le demos click a la primera sección nos dirigiremos a la ventana de noticias, en la cual podremos visualizar las noticias que actualmente constan en la base de datos y que por tanto también son accesibles desde la página web, cada noticia constara de una imagen, del título, de la fecha y el enlace a la noticia que será accesible desde el título.

También se podrá encontrar el botón de añadir una nueva noticia, cabe mencionar que esta ventana consta de una función adicional que se abordará en el Complemento del Trabajo de Fin de Grado y que se encargará de sugerir una serie de noticias extraídas de unas páginas webs objetivo. Se puede ver el diseño de esta ventana en la figura 60.

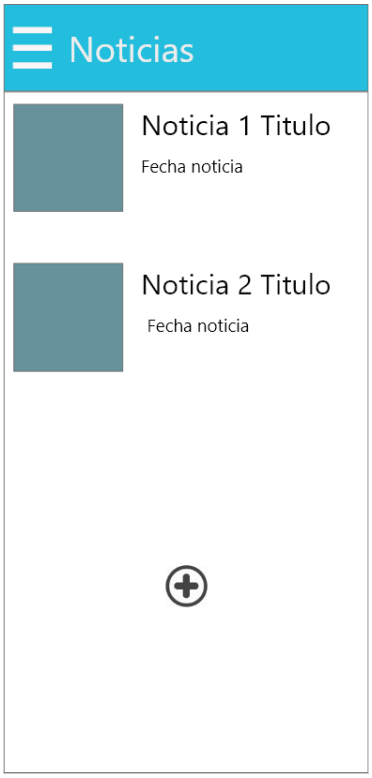

Fig. 60 Ventana de noticias de la aplicación

En caso de que hagamos click sobre una noticia nos redirigirá hacia otra ventana donde se podrá visualizar los datos de dicha noticia, que son título, fecha, enlace de la imagen y de la noticia, los datos aparecen en campos de texto como la pista del campo, con lo cual si el usuario al intentar modificar uno de los cambios quiere saber el valor que tenía antes solo tiene que borrar el contenido actual, en caso de que se haya modificado alguno de los atributos y si se desea actualizar la noticia solo hará falta pulsar el botón de modificar.

Esta misma ventana será la que también se muestre como formulario en caso de que el usuario decida añadir una nueva noticia a la base de datos y pulse el botón de añadir en la ventana de noticias, en este caso los campos de texto se encontrarán vacíos y el botón tendrá como título crear en vez de modificar.

De esta forma podremos hacer uso del mismo diseño para dos funcionalidades diferentes, como se puede ver en el diseño base en la figura 61.

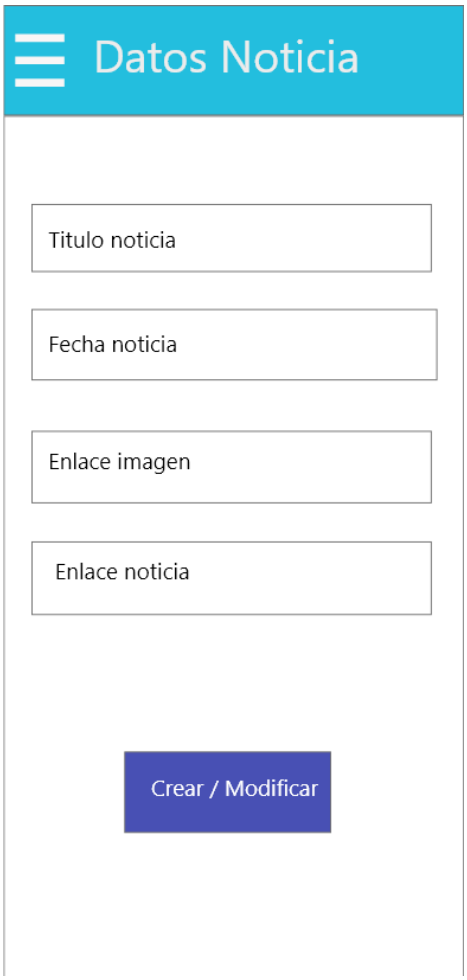

Fig. 61 Ventana de crear nueva noticia/ detalles de la noticia

Con lo anterior concluimos las ventanas necesarias para gestionar lo relativo a las noticias, a continuación, continuaremos con las actividades, que tendrán un planteamiento similar al anterior, en este caso cada actividad estará compuesta por un título, fecha y descripción con un botón que nos permitirá crear una nueva. Podemos ver el resultado en la figura 62.

En caso de que cliquemos sobre alguna de las actividades nos llevará hacia una nueva ventana donde se mostraran los datos, en esta ventana podremos modificar el valor de estos atributos y si pulsamos sobre el botón los actualizaremos con los nuevos valores introducidos. Esta ventana como la anterior también será la que se muestre si pulsamos sobre el botón de crear una nueva actividad, solo que ahora se mostrarán los campos de texto vacíos y el botón que antes era modificar ahora es crear. Podemos ver como luce el diseño en la figura 63.

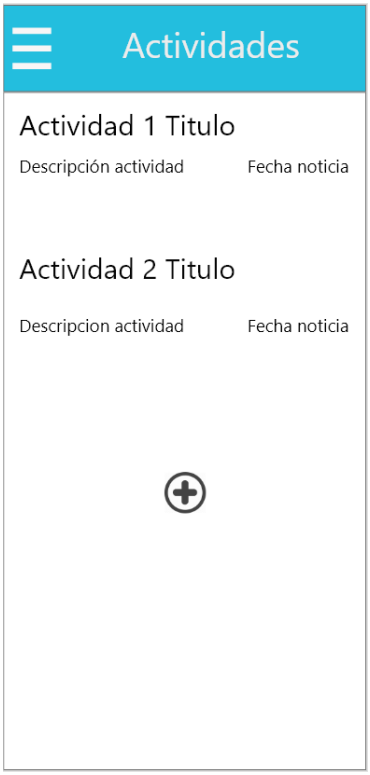

Fig. 62 Ventana de las actividades de la aplicación

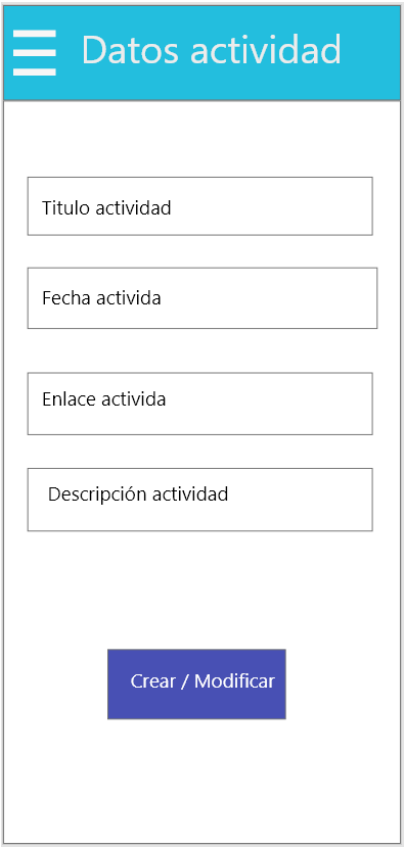

Fig. 63 Ventana de crear nueva noticia/ detales de la actividad

En el caso de la lista de publicaciones nos encontraremos ante la misma estructura que la anterior, en los detalles de cada publicación nos encontraremos los campos para cada atributo de una publicación que son título, fecha, descripción y enlace de descarga. Podemos ver el diseño de la lista de publicaciones en la figura 64 y los detalles de una publicación en la figura 65.

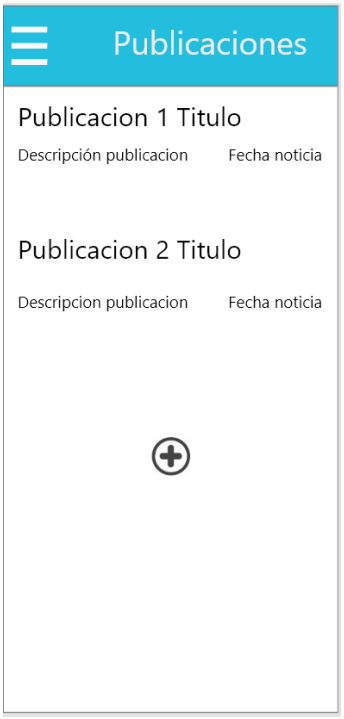

Fig. 64 Ventana de publicaciones de la aplicación

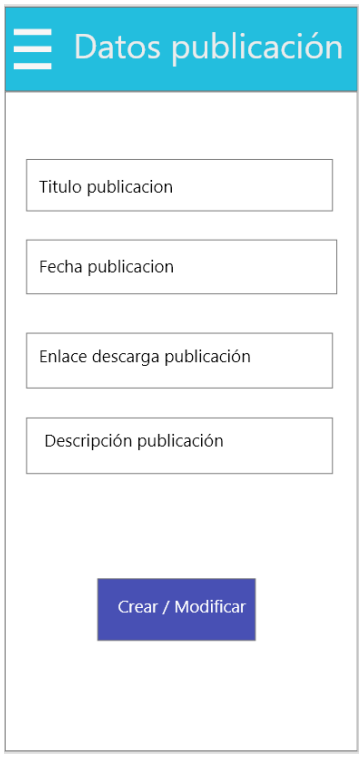

Fig. 65 Ventana de detalles publicación/ crear nueva publicación

En el caso de la newsletter dado que dicha tabla lo único que almacena son los emails de las personas que se han suscrito no tiene sentido mostrar el contenido, lo único de lo que constara la ventana es un botón que permitirá crear un nuevo email.

Las newsletters son publicaciones digitales informativas que se distribuyen a través del correo electrónico con cierta periodicidad, el botón tiene como objetivo el proceso de automatizar el proceso de escoger las nuevas noticias, actividades y publicaciones que se han creado desde que se envió la última newsletter. Podemos observar el diseño de la ventana en la figura 66.

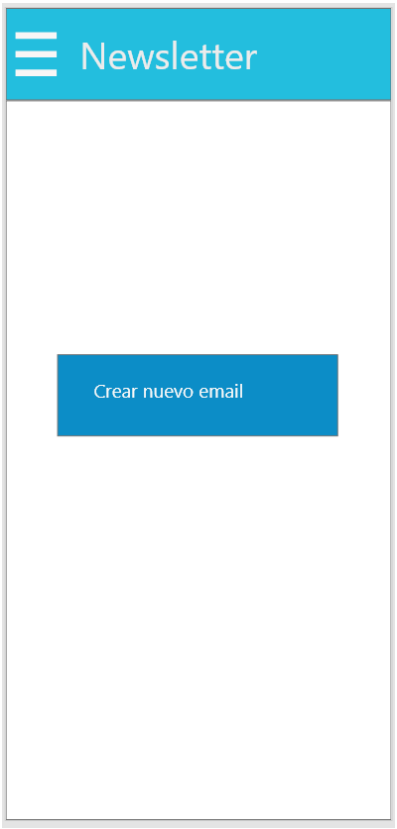

Fig. 66 Ventana de la newletter de la aplicación

Para los mensajes se creará una ventana en la que se muestre la lista de mensajes que existen en el sistema, cada uno representado por el nombre y las primeras palabras del contenido del mensaje, podemos ver el diseño en la figura 67.

En caso de que pulsemos sobre algún mensaje nos redireccionará a una nueva ventana donde podremos ver el contenido completo del mensaje y un campo de texto donde el director podrá redactar la respuesta, el botón de responder enviará un email al autor del mensaje en el que el autor será la cátedra. Podemos observar el diseño en la figura 68.

Fig. 67 Ventana de la lista de mensajes de la aplicación

Fig. 68 Ventana de responder mensaje de la aplicación

3.2.2 Implementación de la aplicación móvil

En este apartado se hablará del proceso a través del cual se ha podido crear la aplicación Android.

En primer lugar, se comentan las herramientas que se han utilizado durante este proceso, dado que Android es un sistema operativo que está desarrollado por Google utilizaremos una de las plataformas desarrollada por ellos para el desarrollo de aplicaciones móvil, en concreto Android Studio, el cual nos provee con una interfaz gráfica en la cual desarrollar nuestros diseños y además cuenta con un emulador que nos permite comprobar el funcionamiento real que tendría la aplicación que se está desarrollando en móviles.

En la figura 69 se puede ver el aspecto de la aplicación cuando abrimos la ventana de diseño, a la izquierda podemos ver las carpetas de las que se encuentra formado nuestro proyecto, y más a la derecha los diferentes componentes entre los cuales podemos elegir para agregar a nuestra ventana.

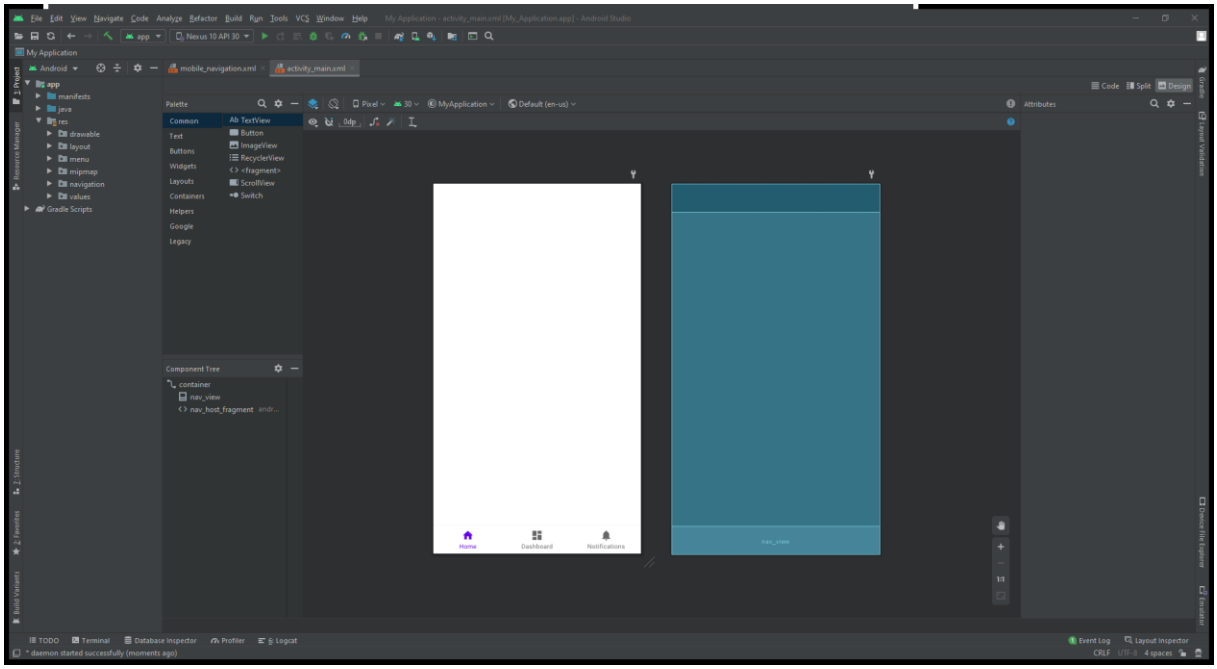

Fig. 69 Interfaz de Android Studio

Para desarrollar en Android Studio podemos escoger entre dos lenguajes, Java o Kotlin, dado que tanto durante “Introducción a la Programación” como en “Metodologías de la Programación” se ha estudiado y trabajado con Java, será este el lenguaje con el cual desarrollaremos la aplicación.

A continuación, se hace una breve explicación de cómo se trabaja con este programa, en primer lugar, tenemos los diseños, también denominados layouts, que se implementan en XML, aunque como se ha mencionado anteriormente se puede hacer uso de una interfaz gráfica que convierte los diseños en código, que se trata de un lenguaje de marcado de propósito general similar a HTML que no está predefinido por lo cual se deber definir las etiquetas a utilizar.

Sin embargo, para su uso como lenguaje de modelado, Android Studio nos define una serie de etiquetas que corresponden a los diferentes componentes que podemos añadir al diseño de nuestra interfaz como pueden ser botones o campos de texto.

Los diseños se implementan en Layouts [21] que son conjuntos de contenedores donde se pueden colocar los elementos según el diseño de nuestra aplicación, para poder referirnos después en el código a ellos, cada elemento contará con un id especifico.

Existen varios tipos de Layouts diferentes entre los que destacaremos el Constraint Layout que nos permite trabajar sin grupos de vistas anidadas creando un orden jerárquico dinámico para los elementos de los que consta nuestra vista, agregando elementos adicionales y dándole restricciones para que independientemente de la pantalla se conserve el lugar de los elementos.

Otro de los más importantes es el Frame Layout que alinea todos los elementos del diseño al lado izquierdo, aunque mediante el uso de márgenes podemos lograr que se muestren en un punto específico de la pantalla.

El último que veremos será el Linear Layout que alinea los elementos que contiene de forma vertical u horizontal dependiendo de la orientación que definamos, se suele utilizar para elementos que deban estar centrados en su totalidad.

Para que una ventana de la aplicación haga uso de uno de los diseños que se han creado habrá que crear una clase en la que se importara el diseño realizado anteriormente, en la clase se podrá definir el comportamiento de la ventana, para ello habrá que inicializar los componentes que forman parte del diseño a través de la id que previamente les hemos asignado y añadirle funcionalidades como, por ejemplo, que cuando un botón sea pulsado abra una nueva ventana. Las clases que pueden hacer uso de los diseños tienen que implementar una superclase, que puede ser o Fragment creando un fragmento o Activity dando lugar a una actividad.

Dado que se ha mencionado tanto las actividades como los fragmentos cabe mencionar cuales son las diferencias que existen entre ellos [22], mientras que las actividades son las pantallas únicas de las que está compuesta una aplicación, los fragmentos son piezas reusables de la interfaz de usuario que múltiples actividades pueden utilizar si así lo desean.

El enfoque que se utilizó en la aplicación fue el de crear una única actividad, la que ya viene definida por defecto en el proyecto de Android Studio y que está compuesta por el toolbar con el cual podemos abrir la barra de navegación lateral.

El resto de la ventana de la actividad principal estará constituida por un elemento que denominaremos content_main y que podemos apreciar en la figura 70, mientras que la barra azul constituirá el toolbar, el resto de la vista que tiene fondo blanco formará parte de este elemento.

La figura 70 presenta un aspecto muy similar al que tendrá nuestra ventana de inicio, a la cual solo le falta el botón de abrir la barra y el nombre de la aplicación. Lo que haremos en este desarrollo es crear una serie de fragmentos, uno para cada ventana distinta de la cual está compuesta la aplicación y según el usuario interactúe con la barra de navegación cargaremos un fragmento diferente que se situará en el espacio ocupado por el elemento content_main.

De esta forma una vez el usuario entre en la aplicación se encontrará ante una ventana de inicio en la cual especificaremos un mensaje de bienvenida, cuando el usuario abra la navegación y por ejemplo pulse la ventana de noticias la aplicación sustituirá el elemento content_main por el fragmento que corresponde a la lista de noticias, y cuando pulse sobre alguna de las noticias sustituirá este fragmento por el correspondiente a los detalles de una noticia.

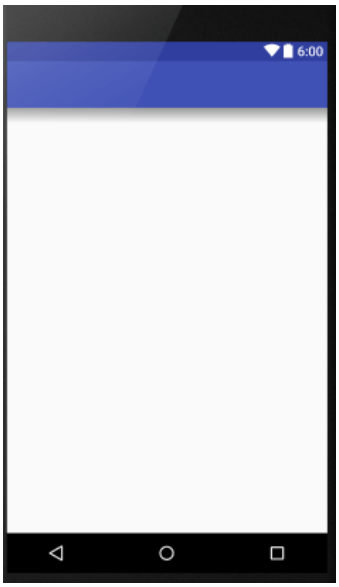

Fig. 70 Ventana de una aplicación formada por un toolbar

Para empezar con la implementación de la aplicación, comenzaremos con la creación de la barra de navegación, para ello se creará cada uno de los elementos individuales de los que se encuentra formada una barra de navegación. Como podemos observar en la figura 71 una barra de navegación está formada por un header o encabezado opcional en el que se puede presentar información sobre la aplicación o sobre el usuario como la cuenta con la que ha iniciado sesión, después nos encontramos ante el menú que está formado por los diferentes elementos que nos permiten navegar entre las diferentes vistas. En la imagen se puede apreciar también como se pueden crear espaciadores que permiten agrupar los elementos y ponerle un nombre al menú.

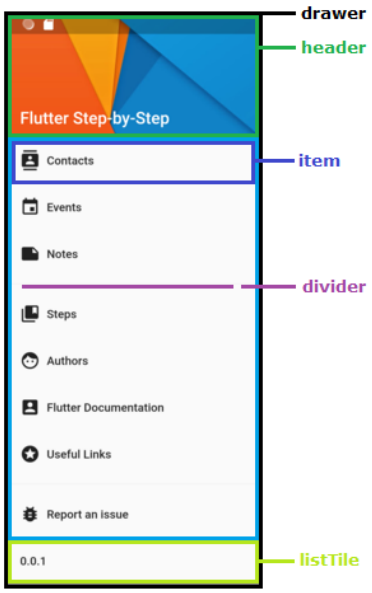

Fig. 71 Estructura de una barra de navegación

En primer lugar, se comenzará con el header en el cual simplemente pondremos el nombre de la aplicación, para ello se creó un nuevo diseño en el que se especifica un TextView que ocupa la totalidad de la anchura del elemento padre y que tendrá como altura la de una barra de navegación, medida establecida por el programa por defecto.

También crearemos el menú para lo cual usaremos el layout reservado para esta función que se denomina “menu” y que está formado por ítems, que a la vez pueden ser otros menus para crear agrupaciones de elementos como se ha podido apreciar en la figura anterior. En este caso crearemos un menú constituido únicamente por ítems, uno para cada vista (noticias, publicaciones…) y para la ventana de inicio. A cada ítem se le asignara un id único, un icono para representar su función y el nombre del mismo.

También crearemos un toolbar que es la parte de arriba donde se encontrará el botón para abrir la barra de navegación y que también contendrá el nombre de la aplicación.

Para poder crear el contenedor en el cual poder intercambiar los fragmentos se crea un nuevo diseño denominado content_main en el que especificamos un FrameLayout cuyo id sea container, siendo este el elemento que se irá intercambiando para crear la navegación.

Una vez tenemos especificados todos los elementos que forman la barra de navegación podremos dirigirnos al diseño de la actividad principal y en el que incluiremos el toolbar que hemos definido y nuestro content_main. También especificaremos un NavigationView en el cual se podrá definir que tanto el menú como el header de la barra de navegación sean los diseños personalizados que hemos creado previamente.

También se modifica el diseño de la actividad principal cuyo Layout por defecto es un Constraint Layout y lo modificaremos a un Drawer Layout el cual nos servirá para añadir la funcionalidad a nuestra barra de navegación lateral. Podemos ver el código del diseño en la figura 72.

En esta figura podemos apreciar que importamos el toolbar que hemos diseñado y el content_main que servirá para determinar el espacio que ocuparán los diferentes fragmentos, también se observa como el NavigationView tiene definido como menú y header los componentes personalizados que se han creado.

En el archivo Java de la actividad principal se declararán las variables para cada uno de los componentes que forman nuestro diseño y a los cuales pretendemos modificar su comportamiento, que serán el Drawer Layout, el toolbar y la NavigationView, después los inicializaremos con los componentes del diseño buscándolos a través de su id con la función findViewByID, para crear la acción de abrir y cerrar la navegación lateral haremos uso del método ActionBarDrawerToggle en el cual tendremos que especificar cuál es la actividad donde lo creamos, dado que nos encontramos dentro de una actividad nos podemos referir a ella mediante *this,* también se tendrá que especificar el Drawer Layout en el cual se sitúa y que será la variable que hemos inicializado previamente, el toolbar en el cual crearemos la acción y una expresión para describir la acción de abrir la barra de navegación y la de cerrarla, que serán “open” y “close” respectivamente.

Podemos ver el código de la clase de la actividad principal en la figura 73, donde apreciamos que tanto la inicialización de las variables como la configuración de la barra de navegación se realiza en el método onCreate, esta función es donde se inicializa la actividad y donde se llama al método setContentView donde se especifica el id del diseño (layout) que se va a utilizar y que como podemos ver en este caso corresponde al de la actividad principal que esta creado por defecto y que hemos modificado previamente añadiéndole los componentes de la navegación.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:id="@+id/drawer">

    <include
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/drawer_toolbar"/>

    <include
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/content_main"/>

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navigation_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:menu="@menu/drawer_menu"
        app:headerLayout="@layout/drawer_header"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"/>


</androidx.drawerlayout.widget.DrawerLayout>
```

Fig. 72 Archivo XML del diseño de la actividad principal

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    drawerLayout = findViewById(R.id.drawer);
    navigationView = findViewById(R.id.navigation_view);
    navigationView.setNavigationItemSelectedListener(this);
    actionBarDrawerToggle = new ActionBarDrawerToggle( activity: this, drawerLayout, toolbar, "open", "close");
    drawerLayout.addDrawerListener(actionBarDrawerToggle);
    actionBarDrawerToggle.setDrawerIndicatorEnabled(true);
    actionBarDrawerToggle.syncState();


}
```

Fig. 73 Método onCreate de la actividad principal

Si ejecutamos la aplicación podemos ver el resultado en un entorno real, en la figura 74 podemos ver cuál es la ventana de inicio de la aplicación que en este caso no muestra ningún mensaje y en la 75 como se muestra la barra de navegación lateral cuando se pulsa en el botón.

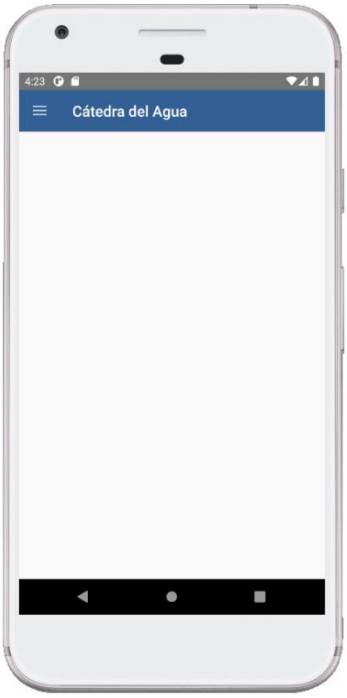

Fig. 74 Ventana de inicio de la aplicación

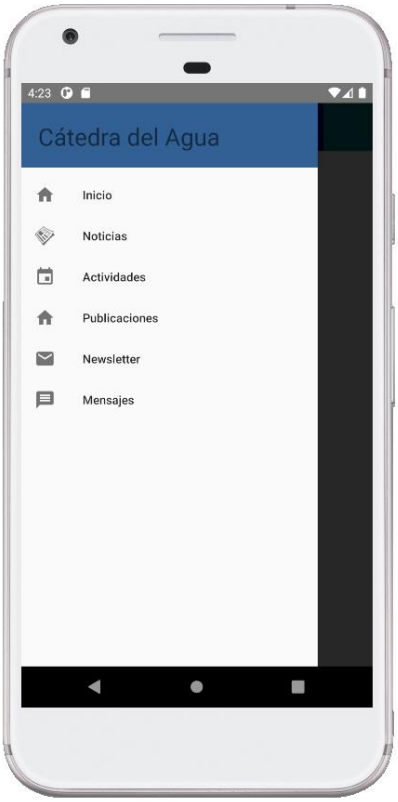

Fig. 75 Barra de navegación lateral de la aplicación

Una vez se tiene creada la barra de navegación podemos comenzar con la creación de los diferentes fragmentos, en primer lugar, empezaremos por crear las listas de elementos como pueden ser noticias o publicaciones, para ello tendremos que crear un componente personalizado para cada elemento (en este caso se hará referencia al componente de las noticias que está formado por el título, fecha, imagen y enlace) y crear una lista de este componente, para lo que nos ayudaremos de RecyclerView el cual nos ayudará a crear una lista en la que cada elemento es un componente personalizado.

RecyclerView nos permite mostrar de manera eficiente grandes conjuntos de datos, se define el aspecto de cada elemento, se proporcionan los datos y la biblioteca RecyclerView se encargará de crear los elementos de forma dinámica cuando se necesite.

En primer lugar, se crea la vista donde se mostrará la lista de noticias de nuestra base de datos, crearemos una sección a través de un Linear Layout horizontal el cual estará compuesto por dos botones y un TextView, el TextView representará el título de la lista, un botón servirá para agregar nuevas noticias y el otro botón tendrá como objetivo sugerir noticias cuya funcionalidad se estudiará en el Complemento del Trabajo de Fin de Grado. Debajo de este layout crearemos la lista del RecylerView donde se mostrarán las noticias. Podemos observar el diseño en la figura 76 en el cual se muestra tanto el resultado como el código XML con el cual se ha logrado.

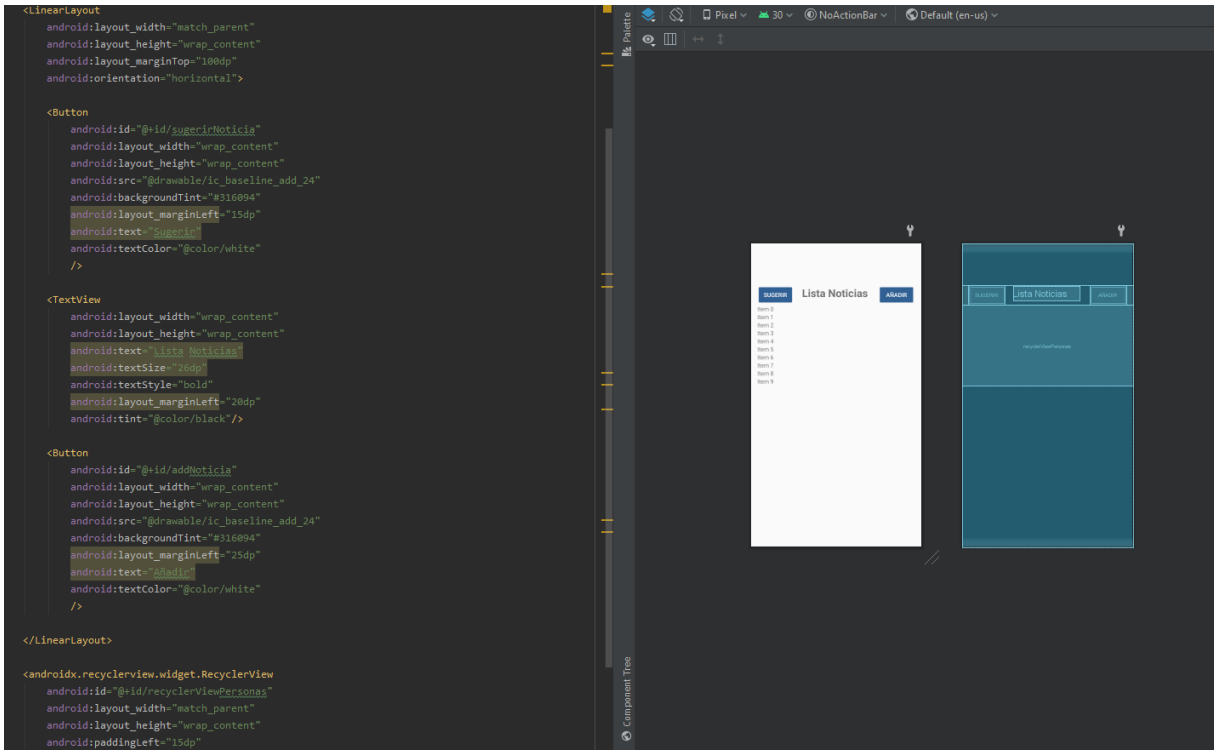

Fig. 76 Diseño y código de la lista de noticias

Una vez hemos creado la vista donde se mostrarán las noticias podemos definir el componente personalizado que representará una noticia, para ello haremos uso del widget Cardview que nos permitirá definir de forma fácil y sencilla un componente en el cual mostrar información.

Dentro de este CardView se crea un Relative Layout en el cual se especifican los elementos de los cuales se compone una noticia, que en este caso serán un ImageView donde mostraremos la imagen de la noticia, un TextView donde se mostrará el título, otro para la fecha y un Button que nos permitirá abrir el enlace de la noticia, este cambio respecto al diseño inicial tiene como objetivo facilitar la interacción del usuario.

En la figura 75 podemos ver el diseño del CardView de la noticia en donde la imagen de la noticia esta sustituida por la que se muestra por defecto cuando no se ha especificado ninguna en el diseño.

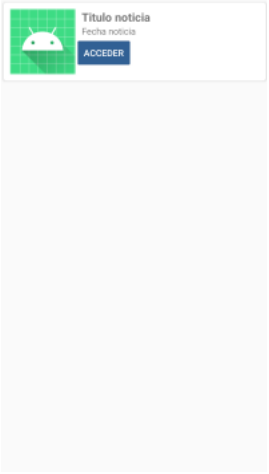

Fig. 77 Diseño de un componente noticia

Una vez hemos creado tanto el diseño de la lista de noticias como el componente de una noticia empezaremos con la parte de codificación, para ello empezaremos con crear una entidad Noticia en la cual se definan los datos de los que está compuesta una noticia, para ello crearemos una nueva clase Java denominada Noticia que tendrá como parámetros 4 strings que representarán el título, la fecha, el enlace de la imagen y el de la noticia, aparte de un id que será un Integer.

También se especifica una clase adaptador para las noticias, un adaptador es un objeto que actúa como puente entre un AdapterView y los datos de esa vista, de esta forma podremos definir como se muestran los datos del componente noticia.

Esta clase contará con los métodos onBindViewHolder y onCreateViewHolder e implementará la clase adaptador del RecyclerView que tendrá como subclase un ViewHolder personalizado que crearemos. Un ViewHolder describe la vista de un elemento y los metadatos del lugar que ocupa en el RecyclerView, esta clase tendrá como parámetros los componentes que forman una noticia.

La clase adaptador tendrá como parámetro un ArrayList de noticias que representará las noticias que se han obtenido de la base de datos, también tendremos un LayoutInflater que nos permite tener instancias de los archivos de diseños XML para crear vistas.

Se crea un constructor en el Adaptador que tendrá por parámetros el array de noticias que se mostrará y un contexto que usaremos para obtener el LayoutInflater.

Al método onBindViewHolder que se encarga de mostrar los datos de la posición especifica que se le pasa por parámetro le asignaremos a los atributos del ViewHolder que representan los elementos de un componente noticia, los valores de la noticia que se encuentre en la posición del array indicada por parámetro, por ejemplo, al componente TextView del título de la noticia cuya posición sea la primera le asignaremos como texto el título de la noticia que se encuentre en esa posición en el array.

En este paso también definiremos una variable ClickOnListener que se podrá definir más adelante y que nos permitirá que cuando se haga clic sobre la noticia lleve a una nueva ventana que muestre sus detalles.

Una vez creado el adaptador podemos crear la clase Java que corresponda al fragmento de la vista de la lista de noticias, para ello se especifica una nueva clase y se hace que herede de Fragmento lo cual nos hará implementar el método onCreateView que es el que se ejecuta por defecto, esta clase tendrá como variables el adaptador que hemos creado antes y un RecylerView.

En el método onCreateView asociaremos el fragmento al diseño de la lista de noticias, inicializaremos el RecyclerView a partir de su id, se declarará un array de noticias y se llamará a un método para mostrar los datos.

Dicho método que podemos apreciar en la figura 78 designará el Layout Manager que utilizara el RecyclerView, inicializará el adaptador con el array de noticias y se lo asignara al RecyclerView, de esta forma ya podemos mostrar todos los datos que queramos. También se puede ver como se inicializa el ClickOnListener del adaptador que de momento no tendrá ninguna funcionalidad.

```
private void mostrarDatos() {
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    adaptadorNoticia = new AdaptadorNoticia(getContext(),noticias);
    recyclerView.setAdapter(adaptadorNoticia);
    adaptadorNoticia.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

        }
    });
}
```

Fig. 78 Método encargado de mostrar las noticias

Sin embargo, aunque se pueda mostrar las noticias que se inicializan de forma local nos encontramos ante dos problemas, el primero es como obtener las noticias de nuestra API y la segunda como mostrar las imágenes, puesto que en Android Studio para mostrar imágenes o iconos como los que se muestran en la barra de navegación los tenemos que importar primero en una carpeta denominada Drawable para más tarde poder referenciarlos, sin embargo como las imágenes no estarán alojadas dentro de la aplicación se tendrá que buscar una forma de mostrarlas a partir del enlace que nos devuelve la API.

Para poder mostrar las imágenes lo que hacemos es importar librerías externas, en este caso se hace uso de la librería Glide [23] que se trata de un framework que se encarga de la gestión de imágenes, videos o GIFs animados, para poder hacer uso de ella nos tendremos que ir al "Build Gradle" de nuestro proyecto y añadirlo como dependencia de nuestra aplicación.

Esta librería cuenta con un método a partir del cual con el contexto y el enlace de la imagen podemos mostrarla en el componente ImageView que se le pase por parámetro.

De esta forma se soluciona uno de los dos problemas, respecto al otro para la extracción de los datos de nuestra API se hará uso de otra librería, en este caso la que recomienda Google para ello, que es Volley la cual también tendremos que añadir como dependencia, dado que para recuperar los datos de la API tenemos que hacer uso de internet, nos tendremos que ir al manifiesto de la aplicación y especificar como permisos de usuario el permiso a Internet.

Volley es una biblioteca HTTP que facilita y agiliza el uso de redes en apps para Android que destaca por sus operaciones de tipo RPC que se usan para completar la IU, por ejemplo, obtener una página de resultados. Se integra fácilmente con cualquier protocolo y, además, incluye compatibilidad con strings sin procesar, imágenes y JSON.

Volley nos permite hacer peticiones a través de diferentes métodos, dado que la API nos devolverá una lista de elementos como pueden ser noticias utilizaremos el método JsonArrayRequest el cual como su nombre indica nos devuelve un array de JSONs, en esta función se tendrá que especificar el método con el cual realizamos la petición, dado que en este caso se pretende extraer las noticias y no modificar ningún dato será GET, habrá que especificar la dirección en la cual realizaremos la petición, también se nos permite incluir un objeto JSON como cuerpo de la petición pero dado que no nos interesa de momento, en este caso lo dejaremos como nulo. También tendremos que especificar dos tipos de respuestas, una cuando la consulta se haya ejecutado de manera satisfactoria y otra cuando haya ocurrido un error.

En el método onResponse que nos indica el escenario exitoso se especifica como parámetro el array JSON que se ha obtenido como respuesta de la petición, lo que haremos será crear un bucle que recorra todos los objetos JSON que componen dicho array y en cada iteración inicializar una nueva noticia y asignar a sus atributos los valores que obtengamos del objeto JSON de esa iteración, para obtener dichos valores simplemente tendremos que especificar el tipo de valor que queremos obtener del objeto, en caso de que sea string utilizar el método getString y pasar por parámetro el nombre del parámetro del objeto JSON. Se puede observar dicho código en la figura 80.

De esta forma habremos creado una nueva noticia a partir de la información que hemos obtenido del objeto JSON, añadiremos la noticia al array de noticias, será este array lo que devuelva el método encargado de hacer la llamada a la API, sin embargo, esto nos presenta un problema y es que estamos trabajando con llamadas asíncronas dado que estamos realizando peticiones a un servicio de Internet y puede darse el caso de que el tiempo que pase entre el momento que realizamos la petición y en el que recibimos la respuesta haya sido demasiado y el método que hemos llamado para obtener la lista de noticias ya se haya ejecutado y devuelto un array vacío dado que no ha esperado a que la respuesta llegue.

En la figura 79 podemos ver cuál es la diferencia entre los procesos síncronos y asíncronos, mientras que en los síncronos el proceso que recibe la respuesta espera por ella, en los asíncronos este proceso sigue trabajando mientras que espera la respuesta lo que puede producir que esta no llegue a tiempo.

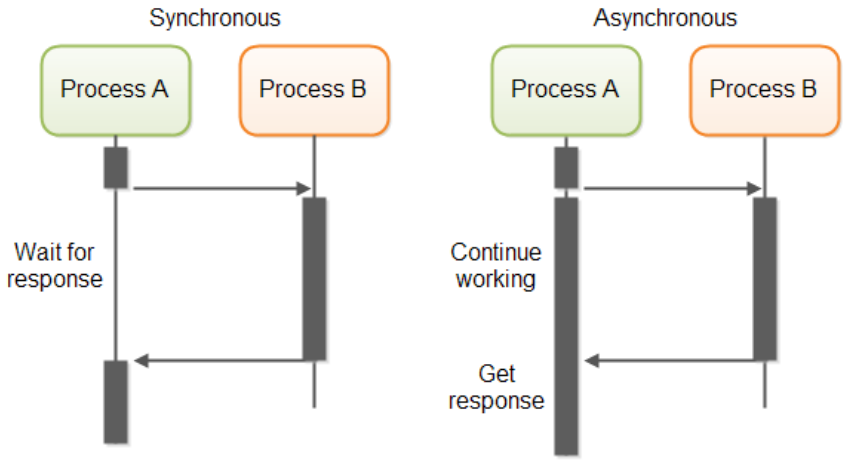

Fig. 79 Diferencia entre procesos síncronos y asíncronos

Para resolver este problema trabajaremos con callbacks [24], los callbacks son funciones que se pasan a otras funciones como argumentos que luego se invocan dentro de la función externa para completar algún tipo de rutina o acción.

Se utilizan para continuar con la ejecución de código después de que se haya completado una operación asíncrona, de esta manera si estamos por ejemplo interactuando con una API que nos devuelve unas coordenadas, la función que realiza la llamada asíncrona toma como parámetro una función callback que a su vez toma las coordenadas que queremos obtener como parámetro y esta función solo se ejecutara una vez se hayan devuelto las coordenadas.

Existen lenguajes como Javascript que prevén este tipo de situaciones y proporcionan una serie de órdenes para tratar con las llamadas asíncronas como pueden ser async o await, sin embargo, dado que se está trabajando con Java y este lenguaje no presenta soporte para los callbacks se tuvo que buscar una manera diferente de tratar con estas situaciones.

Para poder trabajar con callbacks lo que se hace es crear una interfaz en la cual se implementen dos funciones una para cuando la respuesta se haya completado de forma satisfactoria denominada onResponse y otra para cuando haya ocurrido un error denominada onError.

El método onResponse tendrá como parámetro el array de noticias que hemos obtenido de la API mientras que el onError tendrá como parámetro el string que indica el error que ha tenido lugar.

La función que hemos implementado en la cual se produce la llamada a nuestra API para obtener la lista de noticias que hasta ahora devolvía el array de noticias y no tenía ningún parámetro, ahora pasara a ser “void”, es decir, no devolverá ningún valor y tomara como parámetro una instancia de la interfaz que hemos creado anteriormente.

Ahora retomando la función en la que se obtienen las noticias, cuando tenemos el array de noticias que hemos obtenido a partir de la API llamaremos a la función onResponse de la interfaz que se ha pasado por parámetro.

En la figura 80 podemos observar cómo queda la función que realiza la llamada a la API.

```
public void cargarListaNoticias(NoticiaInterfaz noticiaInterface, Context context) {
    ArrayList<Noticia> listaNoticiasJSON = new ArrayList<>();
    String url = "https://www2.ual.es/catagua/api/noticia.php";
    JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET, url, jsonRequest: null, new Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONArray response) {
            try {
                for(int i=0; i<response.length(); i++){

                    JSONObject jsonObject = response.getJSONObject(i);
                    Noticia noticiaObject = new Noticia();
                    noticiaObject.setTituloNoticia(jsonObject.getString( name: "titulo"));
                    noticiaObject.setFechaNoticia(jsonObject.getString( name: "fecha"));
                    noticiaObject.setUrlImagen(jsonObject.getString( name: "img"));
                    noticiaObject.setUrlEnlace(jsonObject.getString( name: "enlace"));
                    noticiaObject.setId(jsonObject.getInt( name: "id"));
                    listaNoticiasJSON.add(noticiaObject);
                }
                noticiaInterface.onResponse(listaNoticiasJSON);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
        }
    });
    MySingleton.getInstance(context).addToRequestQueue(request);

}
```

Fig. 80 Función encargada de obtener la lista de noticias

En esta figura podemos observar también como hacemos uso de una clase MySingleton, dado que si una app hace un uso de la red de forma constante es probable que sea más eficiente configurar una sola instancia de RequestQueue que se extienda durante la vida útil de la misma. Cabe mencionar que cuando hemos definido por completo una petición como es en el caso del JSONArrayRequest para que dicha petición se ejecute hemos de hacer uso de RequestQueue, que es una clase definida por Volley en la cual a través del método add se añaden a una cola las peticiones que queremos que se ejecuten, si no hacemos uso de esta llamada la consulta no se completará y no obtendremos el resultado.

Para no tener que inicializar una nueva instancia de la clase RequestQueue cada vez que vayamos a realizar una petición a la API, se usa el patrón singleton, en el cual se implementa una clase singleton que encapsule la RequestQueue y que cada vez que queramos hacer uso de ella lo que hará será buscar si existe ya una instancia de la clase declarada y hacer uso de ella y si no creará una instancia que esté vigente en el contexto de la aplicación.

Una vez hemos finalizado con la función que se encarga de devolver la lista de noticias devuelta por la API podemos comenzar a ver cómo hacer uso de los resultados que devuelve, en el método onCreateView, que es el que se ejecuta por defecto cuando se crea un fragmento, llamaremos al método de obtener las noticias, para ello se tiene que crear una nueva instancia de la interfaz que hemos definido previamente, esto hará que implementemos tanto los métodos onError como onResponse.

En el caso del método onError mostraremos por consola el error que devuelve la función mientras que en el de la respuesta lo que haremos será asignarle a la variable que almacena el array de las noticias la respuesta, además también se volverá a llamar al método mostrarDatos, puesto que hasta este momento el array estaba vacío, el cual se encarga de inicializar el RecyclerView y pasarle el adaptador que contiene las noticias. Se puede ver cómo queda el método onCreateView en la figura 81.

```
@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    noticias = new ArrayList<>();
    cargarListaNoticias(new NoticiaInterfaz() {
        @Override
        public void onError(String message) { System.out.println(message); }

        @Override
        public void onResponse(ArrayList<Noticia> noticiasB) {
            noticias = noticiasB;
            mostrarDatos();
        }
    }, getContext());

    View view = inflater.inflate(R.layout.fragmento_noticias, container,  attachToRoot: false);
```

Fig. 81 Metodo onCreateView del fragmento de la lista de noticias

En la figura 82 podemos ver como luce el diseño de la ventana de noticias una vez se compila en el emulador Android, en ella visualizamos las tres noticias de las que consta la base de datos de momento y como se disponen los diferentes elementos de los que constan.

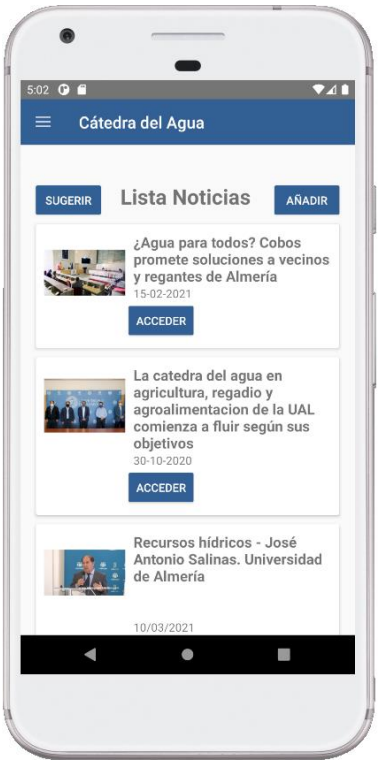

Fig. 82 Ventana de la lista de noticias compilada

En caso de que pulsemos sobre el botón acceder de alguna de las noticias se nos abrirá una nueva ventana que nos redireccionará al navegador en el cual se visualizará la noticia, como se puede observar en la figura 83.

Fig. 83 Página que muestra la noticia

Una vez se ha finalizado con la ventana principal de la sección de noticias empezaremos con implementar la funcionalidad que nos permitirá hacer que cuando cliquemos sobre una noticia nos abra una nueva ventana [25] en la cual se muestren los datos de la noticia sobre la que hemos pulsado y que nos permita modificarlos o en su defecto eliminarla de la base de datos.

En primer lugar, se crea un nuevo diseño en el cual se muestren los detalles de la noticia, como se mencionó previamente en la parte de modelado de la aplicación, este diseño se utilizará tanto para esta ventana como para la del formulario con el cual poder crear una nueva noticia.

Este diseño constara de un Linear Layout cuya orientación sea vertical, en dicho Layout creamos cuatro EditTexts en los cuales mostraremos el título, fecha, enlace de la imagen y de la noticia, también crearemos otro Linear Layout cuya orientación sea horizontal y en los cuales dispondremos los botones de modificar y de borrar. Podemos visualizar el diseño en la figura 84.

Fig. 84 Diseño de la ventana de detalles de una noticia en Android Studio

Una vez se tiene creado el diseño de la ventana podemos crear la clase Java que corresponde al fragmento de dicho diseño, para ello como siempre esta nueva clase heredará de Fragment. En esta clase crearemos las variables que corresponderán a todos los elementos de los que está formado el diseño, en el método onCreateView inicializaremos dichas variables a través de los ids y enlazaremos la clase con el diseño.

Dado que para pasar de la ventana de la lista de noticias a los detalles de una noticia tenemos que enviar a esta segunda ventana cual es la noticia con la que se está tratando, lo que haremos será crear una nueva interfaz en la cual se especificará un método enviarNoticia que tendrá por parámetro la noticia que queremos visualizar.

Para que dicho método funcione también nos dirigiremos a la clase Noticia en la cual se especifican los atributos de los que está formada y haremos que implemente la interfaz Serializable.

También se modificará el código del fragmento de la sección de noticias creando dos nuevas variables, una de tipo actividad y otra cuyo tipo sea el de la nueva interfaz que hemos creado, en esta clase implementaremos el método onAttach, en el método onAttach que recibe por parámetro un contexto, comprobaremos si este es una instancia de una actividad y en caso de que así lo sea igualaremos la variable actividad al contexto y la interfaz será igual a esta actividad.

Siguiendo en la misma clase, también sufrirá cambios el método encargado de mostrar los datos, en él implementaremos finalmente el ClickOnListener, en dicho método ejecutaremos la función de enviar noticia de la interfaz y le pasaremos por parámetro la noticia del array que hemos pulsado y que se consigue a través del método getChildAdapterPosition, se puede ver como queda dicho método en la figura 85.

```
private void mostrarDatos() {
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    adaptadorNoticia = new AdaptadorNoticia(getContext(),noticias);
    recyclerView.setAdapter(adaptadorNoticia);
    adaptadorNoticia.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            comunicaNoticias.enviarNoticia(noticias.get(recyclerView.getChildAdapterPosition(v)));
        }
    });
}
```

Fig. 85 Resultado final del método mostrarDatos

Una vez se tiene hecho todo lo anterior, nos dirigimos a la clase Java del Main Activity e implementamos la interfaz, debido a esto también tendremos que implementar la función enviarNoticia de la interfaz.

En dicha función crearemos una instancia de la clase fragmento del detalle de la noticia y un bundle, que se trata de un objeto en el cual se puede mapear valores cada uno de los cuales corresponde a una clave determinada, en este bundle será donde se adjunte la noticia, agregaremos como argumento al fragmento que hemos declarado antes dicho bundle y finalmente se realizará la transición al fragmento.

En este momento se puede aprovechar para poder explicar cómo tienen lugar esta clase de cambios, para realizarlos se hará uso de la clase que se encarga de gestionar los fragmentos, con dicho gestor obtendremos un objeto que nos permite crear transiciones, con el cual intercambiaremos el fragmento en blanco de la actividad principal denominado container por el fragmento de la ventana que se quiere visualizar.

Se utilizará este esquema tanto para realizar las transiciones entre la lista de noticias y los detalles de la noticia como para poder crear la funcionalidad de sustituir las diferentes ventanas de las que consta la aplicación mediante la interacción con la barra de navegación.

En la figura 86 se puede ver como se realiza este último proceso en el cual comprobamos el id del elemento que se ha pulsado y según esto se realizará la transición a dicha ventana.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    drawerLayout.closeDrawer(GravityCompat.START);
    if(item.getItemId() == R.id.inicio){
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, new FragmentoInicio());
        fragmentTransaction.commit();
    }
    else if(item.getItemId() == R.id.actividades){
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, new FragmentoActividades());
        fragmentTransaction.commit();
    }
    else if(item.getItemId() == R.id.noticias){
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, new FragmentoNoticias());
        fragmentTransaction.commit();
    }
```

Fig. 86 Código encargado de realizar las transiciones de la barra de navegación

Una vez se ha implementado la función de enviar una noticia desde un fragmento hasta otro, podemos irnos al fragmento de los detalles de una noticia en el cual se creará una variable bundle la cual se comprobará que no es nula, si es así significa que el objeto noticia existe y lo que se hará es obtener la noticia del bundle a partir de la clave que se ha utilizado previamente, se realizará un "casting" a dicho objeto para obtener de esta manera la noticia que se desea visualizar.

Una vez se tiene la noticia se puede rellenar los campos de texto con los valores de los atributos de dicha noticia, de esta manera ya habremos implementado la navegación entre dos fragmentos enviando datos entre ellos.

Se puede visualizar en la figura 87 como se muestran los datos de la noticia, en ella podemos ver como los datos de dicha noticia se muestran como una pista de los campos de texto para que de esta manera si el usuario empieza a modificar los datos de un campo de texto, pero quiere saber cuáles eran los que existían previamente solo tiene que borrar el contenido para poder visualizar la pista otra vez.

Fig. 87 Vista de los detalles de una noticia

Una vez se ha implementado como se muestran los datos pasaremos a implementar la funcionalidad de los botones modificar y de borrar, cada uno de los cuales estará encargado de realizar una consulta diferente a la base de datos.

Para hacer que funcione el botón de borrar lo que haremos será añadirle al componente un evento ClickOnListener que ejecute el método deleteNoticia, dicho método enviará una petición de tipo DELETE al archivo de la noticia de la API indicando el id de la noticia.

El id de la noticia se obtiene de la variable que contiene la noticia que se está visualizando, dado que en el caso anterior se pretendía obtener un array JSON con la lista de noticias pero en este caso no pretendemos obtener ningún resultado sino modificar los datos, no importa el método que utilicemos para preparar la consulta, en este caso se utiliza un JSONObjectRequest, por otra parte tanto en el caso de onResponse como en el de onError solo se mostrará por consola el objeto JSON de respuesta o el mensaje de error puesto que no hay datos con los cuales tratar.

En este caso dado que no hay que estar esperando a que se ejecute la consulta y obtener los resultados no es necesario crear una nueva interfaz para trabajar con los callbacks. En la figura 88 se puede observar el resultado final del método. Cabe mencionar que dado que se está trabajando todavía en local no se ha introducido de momento la API key para aumentar la seguridad, dado que cualquier persona que ejecute esta orden logrará borrar de la base de datos la noticia cuya id se utilice para realizar la petición.

La clave de la API que se utilizará se incorporará como argumento del JSONObject que le pasamos por parámetro a la petición.

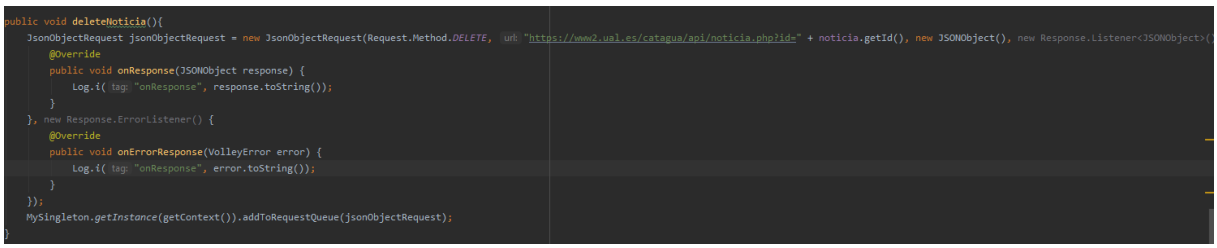

Fig. 88 Método deleteNoticia

Se seguirá el mismo proceso con el método de actualizar en el cual también utilizaremos un JSONObjectRequest en el que utilizaremos el método PUT, la dirección que utilizaremos para realizar la petición será la misma indicando el id de la noticia cuyos datos deseamos actualizar, la diferencia reside en el hecho de que se creará un HashMap que almacenara valores y claves de tipo string y en los cuales especificaremos los atributos y los valores de estos de la noticia que se va a actualizar.

Se hará uso de este HashMap para crear un objeto JSON que será lo que se envié como cuerpo de la petición. También tendremos que especificar cuáles son los header de nuestra petición. Se puede observar el método en la figura 89.

```
public void actualizarNoticia(){
    Map<String, String> params = new HashMap<>();
    params.put("titulo", titulo.getText().toString());
    params.put("fecha", fecha.getText().toString());
    params.put("img", enlaceImagen.getText().toString());
    params.put("enlace", enlaceNoticia.getText().toString());
    Log.i( tag: "JSON", params.toString());
    JsonObjectRequest jsonObject = new JsonObjectRequest(Request.Method.PUT,  url: "https://www2.ual.es/catagua/api/noticia.php?id=" + noticia.getId(),new JSONObject(params), new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            System.out.println(response.toString());
        }
    }, new Response.ErrorListener(){
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "Error", error.toString());
        }
    }){
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String,String> params = new HashMap<>();
            params.put("Content-Type","application/x-www-form-urlencoded; charset=utf-8");
            return params;
        }
    };
    MySingleton.getInstance(getContext()).addToRequestQueue(jsonObject);
}
```

Fig. 89 Método actualizarNoticia

De esta forma ya se explicado tanto la creación del fragmento que presenta la lista de noticias como seleccionar una de ellas y poder modificarla o eliminarla, solo falta cubrir la creación del formulario que nos permitirá añadir nuevas noticias a nuestra base de datos.

Para ello dado que se reciclará el diseño utilizado para ver los detalles de una noticia solo hace falta crear la clase Java que cree el nuevo fragmento, el método onCreateView de este fragmento hará lo mismo que el de detalles, es decir, se encargará de enlazar la clase al diseño de los detalles con el uso del método inflate, se declararan las variables de los componentes y se inicializarán a través de los ids, en este caso la única diferencia vendrá de que se modificara el título de los botones, en vez de modificar será crear y de borrar cancelar.

El método que se ejecute cuando se pulse el botón de crear tendrá la misma estructura que el de modificar los datos de una noticia, solo que ahora no hará falta especificar el id de la noticia dado que no se está tratando de modificar o eliminar datos y que en vez de usar el método PUT para realizar la petición se utilizara el método POST.

Se puede ver el resultado del método en la figura 90.

```
public void crearActividad(){
    Map<String, String> params = new HashMap<>();
    params.put("titulo", titulo.getText().toString());
    params.put("fecha", fecha.getText().toString());
    params.put("descripcion", descripcion.getText().toString());
    params.put("enlace_pdf", enlace_pdf.getText().toString());
    params.put("programa", programa.getText().toString());
    JsonObjectRequest jsonObject = new JsonObjectRequest(Request.Method.POST,  url: "https://www2.ual.es/catagua/api/actividad.php",new JSONObject(params), new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            Log.i( tag: "Respuesta", response.toString());
        }
    }, new Response.ErrorListener(){
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "Error", error.toString());
        }
    }){
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String,String> params = new HashMap<>();
            params.put("Content-Type","application/x-www-form-urlencoded; charset=utf-8");
            params.put("User-agent", "My useragent");
            return params;
        }
    };
    MySingleton.getInstance(getContext()).addToRequestQueue(jsonObject);
}
```

Fig. 90 Resultado del método crearActividad

Con esto se ha concluido con la creación de los fragmentos relacionados con la gestión de las noticias de nuestra base de datos, seguiremos la misma estructura tanto para las publicaciones como para las actividades, es decir, crear los diseños, tanto de la lista de elementos como el de un elemento del RecyclerView, crear la entidad, el adaptador, el fragmento de la lista de elementos, añadirle la navegabilidad a través de la creación de la interfaz e implementar los métodos que llamen a la API, que se crearán de la misma forma únicamente cambiando las direcciones a las cuales se dirigen las peticiones y los parámetros de las mismas.

De esta forma se obtendrán unas imágenes como las de la figura 91 en la cual se aprecia la lista de actividades, la 92 en la cual se muestra los detalles de una de las actividades, la 93 que hace lo propio para la lista de publicaciones y la 94 que muestra los detalles de una publicación.

Fig. 91 Lista de actividades

Fig. 92 Detalles de una actividad

Fig. 93 Lista de publicaciones

Fig. 94 Detalles de una publicación

Con esto se puede dar por concluida la descripción de cómo se realiza la creación de todos los fragmentos relacionados con la gestión de las noticias, publicaciones y actividades, con lo cual solo falta desarrollar los relacionados con la newsletter y los mensajes.

En el caso de los mensajes se seguirán los mismos pasos que se han seguido anteriormente, en primer lugar, se creará el diseño de la lista de mensajes, el de un componente mensaje que estará formado por el nombre del autor y el contenido del mensaje. Después se implementará el RecyclerView y la navegabilidad entre los dos fragmentos.

Con todo esto tendremos una lista de mensajes como la que se puede apreciar en la figura 95.

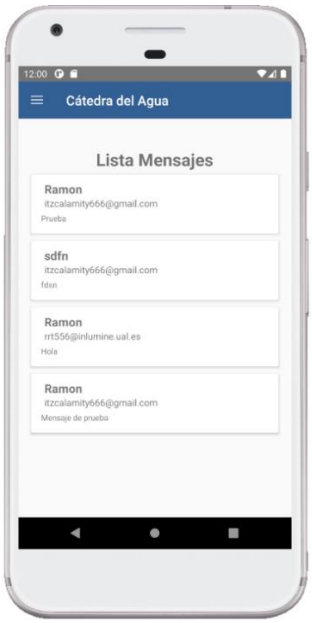

Fig. 95 Lista de mensajes

Una vez tenemos implementada la lista de mensajes, se puede comenzar con la ventana de los detalles de un mensaje, en esta ventana como ha pedido el director de la Cátedra se podrá responder al mensaje que se está visualizando, para ello se creará un Linear Layout con dos TextView en los cuales se muestre el nombre del autor y el contenido del mensaje, también tendremos un EditText donde el usuario podrá ir redactando la respuesta sin perder de vista el contenido del mensaje, también contará con otro Linear Layout cuya orientación sea horizontal y en el cual se muestre un botón para mandar la respuesta y otro para que cuando se haya respondido se pueda eliminar dicho mensaje. En la figura 96 se puede apreciar el diseño.

Fig. 96 Diseño de respuesta de un mensaje

La clase Java que se crea para dar vida al fragmento inicializará todos los componentes de dicho diseño e implementará un onClickListener para cada botón, en el del botón responder se llamará a la función responder y en el de borrar a borrarMensaje.

La función responder creará un array de strings que representarán a los destinatarios de la respuesta, dado que en este caso solo habrá uno, el email que ha dejado el usuario que ha escrito el mensaje, le introduciremos dicho email de contacto, siendo este el único elemento del que estará compuesto el array.

Se creará un "intent" que será del tipo ACTION_SEND, un intent es una descripción abstracta de una operación que tendrá lugar que permite lanzar nuevas actividades o enviar datos, este intent servirá para enviar información a terceros y en él se especificará la serie de parámetros de la acción a través de unos atributos denominados EXTRA, por ejemplo el destinatario se especifica a través de EXTRA_EMAIL , este atributo solo acepta objetos cuyo formatos sean el de un array de strings que fue la razón por la cual se creó el array en primer lugar, el asunto del email se especificará a través de EXTRA_SUBJECT y estará establecido por defecto, el contenido del email a través de EXTRA_TEXT y corresponderá al texto que se ha introducido en el EditText.

También indicaremos el tipo de dato MIME que en este caso será el RFC 822 [26] que se trata de una especificación que define el formato de un mensaje electrónico y que consiste en los header y el cuerpo del mensaje opcional.

Finalmente se creará una nueva actividad la cual permitirá al usuario elegir la aplicación a través de la cual enviar los datos mediante el método createChooser. En la figura 97 podemos ver el método encargado de crear el mensaje.

```
public void enviarMensaje(){
    String[] destino = new String[1];
    destino[0] = mensaje.getEmail();
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.putExtra(Intent.EXTRA_EMAIL, destino);
    intent.putExtra(Intent.EXTRA_SUBJECT,  value: "Respuesta de la Cátedra del Agua");
    intent.putExtra(Intent.EXTRA_TEXT, respuesta.getText().toString());
    intent.setType("message/rfc822");
    startActivity(Intent.createChooser(intent,  title: "Elija un cliente de Email"));
}
```

Fig. 97 Metodo enviarMensaje

En la figura 98 se puede ver cómo se puede redactar una respuesta, en la 99 lo que ocurre cuando se le da al botón de responder y que corresponde a la función createChooser y en la 100 lo que ocurre si elegimos Gmail en el cual se nos aparece ya rellenados los campos del destinatario, asunto y el contenido del mensaje con lo cual solo hace falta enviar la respuesta.

Fig. 98 Redactar respuesta a un mensaje

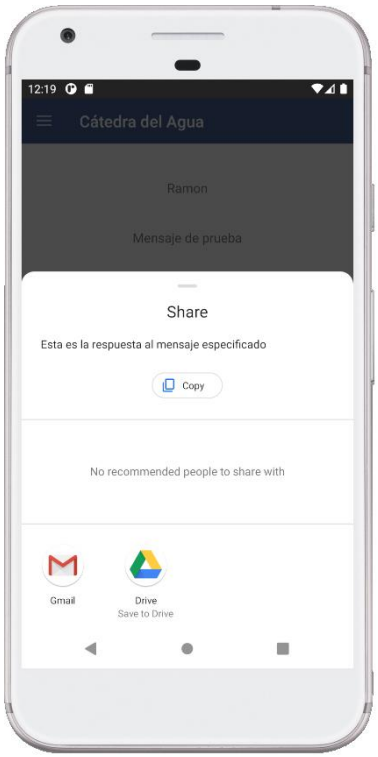

Fig. 99 Resultado de pulsar el botón responder

Fig. 100 Resultado de elegir la aplicación Gmail

Con esto se ha finalizado la parte relativa a los mensajes y solo falta la de la newsletter cuyo fragmento está compuesto solo por un botón que creará una petición del tipo POST que no insertará ningún dato, sino que tendrá como objetivo activar una función que enviará una nueva newsletter a todos los emails que se han suscrito y que se encargará de ejecutar la parte del servidor web. En la petición se especificará una API key que asegurará que dicha función sea segura y solo pueda ser ejecutada por la aplicación.

En la figura 101 se puede ver la función que se ejecuta cuando se pulsa el botón y que se encarga de activar la función de enviar la newsletter a los suscritos y en la 102 como luce el fragmento.

```
public void crearNewsletter(){
    Map<String, String> params = new HashMap<>();
    params.put("codigo", "0987654");
    JsonObjectRequest jsonObject = new JsonObjectRequest(Request.Method.POST, url: "https://www2.ual.es/catagua/api/newsletter.php",new JSONObject(params), new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            System.out.println(response.toString());
        }
    }, new Response.ErrorListener(){
        @Override
        public void onErrorResponse(VolleyError error) {
        }
    }){
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String,String> params = new HashMap<>();
            params.put("Content-Type","application/x-www-form-urlencoded; charset=utf-8");
            params.put("User-agent", "My useragent");
            return params;
        }
    };
    MySingleton.getInstance(getContext()).addToRequestQueue(jsonObject);
}
```

Fig. 101 Método encargado de crear la newsletter

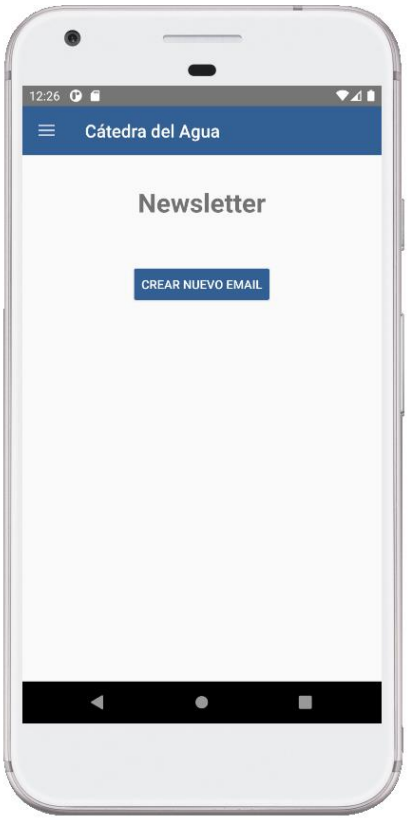

Fig. 102 Aspecto de la ventana de la newsletter

Con todo esto podemos dar por terminado el desarrollo de la aplicación dado que hemos creado una serie de funcionalidades que nos permiten realizar las operaciones básicas CURD sobre todas las tablas de las que se encuentra compuesta la base de datos.

Para crear finalmente el APK de nuestra aplicación simplemente tenemos que irnos a la sección Build APK de Build y crear una key que nos permitirá certificar que la aplicación ha sido creada por nosotros como se muestra en la figura 103.

Fig. 103 Ventana de creación de la key

3.3 Creación de la newsletter

En este apartado se habla acerca del proceso seguido en el desarrollo de la newletter, como se ha mencionado anteriormente una newsletter es una publicación digital que se distribuye a través de correo electrónico con cierta periodicidad, en este caso, lo que se hará es que cuando el director de la cátedra vea oportuno crear una newsletter pulse el botón dedicado a esta función en la aplicación y esto provocará que se ejecute una función la cual se encargará de enviar una plantilla predeterminada de email a todas las personas que se encuentren suscritas en ese momento.

Para crear esta funcionalidad se tendrán que realizar tres procesos diferentes, el primero conseguir la lista de usuarios que se han suscrito a la newsletter procurando que no haya emails repetidos que provoquen que un usuario reciba dos o más veces el mismo mensaje, crear la plantilla del email que se va a enviar y crear una función que se encargue de enviar la plantilla a todos los emails de los usuarios suscritos.

En primer lugar, se abordará la función de conseguir los usuarios suscritos a la newsletter, para ello se llamará al método que devuelve todos los emails almacenados en la tabla contacta, se creará un nuevo array al cual se irá añadiendo todos los emails que ha devuelto el método comprobando que no estén incluidos ya.

Para crear la plantilla se creará una función que devuelva un string que contenga todo el código HTML del mensaje que se va a enviar, en este código se incluirá el logo de la cátedra, una serie de noticias y enlaces para acceder a las secciones de publicaciones, noticias y actividades.

Una vez se tienen los emails, y el contenido del mensaje solo falta enviarlo, para ello se hará uso de la función mail de PHP que permite enviar correos, y en la cual se ha de especificar el destinatario, el asunto y el contenido del mensaje, aparte de headers y parámetros opcionales. Se creará una función que acepte por parámetros el remitente, el contenido y el asunto y se encargue de enviar dicho email con la información que se le provee y que se puede visualizar en la figura 104.

```php
function enviarEmail($remitente, $contenido, $asunto){
    $header = "From: Cátedra del Agua <do-not-reply@catagua.com> \r\n";
    $header .= "MIME-Version: 1.0\r\n";
    $header .= "Content-type: text/html\r\n";
    $retval = mail ($remitente,$asunto,$contenido,$header);

    if( $retval == true ) {
        echo "Message sent successfully...";
    } else {
        echo "Message could not be sent...";
    }
}
```

Fig. 104 Método enviarEmail

Con las tres funciones definidas, se implementará el código encargado de recibir la petición que active el envío, esta petición será de tipo POST y pasará por los parámetros del cuerpo de la petición un código especifico, en el archivo de la newsletter de la API se comprobarán estas condiciones y si se cumplen, se recorrerá el array de emails enviando la newsletter, pasando por parámetro a la función de enviar el email el remitente, estableciendo el asunto y recuperando la plantilla del mensaje. Podemos observar este código en la figura 105.

```php
case 'POST':
    $body = json_decode(file_get_contents('php://input'),true);
    if($body['codigo'] == '0987654'){
        $asunto = "Novedades de la Cátedra del Agua, Agricultura, Agroalimentación y Regadio";
        $array =  json_decode(getTodaLaNewsletter(),JSON_UNESCAPED_UNICODE);
        foreach($array as $i){
            $remitente = "" . $i['email'];
            enviarEmail($remitente,getMensaje(),$asunto);
        }
    }
break;
```

Fig. 105 Código encargado de comprobar la petición y enviar la newsletter

En la figura 106 y 107 se puede observar cómo se ve la newsletter.

Fig. 106 Aspecto de la newsletter

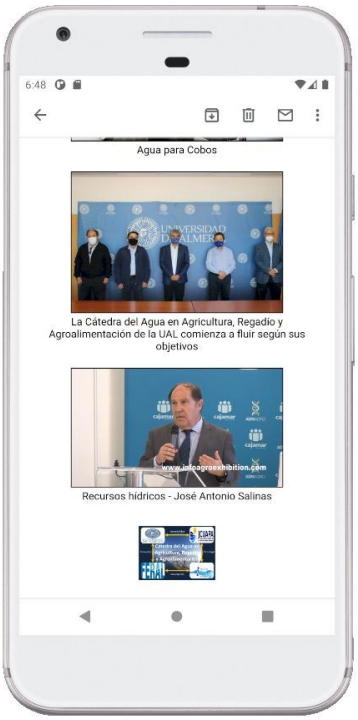

Fig. 107 Aspecto de la newsletter (2)

3.4 Publicación de la página

Una vez se ha finalizado el desarrollo de la página web era necesario intentar publicarla lo antes posible dado que el factor temporal era uno de los más importante en el desarrollo de este trabajo para poder así aumentar la visibilidad de la Cátedra del Agua.

Hasta que el STIC no nos proveyó con las claves para poder acceder tanto al phpMyAdmin de la base de datos y el usuario y contraseña del servidor FTP no se pudo comenzar el proceso de publicación de la página.

Una vez se obtuvieron las credenciales necesarias solo hacía falta subir la base de datos, dado que la base de datos local con la que se estaba de trabajando contenía la estructura y los datos necesarios que se necesitaban desde un inicio como era el caso de las noticias, con el hecho de descargar el archivo sql que contiene toda la estructura de la base de datos con la que se está trabajando e importar el archivo en el administrador de la base de datos del servidor ya crearía una réplica de la base de datos.

Para publicar los archivos correspondientes a la estructura de la página se usó el programa Filezilla cuya interfaz se muestra en la figura 108, el cual nos permite agregar servidores externos a través de su dirección y el usuario y contraseña con los que acceder a ellos, de esta forma podemos tener la estructura del directorio donde se ejecutarán los archivos que subamos y poder seleccionar estos desde un explorador de archivos que nos permite seleccionar la carpeta que queremos subir.

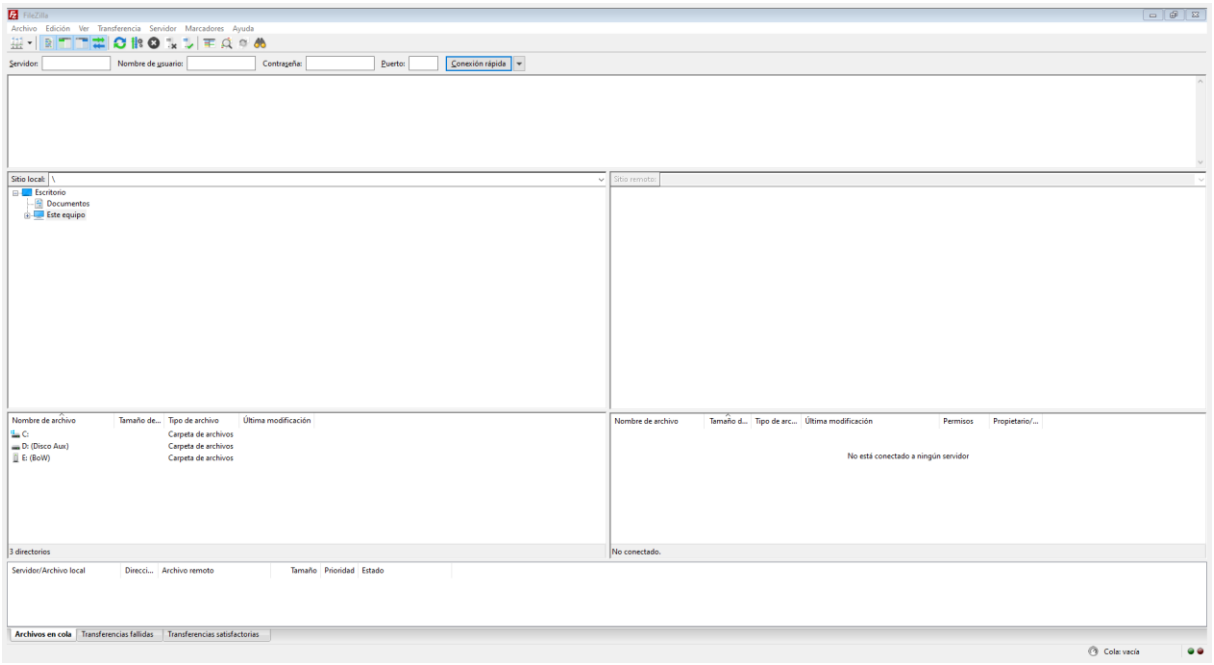

Fig. 108 Interfaz del programa Filezilla

De esta manera solo hacía falta subir los archivos del directorio en el cual estábamos trabajando al servidor FTP únicamente cambiando el archivo db.php en el cual tendremos que especificar las nuevas credenciales con las cuales podernos conectar a la base de datos.

Podemos observar en la figura 109 tanto el directorio local en el que trabajamos como el servidor al cual tenemos que subir los archivos.

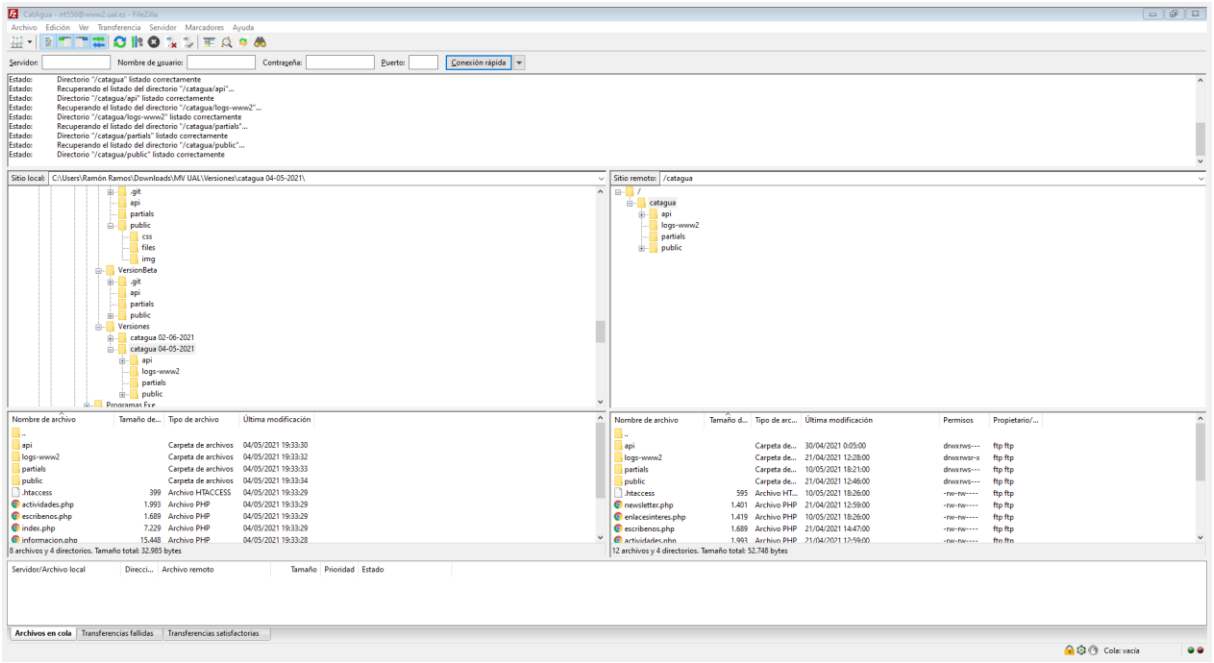

Fig. 109 Directorio local y del servidor de la página web

De esta forma podremos descargarnos también la versión que está funcionando en el servidor y crear copias de seguridad para poder así trabajar en modificaciones que mejoren el comportamiento de la página, para subir estas modificaciones simplemente podemos sustituir los archivos que han tenido cambios o volver a subir el directorio entero de nuestro proyecto.

4 Conclusiones

Como resultado final, en mi opinión el trabajo ha sido satisfactorio, puesto que se ha creado una página funcional cumpliendo con los requisitos del director, a través de varios prototipos, e intentando en todo momento acelerar el proceso para cumplir con el objetivo de tener la página publica lo antes posible. Dicho esto, el trabajo en la página no está completo, puesto que se seguirá trabajando junto a la Cátedra del Agua para incorporar nuevas características y modificaciones que permitan una mejor experiencia de usuario y satisfagan al cliente.

En este sentido, la experiencia de trabajar en un problema real, en el que se intentan satisfacer las necesidades tanto de un cliente como de unos usuarios reales me ha ayudado a entender la magnitud del trabajo y la presión con la que se trabaja, dando lugar a un resultado que va a estar activo durante un largo tiempo.

El desarrollo de la página web se ha desarrollado escuchando las indicaciones del cliente en todo momento y tomando su opinión como el criterio diferencial a la hora de implementar cambios tanto en el diseño como en la funcionalidad de la misma. Aunque las tecnologías con las cuales se ha trabajado en este aspecto no eran nuevas para mi he descubierto nuevas formas de mejorar el proceso de desarrollo, la estructura del proyecto y añadiendo modificaciones como el .htdocs que mejoran el enrutamiento de la página.

En cuanto a la creación de la aplicación móvil todo lo relativo a su desarrollo ha sido nuevo para mí, puesto que, aunque ya trabajé en el desarrollo de aplicaciones móvil en Líneas de Productos Software 1, en ese caso se trabajó con una herramienta diferente, en concreto xCode, trabajando con un lenguaje completamente diferente como es Swift y para otra plataforma, iOS.

Probablemente sea la parte de este proyecto de la que más orgulloso me siento, puesto que adicionalmente a adaptarse a una nueva forma de trabajo, se han tenido que hacer uso de servicios externos como una API lo cual añade un nivel de complejidad más, pero a pesar de ello, se ha conseguido una aplicación que se adapta perfectamente a cualquier dispositivo en el cual se ejecute sin tener que hacer un uso excesivo de los recursos y la cual es muy robusta puesto que se ha trabajado para reducir el número de factores que puedan inducir a los fallos consiguiendo así estabilidad.

La aplicación móvil seguirá también recibiendo actualizaciones que mejoren su funcionamiento y que reflejen los cambios que tengan lugar en la página web, además de posibles modificaciones como puede ser la adición de un sistema de gestión de usuarios si la cátedra sigue creciendo y se hace necesario una creación de perfiles que permitan modificar la página.

En resumen, se ha creado una página web y una aplicación móvil que a pesar de los contratiempos que se han sufrido durante el proceso de desarrollo de ambas han logrado satisfacer los objetivos que se propusieron en un primer lugar cuando se aceptó iniciar este proyecto.

Por último, también considero importante mencionar una serie de asignaturas que me han aportado una experiencia y conocimiento sobre los cuales se ha podido ir desarrollando para completar el trabajo como son Líneas de Productos Software, Tecnologías Web y Desarrollo Rápido de Aplicaciones.

5 Bibliografía

[1] MDN Web Docs. HTML: Lenguaje de etiquetas de hipertexto (2021). Disponible en: https://developer.mozilla.org/es/docs/Web/HTML

[2] Wikipedia. HTML5. Disponible en:

https://es.wikipedia.org/wiki/HTML5

[3] MDN Web Docs. CSS Básico (2021) Disponible en:

https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics

[4] Wikipedia. Hoja de estilos en cascada (2021). Disponible en:

https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

[5] Desarrollo web. Introducción a CSS3 (2008). Disponible en:

https://desarrolloweb.com/articulos/introduccion-css3.html

[6] MDN Web Docs. ¿Qué es JavaScript? (2021). Disponible en:

https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[7] PHP. ¿Qué es PHP? (2020). Disponible en:

https://www.php.net/manual/es/intro-whatis.php

[8] Staff Creativa. 12 ventajas de la programación PHP que debes saber (2014). Disponible en:

https://www.staffcreativa.pe/blog/ventajas-programacion-php/

[9] W3 Techs. Usage statistics of PHP for websites. Disponible en:

https://w3techs.com/technologies/details/pl-php

[10] Wikipedia. Bootstrap (framework). Disponible en:

https://es.wikipedia.org/wiki/Bootstrap_(framework)

[11] Wikipedia. Adobe XD (2020). Disponible en:

https://es.wikipedia.org/wiki/Adobe_XD

[12] Rajesh Bhujbal´s Blog. Mysqli_connect Vs Mysql_connect (2014). Disponible en:

https://rbhujbal.wordpress.com/2014/01/30/mysqli_connect-vs-mysql_connect/

[13] Tim Fisher. What is an HTACESS file? (2020). Disponible en:

https://www.lifewire.com/htaccess-file-2621687

[14] SoapUI. SOAP vs REST 101: Understand the differences. Disponible en:

https://www.soapui.org/learn/api/soap-vs-rest-api/

[15] MDN Web Docs. Métodos de petición HTTP. Disponible en:

https://developer.mozilla.org/es/docs/Web/HTTP/Methods

[16] MDN Web Docs. Generalidades de protocolo HTTP. Disponible en:

https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[17] Jonathan Ordóñez. ¿Qué es una API REST? Disponible en:

https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/

[18] Restful API. What is REST. Disponible en:

https://restfulapi.net

[19] Sophea Mak. How to secure the Rest APIs (2020). Disponible en:

https://medium.com/javarevisited/how-to-secure-the-rest-apis-b682e21821a1

[20] Auth0 Docs. OAuth 2.0 Authorization Framework. Disponible en:

https://auth0.com/docs/protocols/protocol-oauth2

[21] Nube Colectiva. Que son los Layouts y cuales existen en Android Studio (2018). Disponible en:

https://blog.nubecolectiva.com/que-son-los-layouts-y-cuales-existen-en-android-studio/

[22] Darren Finch. Fragments vs Activities for Android Development (2020) Disponible en:

https://darrenfinch.com/fragments-vs-activities-for-android-development-ui-architecture-guide/

[23] Bumptech. Glide (2021). Disponible en:

https://github.com/bumptech/glide

[24] MDN Web Docs. Función Callback. Disponible en:

https://developer.mozilla.org/es/docs/Glossary/Callback_function

[25] CMSDK. Pass data from RecyclerView in Fragmento to another Fragment (2018). Disponible en:

https://cmsdk.com/android/pass-data-from-recyclerview-in-fragment-to-another-fragment.html

[26] Microsoft Docs. RFC 822 Message Format (2013). Disponible en:

https://docs.microsoft.com/en-us/previous-versions/office/developer/exchange-server-2010/aa493918(v=exchg.140)