



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención de Ingeniería Software)

**Evolución y diseño de nuevas
funcionalidades en una aplicación
móvil de carácter informativo
dirigida a estudiantes universitarios**

Autor:

D. Rafael Higuelmo San Millán

Tutores:

**Dr. D. Jesús M. Vegas Hernández
Dra. Dña. Noemí Merayo Álvarez**

Resumen

Los alumnos de la Universidad de Valladolid demandan nuevas herramientas para informarse de noticias del ámbito universitario que les incumban como estudiantes. Noticias sobre prácticas, eventos, congresos, conferencias o becas de interés para alumnos universitarios, son algunas de las noticias a las que los alumnos no tenían acceso de un modo rápido e integrado. Por ello, surgió la idea de desarrollar una aplicación móvil para Android que cumpliera con ese cometido, llamada UVaNOW. Esta aplicación tiene ciertas similitudes a aplicaciones existentes en otras universidades que se analizaron en un paso previo con la intención de establecer unas pautas para el diseño de la nueva aplicación. UVaNow es una aplicación que se nutre de una página web `inform@Uva` (<http://www.informauva.com/>) administrada por alumnado de la Facultad de Periodismo de Valladolid. La versión inicial de la aplicación, aunque era totalmente funcional y operativa, tenía mucho margen de mejora en cuanto a funcionalidades, diseño y ciertos fallos detectados. Así pues, este Trabajo de Fin de Grado tiene como objetivo mejorar estas carencias, hacerla más llamativa e integrar nuevas funcionalidades para así conseguir que llegue a un mayor número de usuarios objetivo. Además, también se realizó un cambio completo del backend de la aplicación, utilizando otros lenguajes de programación menos obsoletos y permitiendo una mayor capacidad de mejorar la aplicación en futuras versiones.

Palabras clave

Android, universidad, base de datos, Wordpress, servicio web, MySQL, phpMyAdmin, Tomcat, UVaNow, Java.

Abstract

The students of the University of Valladolid demand new tools to find out about news from the university environment that concern them as students. News about internships, events, congresses, conferences or scholarships of interest to university students, are some of the news that students did not have access to in a fast and integrated way. For this reason, the idea arose to develop a mobile application for Android that would fulfill this task, called UVaNOW. This application has certain similarities to existing applications in other universities that were analyzed in a previous step with the intention of establishing guidelines for the design of the new application. UVaNow is an application that is fed by a web page inform @ Uva (<http://www.informauva.com/>) managed by students from the Faculty of Journalism of Valladolid. The initial version of the application, although it was fully functional and operational, had a lot of room for improvement in terms of functionalities, design and certain detected bugs. Thus, this Final Degree Project aims to improve these shortcomings, make it more attractive and integrate new functionalities in order to reach a greater number of target users. In addition, a complete change of the backend of the application was also made, using other less obsolete programming languages and allowing a greater ability to improve the application in future versions.

Keywords

Android, university, data base, Wordpress, web service, MySQL, phpMyAdmin, Tomcat, UVaNow, Java.

Agradecimientos

Lo primero de todo agradecer a mis tutores el seguimiento continuo, ayuda y dedicación que han tenido, además del tiempo que han empleado en mí. Es de agradecer no solo esto sino también el concederme la ocasión de realizar este Trabajo de Fin de Grado. Dar las gracias a los técnicos que nos dieron el nuevo servidor remoto que necesitábamos en el menor tiempo posible y a mis tutores también por gestionar todo eso con premura.

A las personas de periodismo con la que mantuve el contacto a lo largo del proceso y que colaboraron siempre que pudieron, también les quiero dar las gracias.

Y para finalizar, agradecer a mi familia y amigos por apoyarme siempre y estar ahí.

Tabla de contenidos

1	Introducción	11
1.1	Motivación	11
1.2	Objetivos	12
1.3	Punto de Partida de la Aplicación	13
1.4	Estructura de la memoria del TFG	15
2	Planificación	17
2.1	Metodología y Planificación	17
2.1.1	Documentación Previa y Metodología	17
2.1.2	Planificación	18
2.2	Seguimiento	20
2.3	Riesgos	22
2.4	Presupuesto	23
2.5	Herramientas software empleadas	23
3	Actualización e Implementación de Nuevas Funcionalidades en la Aplicación UVaNow	27
3.1	Introducción	27
3.2	Análisis	28
3.3	Diseño	29
3.4	Implementación	30
3.4.1	Mejoras en el buscador de la aplicación	30
3.4.2	Mejoras en el slider y contenido de noticias	31
3.4.3	Mejoras en el diseño de visualización de las noticias	34
3.5	Diseño e implementación de nuevas funcionalidades	37
3.5.1	Guardar noticias en el calendario del móvil	37

3.5.2	Compartir noticias a través de redes sociales	39
3.6	Pruebas	40
3.7	Conclusiones	42
4	Migración de la Base de Datos y el Servidor Web	43
4.1	Introducción	43
4.2	Modo de acceso al Servidor	44
4.3	Análisis	45
4.4	Diseño	46
4.5	Implementación	47
4.5.1	Instalación de JDK en el servidor	47
4.5.2	Instalación y configuración de MySQL	48
4.5.3	Instalación y configuración de phpMyAdmin	50
4.5.4	Migración y modificaciones de la Base de Datos	53
4.5.5	Copias de seguridad Base de Datos	56
4.5.6	Instalación y Configuración Apache Tomcat	58
4.6	Pruebas	61
4.7	Conclusiones	62
5	Implementación de los Servicios Web en Java	63
5.1	Introducción	63
5.2	Análisis	63
5.3	Diseño	64
5.4	Implementación	66
5.4.1	Servicio Web que conecta con la Aplicación Android	66
5.4.1.1	Herramientas y tecnologías usadas en el desarrollo del servicio web	66
5.4.1.2	Descripción del funcionamiento y métodos del servicio web	67
5.4.2	Aplicaciones Java que actualizan la Base de Datos	81

5.4.2.1	Descripción de las aplicaciones Java.....	81
5.4.3	Script de ejecución diaria de aplicación.	88
5.5	Pruebas.....	89
6	Subida al Market y Requisitos de la App.....	91
6.1	Introducción.....	91
6.2	Encuesta y opiniones antes de subir al Market	91
6.3	Subida de la aplicación al Market.....	92
6.4	Requisitos de la aplicación UVaNow.....	94
7	Conclusiones y líneas futuras	95
7.1	Conclusiones.....	95
7.2	Líneas futuras	96
8	Bibliografía	97

Índice de Figuras

Figura 1: Esquema general de la aplicación.....	14
Figura 2: Planificación del proyecto	19
Figura 3: Seguimiento del proyecto	21
Figura 4: Primera parte del diagrama de clases de la aplicación Android	29
Figura 5: Segunda parte del diagrama de clases de la aplicación Android	30
Figura 6: Buscador de noticias	31
Figura 7: Elemento WebView para mejorar el formato de las noticias	32
Figura 8: Configuraciones aplicadas a WebView	32
Figura 9: Corrección del Slider	34
Figura 10: Presentación de las noticias en la anterior versión	34
Figura 11: Aplicación de la Universidad Complutense de Madrid	35
Figura 12: Presentación de las noticias en la versión actual	36
Figura 13: Imagen del contenido de una noticia y nuevo botón para guardar noticias.....	38
Figura 14: Aplicación Calendario con los datos de la noticia guardada	39
Figura 15: Nuevos botones para compartir en Redes Sociales	40
Figura 16: Acceso a escritorio remoto con interfaz gráfica	44
Figura 17: Arquitectura del alojamiento web.....	46
Figura 18: Página web de Oracle donde descargar JDK 8u131	47
Figura 19: Lista de JDK instaladas en el servidor.....	48
Figura 20: Configuración en instalación MySQL, petición de contraseña root.....	49
Figura 21: Elección de seguridad dentro de instalación MySQL.....	50
Figura 22: Comprobación del estado del servidor apache2	51
Figura 23: Configuración contraseña phpMyAdmin	53
Figura 24: Introducción de datos en la BD por medio de Inserts.....	54

Figura 25: Esquema de la Base de Datos actual	55
Figura 26: Desglose de las tablas y columnas de la Base de Datos	56
Figura 27: Script que genera en el directorio seleccionado una copia de la Base de Datos fechada	56
Figura 28: Creación de roles y usuario para Apache Tomcat	59
Figura 29: Página principal de Apache Tomcat	60
Figura 30: Apartado donde seleccionar y desplegar aplicación en formato War.....	60
Figura 31: Servicios web empleados en la anterior versión de UVaNOW	64
Figura 32: Diagrama de los recursos del servicio web desarrollado	65
Figura 33: Esquema de funcionamiento de la aplicación Rest.....	67
Figura 34: Propiedades establecidas en el archivo Application.properties	69
Figura 35: Estructura de la aplicación y package Core	70
Figura 36: Clase Application, clase Main del proyecto Java	70
Figura 37: Paquetes Controllers y ControllersImpl	71
Figura 38: Interfaz del controlador Becas del paquete Controllers.....	71
Figura 39: Clase que implementa interfaz del controlador Becas.....	72
Figura 40: Implementación controlador del Slider, parte uno	73
Figura 41: Implementación controlador del Slider, parte dos	74
Figura 42: Paquetes Dominio, Repository, Service y ServiceImpl del esquema de paquetes	75
Figura 43: Entidad del Dominio con sus atributos	76
Figura 44: Clase Repository capa donde se integra el JPA.....	77
Figura 45: Clase interfaz Servicio	77
Figura 46: Clase BecasServiceImpl, ejemplo de implementación de servicio.....	78
Figura 47: Llamada al servicio Rest para obtener todas las noticias de una categoría	79
Figura 48: Clase SumarVisita	80
Figura 49: Implementación de la clase SumarVisita.....	80
Figura 50: Estructura de paquetes de la aplicación de actualización de la Base de Datos.....	81

Figura 51: Clase conexionDB de la aplicación Java	82
Figura 52: Clase UpdateFastBDwp.....	83
Figura 53: Primera parte del método updatePrácticas.....	84
Figura 54: Método para pasar el JSONArray a un ArrayList de Noticias	85
Figura 55: Segunda parte método updatePracticas	86
Figura 56: Introducción y actualización de noticias en el segundo programa	87
Figura 57: Script de ejecución de la aplicación Java de actualización diaria	88
Figura 58: Fichero Cron donde se incluye la línea para la ejecución diaria del Script.....	88
Figura 59: Creación de una nueva versión de la aplicación en producción	93

Índice de tablas

Tabla 1: Tabla de riesgos del proyecto.....	22
Tabla 2: Pruebas de la parte del Cliente Android.....	42
Tabla 3: Pruebas de las configuraciones e instalaciones realizadas en el servidor	61
Tabla 4: Pruebas de los servicios web implementados	90
Tabla 5: Resultados de la encuesta sobre la aplicación.....	92
Tabla 6: Características principales de la aplicación	94

1

Introducción

1.1 Motivación

El acceso a la información por parte de las personas y sobre todo de las personas jóvenes ha ido evolucionando hacia una información de acceso más inmediato y en medios digitales. Cada vez más se busca la rapidez a la hora de ser informados y casi nada es más rápido hoy en día que nuestros dispositivos móviles, que además nos acompañan la mayor parte del tiempo. Por ello es habitual que en las universidades surjan aplicaciones destinadas a informar a los alumnos de una manera más ágil que la que tenían habitualmente. La Universidad de Valladolid carece hoy en día de una aplicación de estas características, hasta que hace un tiempo surgió dicha posibilidad y fue desarrollada en un Trabajo de Fin de Grado en el que colaboraron la Facultad de Periodismo con la Escuela Técnica Superior de Ingenieros de Telecomunicaciones de la Universidad de Valladolid.

En un primer momento se realizó una encuesta previa para analizar si la idea de informar a jóvenes universitarios a través de una App móvil personalizada era aceptada, que lo fue por gran mayoría, y también para ver las categorías y/o tipos de noticias más importantes para los alumnos. La aplicación fue desarrollada, pero, pese a ser ya funcional tenía ciertas carencias, errores y falta de algunas funcionalidades interesantes que permitieran afianzar el uso de la aplicación entre el alumnado. Por ello surge este Trabajo de Fin de Grado, con el objetivo de evolucionar la aplicación móvil inicial, denominada UVaNOW, añadiéndola nuevas funcionalidades, mejorando el diseño, corrigiendo errores y estableciendo unos servicios web mejorados para dicha aplicación. Además, también se necesitaba un sistema de monitorización automatizado que aportara cierta información que permitiera determinar qué noticias o categorías son las más visitadas por los alumnos. Esta monitorización fue otro de los motivos por los cuales se desarrolló este proyecto.

1.2 Objetivos

En primer lugar, es reseñable comentar que algunos de los objetivos fueron incorporados durante el transcurso del proyecto. Los objetivos que se establecieron para este Trabajo de Fin de Grado son los siguientes:

- Desarrollar unos servicios web que ofrezcan mejores prestaciones y seguridad que los creados en primera instancia para la aplicación Android.
- Sustituir los programas escritos en PHP, que se ejecutan de manera manual y que tienen como objetivo traer la información de la página web [inform@Uva \(http://www.informauva.com/\)](http://www.informauva.com/) a la Base de Datos, sustituirnos los programas que ofrezcan la misma funcionalidad, pero en un lenguaje menos obsoleto que PHP y de que se ejecuten de manera automática.
- Diseñar e implementar un sistema de monitorización de los accesos a las noticias a través de la aplicación móvil.
- Instalar y configurar las herramientas necesarias en un nuevo servidor remoto y migrar la Base de Datos original a dicho servidor.
- Realizar un sistema para generar copias de seguridad de la Base de Datos de manera automática.
- Implementar nuevas funcionalidades, cambios y correcciones en la App, en concreto:
 - Implementar en la Aplicación una función de guardado de eventos que permita al usuario ubicarlos en el calendario.
 - Desarrollar la funcionalidad de compartir cualquier noticia de la aplicación a través de redes sociales.
 - Cambiar el formato visual con el que las noticias aparecen en la aplicación por una en la que se vean las noticias con sus imágenes.
 - Corregir problemas de la aplicación en cuestión de formato del contenido de la noticia, Slider y buscador de noticias.

1.3 Punto de Partida de la Aplicación

Este trabajo de Fin de Grado se desarrolla a partir del trabajo realizado en un diseño e implementación preliminar de una aplicación móvil, denominada UvaNow, y realizada en un Trabajo de Fin de Grado anterior.

UvaNow es una aplicación que fue creada para cubrir la necesidad de los estudiantes de la Universidad de Valladolid de tener una fuente de información accesible y cómoda desde sus propios dispositivos móviles debido a la no existencia de una aplicación de esas características para esta universidad. Esta aplicación se nutre directamente de la página web informativa informa@UVa (<http://www.informauva.com/>) [1], que es gestionada y actualizada diariamente por los alumnos de la Facultad de Periodismo de Valladolid. De esta página web se obtiene información en forma de noticias categorizadas, tales como becas, eventos, congresos entre otras, y que han sido elegidas como las de mayor relevancia para los alumnos universitarios.

Tanto la idea como el diseño de la aplicación surgieron debido a la necesidad de buscar nuevas formas de informar a ese público objetivo, formado por estudiantes universitarios, usando para ello sus dispositivos móviles. Así, partiendo del análisis y la investigación de otras aplicaciones con ciertas características similares existentes en otras universidades, se llegaron a ciertas conclusiones de diseño que se intentaron plasmar en la primera versión implementada. En este TFG, uno de los objetivos ha sido acercarse más a ese diseño inicialmente planeado, haciendo que la aplicación sea más vistosa, agradable e intuitiva y que cuente con un mayor número de funcionalidades.

La página web infroma@Uva está diseñada en Wordpress [2], que es un sistema de gestión de contenidos utilizado para crear blogs o páginas web de manera más sencilla y ágil. En el anterior trabajo realizado sobre la aplicación se llegó a la conclusión que Wordpress presenta una serie de problemas a la hora de interactuar con los datos de la plataforma con otros lenguajes de programación, como en nuestro caso Android. Para ello hacemos uso de la API (*Application Programming Interface*) propia que tiene esta plataforma, la WP API Rest [3], de la cual obtenemos los datos, en concreto las noticias de las categorías que hemos elegido y las guardamos en la base de datos intermedia alojada en el servidor que tenemos implementado. Esta API permite la conexión con Wordpress desde cualquier aplicación y desde cualquier lenguaje de programación, lo que facilita el proceso de traspaso de información a la base de datos.

La App ha sido desarrollada en Android, por ser el sistema operativo más extendido hoy en día, aunque está abierta la posibilidad de que un futuro se desarrolle para otros sistemas operativos mayoritarios como iOS. Los datos empleados por la aplicación surgen de una base de datos intermedia diseñada en MySQL [4] y alojada en un servidor que se han montado desde cero en este TFG. En este sentido, cabe indicar que en la primera versión de la App se hacía uso de un servidor que carecía de ciertos permisos y funcionalidades que necesitábamos tener para la evolución de la aplicación y por esto fue sustituido por otro servidor con mayor alcance de permisos, para instalar y establecer los servicios web y demás proyectos y funcionalidades

que necesitábamos implementar. Esa base de datos intermedia, también diseñada para nuestra App, se nutre de los datos y noticias aportadas por la página web informa@UVa a través de la WP API Rest. Esta es llamada a través de un programa que se ejecuta de manera diaria añadiendo a la base de datos las nuevas noticias de la página web en su correspondiente categoría, de modo que se actualizan los datos que se introducen en la página web de los alumnos de Periodismo en la base de datos intermedia.

Para otorgar seguridad a todo el proceso, la aplicación móvil accede a la base de datos por medio de un servicio web escrito en lenguaje Java [5] y mediante el uso del framework llamado Spring [6] y más concretamente con Spring Boot. Este servicio web está alojado en el servidor, desplegado en un Tomcat7 [7] que previamente instalamos. Además, también contamos con dos proyectos Java adicionales alojados en el servidor y encargados de llamar a la API de Wordpress anteriormente mencionada para obtener las noticias de la página web. De este modo, el primer proyecto es ejecutado de manera diaria, añadiendo las últimas noticias generadas en Wordpress a la base de datos. El segundo proyecto es un actualizador del contenido de las noticias, que sirve como respaldo para que en el improbable caso de que una noticia tenga que ser modificada, se actualice el contenido de la base de datos. Toda esta estructura de las diferentes herramientas y bloques desarrollados para la implementación de la App se muestran en la Figura 1.

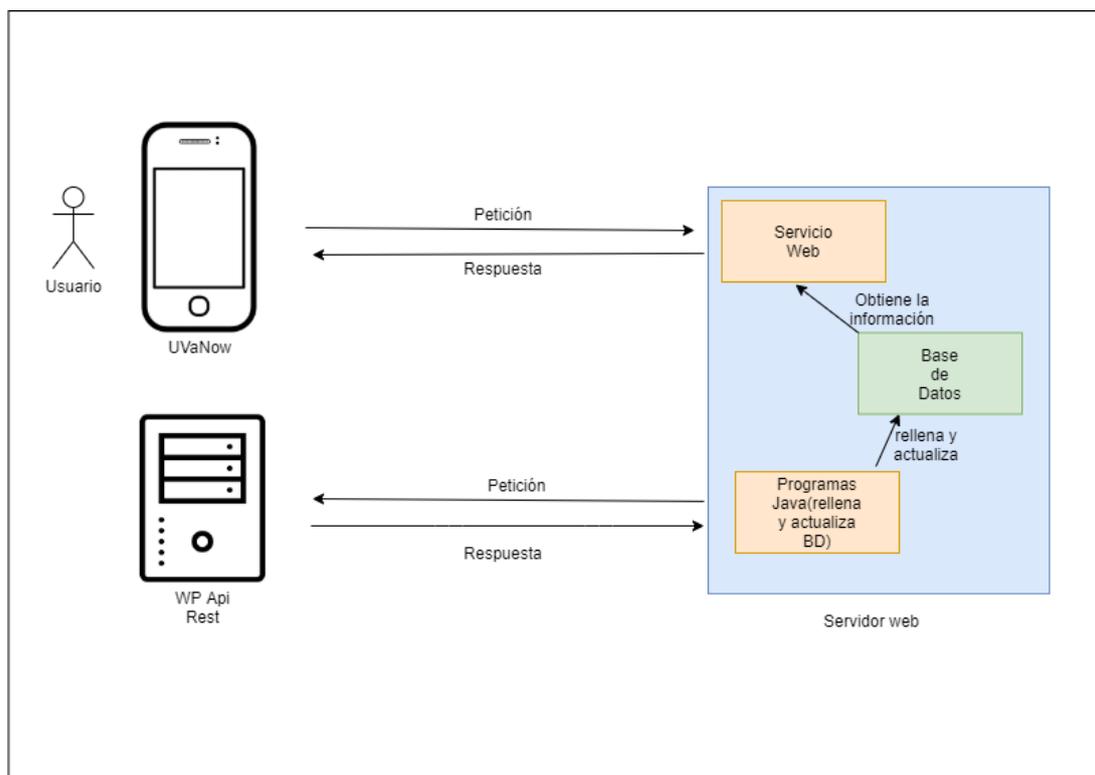


Figura 1: Esquema general de la aplicación

1.4 Estructura de la memoria del TFG

En el Capítulo 2 se describen todas las herramientas software utilizadas en el Trabajo de Fin de Grado (TFG), así como la metodología empleada en este proceso.

En el Capítulo 3 se detalla la actualización e implementación de las nuevas funcionalidades en la aplicación UVaNow.

En el Capítulo 4 se describe la instalación del nuevo servidor web, así como todas las herramientas y configuraciones que han sido necesarias. Respecto a la Base de Datos, se explica su despliegue, los cambios realizados en los campos requeridos para incorporar las nuevas funcionalidades y todo el proceso de monitorización implementado.

El Capítulo 5 abarca la implementación de los servicios web escritos en Java, desde el servicio web que conecta la APP de Android con el servidor web hasta los proyectos Java encargados de la actualización de la base de datos a partir del servicio Rest, propio de Wordpress, de la página web.

Para finalizar, el Capítulo 6 describe la subida al Market de la APP y sus requisitos y en el Capítulo 7 las conclusiones y las líneas futuras de este TFG.



2 Planificación

2.1 Metodología y Planificación

2.1.1 Documentación Previa y Metodología

Para la realización de este Trabajo de Fin de Grado se requirió un breve recordatorio de todos los conocimientos adquiridos a lo largo de la carrera y que nos iban a hacer falta. Principalmente el proyecto se basa en programación en Java, lenguaje predominante en la mayoría de las asignaturas cursadas en la carrera de Ingeniería Informática de la Universidad de Valladolid.

El proyecto usa principalmente Java, ya que Android es el lenguaje de programación que utiliza y también porque uno de los principales objetivos era la sustitución de los servicios web implementados en la primera versión en lenguaje PHP[8] por unos servicios web implementados en Java. Esto es así porque PHP es un lenguaje que empieza a estar desfasado y que carece de la seguridad que se puede aportar con otros lenguajes de programación como Java. Es importante remarcar que carecía de conocimientos previos sobre el desarrollo de aplicaciones Android y parte de la investigación previa al desarrollo consistió en tomar unas nociones básicas para luego en el transcurso del proyecto ir ampliando mis conocimientos según resolvía problemas o implementaba nuevas funcionalidades.

Otra parte destacada es la que tiene que ver con la configuración de distintas herramientas dentro del nuevo servidor proporcionado, así como el diseño de scripts para realizar copias automáticas de la base de datos o la ejecución diaria del programa de actualización de la base de datos creado también en Java.

Muchos de los problemas que obstaculizaron el avance del desarrollo o dudas que surgían han sido resueltas principalmente en dos páginas web, aunque como se aprecia en la bibliografía no fueron las únicas fuentes de información relevantes. La primera página que destacar fue *Android Developer* [9], muy importante para resolver dudas referentes al propio desarrollo en Android. Y la segunda, es una de las páginas más conocidas de programación que hay, *stackoverflow* [10], de donde se obtuvo muchas soluciones en todo ámbito del proyecto, desde la programación Android hasta el desarrollo de los servicios web, pasando por la configuración del entorno de trabajo en el servidor y el despliegue de aplicaciones en Tomcat.

El proceso de elaboración del proyecto se desempeñó mediante objetivos a corto y medio plazo que se estipulaban todas las semanas en las reuniones con los tutores y de los cuales se hacía seguimiento de su avance. La metodología empleada fue en cascada, con tres iteraciones que incluían tanto desarrollo como pruebas y estas iteraciones se han ido sucediendo en el tiempo de forma escalonada.

A parte de estos objetivos, también obtuvimos la información y el punto de vista de los alumnos de la Facultad de Periodismo de Valladolid, en una serie de reuniones y pruebas cerradas de la aplicación. De estas pruebas fueron surgiendo sugerencias y opiniones tanto generales de la aplicación como específicas de alguna de las funcionalidades o diseños implementados.

2.1.2 Planificación

Dentro del proceso de desarrollo, a pesar de marcar objetivos semanales, se planteó un orden en los objetivos principales del proyecto. Aunque, no siempre se podían implementar unos objetivos sin resolver otros, se siguió el planteamiento siguiente:

- **1º Trabajo sobre el Cliente Android:** Realizar cambios en el diseño inicial e integrar nuevas funcionalidades dentro de la aplicación Android. En primer lugar, se decidió comenzar con la programación en Android Studio por ser algo nuevo para mí y para familiarizarme cuanto antes con el desarrollo Android. Se realizaron las siguientes actividades:
 - Corregir problemas que presentaba en la aplicación en algunos puntos y rediseñar ciertas partes importantes de UVaNow.
 - También se implementaron nuevas funcionalidades. En algunos casos no era posible la finalización completa de alguno de los objetivos por necesitar los nuevos servicios web para su correcto funcionamiento, por ello se terminaron de probar después del despliegue de éstos en el servidor.
 - Realización de las pruebas necesarias para la comprobación del correcto funcionamiento de los cambios realizados, así como la realización de una prueba cerrada con personas de la Facultad de Periodismo.

Como es lógico, dentro de este conjunto de actividades se requirió investigar y aprender sobre el desarrollo en Android y finalizó con una prueba cerrada con los alumnos de la Facultad de Periodismo para que aportasen su opinión e indicasen sugerencias sobre la aplicación.

- **2º Desarrollo de los Servicios Web:** Desarrollar los nuevos servicios web en Java, así como implementar la funcionalidad de monitorización de la Aplicación móvil.
 - Un servicio web del que UVaNOW obtiene la información de la base de datos y el otro un programa que llamara a la API de Wordpress y rellenara la base de datos con las noticias subidas a la página web `inform@UVa`.
 - Creación de dos programas, el primero, uno que se ejecuta diariamente y rellena la base de datos con las nuevas noticias obtenidas y el segundo, un programa destinado para aquellos casos poco frecuentes en los que se necesite editar

alguna noticia desde Wordpress y que se encarga de actualizar todos los datos de la base de datos.

- Realizar una prueba con un gran volumen de usuarios, otorgándoles además un formulario para que plasmaran su opinión e impresiones sobre la aplicación.
- Implementar la creación de copias de seguridad de la base de datos cada 15 días.

En este último grupo de actividades hubo una gran labor de investigación, desde cómo desplegar aplicaciones hasta la creación de servicios web mediante el framework SpringBoot, pasando por el desarrollo de scripts para ejecutar programas de manera diaria.

Una vez finalizado todos los conjuntos de actividades planteados, realizado las pruebas y corregido los fallos detectados en esas pruebas, se procedió a su subida a la Play Store de Android para ser puesta en manos del público objetivo de una manera abierta y robusta. Para terminar el proyecto, se dejó la documentación del TFG apoyándose en los apuntes semanales realizados en Microsoft Teams[11] para escribir de manera precisa el proceso llevado a cabo en este TFG. Como se aprecia en la figura 2, la planificación se basa en la resolución de los dos conjuntos de actividades anteriormente descritos con sus respectivas pruebas y por último la documentación del proyecto.

Actividad	FECHA PREVISTA INICIO	DÍAS TRABAJADOS	FECHA FINAL PREVISTA
Documentación	13/04/2020	49	01/06/2020
Prueba Servicios Web	13/04/2020	27	10/05/2020
Desarrollo de Servicios Web	12/01/2020	91	12/04/2020
Prueba del Cliente Android	24/12/2019	25	18/01/2020
Trabajo sobre Cliente Android	01/10/2019	102	11/01/2020

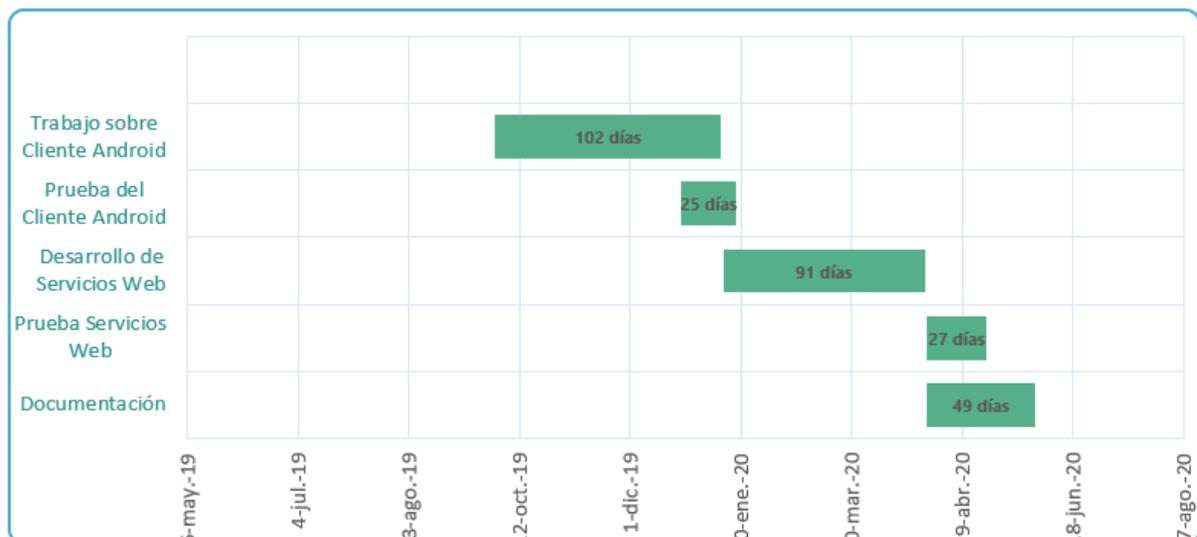


Figura 2: Planificación del proyecto

2.2 Seguimiento

El proyecto comenzó la semana del 28 de octubre de 2020 y acabó en la semana del 24 de junio de 2021. Como comentábamos en el apartado anterior, el proyecto se desarrolló dividido en dos grandes conjuntos de actividades. El primero constituía los arreglos, así como la implementación de las nuevas funcionalidades y cambios en el diseño de la aplicación, básicamente casi todo el trabajo sobre Android Studio. Se inició el 28 de octubre, el día 24 de diciembre de 2020 se realizaron las pruebas de la aplicación con los cambios hasta el momento para obtener las opiniones y posibles problemas en la parte implementada. Después del periodo de pruebas, se establecieron unas semanas para corregir y mejorar la aplicación con la información obtenida. El grupo de actividades se dio por concluido el 16 de febrero de 2021 cuando empezó el segundo conjunto de actividades, donde se implementaron los nuevos servicios web. A su vez se trabajó en la instalación y configuración de un nuevo servidor remoto proporcionado por la universidad ya que el anterior no tenía los permisos y características que necesitábamos. Esto abre un nuevo conjunto de actividades que realizar:

- **Migración de la Base de datos y de los Servicios Web:** Se nos concedió un nuevo servidor en el que tuviéramos permisos para instalar y configurar los servicios web y la base de datos como teníamos pensado, ya que con el anterior alojamiento web no podíamos.
 - Configurar y migrar la base de datos al nuevo alojamiento web.
 - Se realizaron los cambios en la base de datos necesarios.
 - Instalación y configuración de Java, MySQL, phpMyAdmin y Tomcat 7 en el servidor remoto.

Durante el proceso de configuración de las herramientas que íbamos a utilizar se siguieron numerosas guías de instalación. A parte de esto, para trabajar tanto en la configuración como en el desarrollo de los servicios web, nos fue de mucha ayuda el uso de una VPN privada de la Universidad de Valladolid para poder acceder al servidor de manera remota con interfaz gráfica.

. El desarrollo de los servicios web empezó más plenamente después de realizar la migración al nuevo alojamiento web porque dependíamos de que estuviera operativo. A partir del 12 de marzo, se trabajó paralelamente desarrollando los servicios web en java mientras fuimos configurando e instalando las herramientas necesarias en el nuevo servidor remoto que nos acababan de proporcionar, sin olvidar la migración y modificación de la base de datos. En dos

semanas se completó la mayor parte de la migración, dejando el servidor listo para funcionar con la aplicación mientras seguíamos desarrollando los servicios web con la facilidad que nos aportaba ahora el poder hacer pruebas en el servidor. En la semana del 12 de abril de 2021 finalizó el desarrollo de los servicios web y la monitorización de la aplicación. Posteriormente, la siguiente semana se empleó para la limpieza del código, realizar las copias de seguridad automáticas de la base de datos y corregir pequeños fallos. Para finalizar, hay que comentar que la aplicación empezó un proceso de prueba con 296 usuarios en abril y finalizó el 3 de junio con resultados satisfactorios. Desde el 19 de abril hasta la segunda semana de junio se procedió a la escritura de la documentación de Trabajo de Fin de Grado. Todo lo explicado se ve mejor y de manera gráficamente en la Figura 3. Me parece importante aclarar que la forma de proceder varió debido al cambio de servidor y por tanto pese a que ya desde un primer momento sabíamos que trabajaríamos en distintas partes de la aplicación al mismo tiempo, el orden se vio afectado obligándonos a trabajar en la migración antes de poder comenzar a avanzar plenamente en la creación de los servicios web.

Actividad	FECHA PREVISTA INICIO	DÍAS TRABAJADOS	FECHA FINAL PREVISTA
Documentación	19/04/2020	66	24/06/2020
Migración de BD y Servicios Web	12/03/2020	16	28/03/2020
Prueba Servicios Web	13/04/2020	51	03/06/2020
Desarrollo de Servicios Web	09/02/2020	63	12/04/2020
Prueba del Cliente Android	24/12/2019	54	16/02/2020
Trabajo sobre Cliente Android	28/10/2019	111	16/02/2020



Figura 3: Seguimiento del proyecto

2.3 Riesgos

Como todo proyecto, éste no está exento de riesgos que pueden suponer el incumplimiento del plan inicial establecido, obligándonos a realizar ajustes o incluir nuevas actividades según vayan surgiendo para corregir lo mayor posible el desvío del plan. Entre los riesgos que pueden surgir están las que se indican en la Tabla 1:

Referencia	Riesgo	Probabilidad	Impacto	Gestión del Riesgo
R1	Tiempos y costos poco realistas	Moderado	Moderado	Una buena planificación, empleando técnicas de estimación; un desarrollo incremental; análisis correcto del proyecto.
R2	Contratiempos derivados de problemas externos, como problemas con el alojamiento web	Moderado	Alto	Una rápida comunicación para conseguir un alojamiento web nuevo con las características necesarias; cambios en el alojamiento actual
R3	Cambios tardíos de requisitos	Bajo	Moderado	Control de cambios; Desarrollo incremental
R4	Actividad tarde más de lo esperado en desarrollarse	Moderado	Moderado	
R5	Desarrollo de funciones de software incorrectas	Bajo	Alto	Mejor evaluación del software; encuestas a los usuarios; creación de prototipos
R6	Enfermedad de individuo que afecte a desarrollo de actividad	Alto	Bajo	Tenerlo en cuenta en la planificación
R7	Desarrollo técnicamente complejo	Moderado	Moderado	Análisis técnico; formación; creación de prototipos

Tabla 1: Tabla de riesgos del proyecto

2.4 Presupuesto

Ahora estimaremos el coste que tendría un proyecto como este. En este proyecto ha intervenido una persona trabajando de desarrollador full stack, es decir que trabaja tanto de front como de back. El sueldo medio de un desarrollador full stack en España son 17.56 € a la hora según el portal web Talent.com [12]. El proyecto como Trabajo de Fin de Grado está establecido en 300h que multiplicado por el salario del desarrollador son 5268 € en términos de mano de obra. Suponemos que aparte de las pruebas en las máquinas virtuales de Android Studio, compramos un móvil para realizar las pruebas de una gama media. Esto supondría unos 199 € por un Xiaomi Redmi Note como ejemplo de smartphone de gama media. Podemos estimar el coste que supondría el alquiler del servidor desde que le empezamos a usar en el proyecto, esto fue en febrero así que diremos que el uso del servidor se extiende por 5 meses lo que supondría que siendo un alojamiento web compartido a unos 6,21 € al mes dando un total de 31.05 € de coste del servidor. Como todas las herramientas empleadas en el desarrollo son gratuitas, el coste por esta parte es cero. Ya solo queda calcular el coste que nos supone tener encendido un ordenador de sobremesa con el que trabajar en el proyecto. Un ordenador de sobremesa con unas especificaciones medias en el mercado actual como por ejemplo un ordenador con un procesador i5-9600k con 8 GB de memoria RAM y una GTX 1060 de 6GB de tarjeta gráfica tiene una media de consumo de 400 vatios más un monitor que consume unos 40 W dándonos un total de 440W que por las 300h del proyecto y por un precio estimado de 0,14 € cada kWh nos da un coste de 18.48 € de coste en consumo de electricidad. El coste total del proyecto sería de unos 5516,53 €.

2.5 Herramientas software empleadas

En este apartado se desglosan y describen todas las herramientas utilizadas en el diseño y desarrollo de este Trabajo de Fin de grado.

- *Android Studio* [13]: Es el entorno de desarrollo integrado oficial de Android para desarrollar aplicaciones. Fue lanzado de manera estable en 2014 sustituyendo a Eclipse como IDE oficial para el desarrollo en Android. Está basado en el software IntelliJ IDEA de JetBrains [14]. Este entorno admite varios lenguajes de programación como Kotlin, Java o C++. Entre sus principales características está el soporte para el desarrollo basado en Gradle, herramientas para detectar problemas de rendimiento, usabilidad y otros problemas, además también cuenta con diferentes plantillas para los diseños más habituales de Android y otros componentes. Es importante destacar su emulador, ya que permite probar en un entorno simulado la aplicación en diferentes

versiones de Android, diferentes teléfonos y dimensiones de pantalla. Durante el proceso de desarrollo de este proyecto el entorno ha recibido varias actualizaciones, finalizando el proceso con la versión 4.1.1 de Android Studio. En su creación se seleccionó la API 17 que equivale a la versión 4.2 Jelly Bean de Android. Para esta evolución de la aplicación se decidió que la API mínima de la App fuera la 21, lo que permite ser estable para la gran mayoría de dispositivos Android actuales y a su vez tener las ventajas a mayores que tiene la API 21 sobre la 17 que tenía la aplicación en primera instancia. Esta herramienta es fundamental, ya que en ella se han desarrollado todas las mejoras y nuevas funcionalidades de la aplicación, tales como el diseño y cambios visuales, fluidez, utilidad, nuevas funcionalidades y los ajustes necesarios para conectar y manejar la información de los nuevos servicios web.

- *PhpMyAdmin* [15]: Es una herramienta de software libre, utilizada para la gestión de bases de datos MySQL. PhpMyAdmin está desarrollado en el lenguaje de programación PHP [11]. Esta herramienta permite dicha gestión de las bases de datos tanto por su interfaz gráfica como por medio de consultas SQL, permitiendo todas las operaciones frecuentes como el manejo, creación de bases de datos, tablas, columnas y demás elementos. También facilita la exportación e importación de bases de datos, funcionalidad aprovechada en la migración del servidor en este proyecto.
- *VisualStudioCode* [16]: Es un editor de código creado por Microsoft y funciona en Windows, Linux y macOS. Contiene un gran número de características importantes, tales como control de versiones por Git integrado, resaltado de sintaxis, mucha personalización para poner atajos de teclado, temas del editor y demás preferencias. Visual Studio Code fue utilizado para la visualización de los servicios web PHP implementados en la primera versión de la aplicación.
- *Spring Tool Suite* [17]: Es un IDE basado en Eclipse, pero personalizado para trabajar con el Framework llamado Spring. A parte de dar soporte a este framework, tiene asistentes para la creación de proyectos en Spring, herramientas para la administración de beans, y más funcionalidades que facilitan el uso de Spring Boot que es lo que usamos para el desarrollo del servicio web.
- *Spring Boot*: Es una infraestructura diseñada para suprimir la mayoría del trabajo de configuración necesaria para las aplicaciones basadas en el Framework Spring. Permite configurar de manera automática las bibliotecas Spring y de terceros, proporciona dependencia de inicio para simplificar la configuración de la compilación, permite la creación de aplicaciones independientes Spring y también permite incrustar Tomcat u otros servidores directamente.
- *MySQL*: Es un sistema de gestión de base de datos relacional de código abierto desarrollado por MySQLAB [18] y una de las más populares junto a Microsoft SQL Server y Oracle. Dentro de nuestro alojamiento web se instaló y configuró MySQL para gestionar la base de datos.

-
- *GIMP* [19]: Es un programa de edición de imágenes que pertenece al proyecto GNU (GNU is Not Unix) de uso libre y gratuito. Su uso fue minoritario en el proyecto, pero se requirió esta herramienta para la creación de un nuevo botón.
 - *PuTTY* [20]: Es un cliente SSH de licencia libre disponible para Windows, Unix y Mac OS. Este programa permite conectarse a servidores remotos por medio de una consola, permitiendo también el envío de archivos entre ambas máquinas, en la que te conectas y la máquina remota. Útil para conectar con el servidor donde se alojan la base de datos y los servicios web.
 - *Microsoft Teams*: Es una plataforma diseñada para la comunicación y colaboración de las personas. Posee un chat persistente, reuniones por videoconferencia y la posibilidad de guardar los archivos e integrar aplicaciones. Esta ha sido una herramienta importante durante este Trabajo de Fin de Grado siendo usado como repositorio del trabajo anterior.
 - *FileZilla Client* [21]: Es una aplicación FTP gratuita y libre que contiene un cliente y un servidor y soporta ciertos protocolos como FTP (*File Transfer Protocol*), SFTP (*Secure File Transfer Protocol*) y FTP sobre SSL/TLS (*Secure Socket Layer*). Funciona sobre dispositivos Windows, Linux, FreeBSD y macOS. Incluye una interfaz gráfica y muchas características útiles que lo hacen un programa sencillo y confiable. Se ha utilizado en numerosas ocasiones para subir ficheros a nuestro alojamiento web.
 - Alojamiento web: Es un espacio donde se aloja un sitio web y se almacena información y servicios. Nuestro alojamiento en un primer momento se encontraba en la dirección <http://albergueweb1.uva.es/labcomuva> pero por la falta de permisos para instalar programas, administrar los ficheros y servicios y desplegar éstos, ha sido proporcionado otro alojamiento web, <http://157.88.130.185>.
 - *Postman* [22]: Es un programa que permite el envío de peticiones http Rest sin necesidad de crear un cliente para ello. Este programa nos permite probar el servicio web creado y desplegado en nuestro alojamiento web para comprobar si devuelve o realiza las peticiones correctamente de manera sencilla. También es de gran utilidad para comprobar las respuestas hechas a la WP Api Rest y saber así, como gestionar esa información recibida.
 - *Apache Tomcat*: Es un contenedor de servlet desarrollado por Apache Software Foundation. Catalina es el nombre que pasó a recibir el contenedor de servlets de Tomcat a partir de la versión 4x. Implementa especificaciones de Java Server Pages y Sun Microsystems para servlets, ambas basadas en Java. Fue instalado y configurado en el alojamiento web el Apache Tomcat 7 donde se desplegó el servicio web, esto es, la API Rest creada para conectar la aplicación Android con la base de datos alojada en nuestro servidor.
 - *Servicio Web (Web Service)* [23]: Un servicio web es una vía de intercomunicación entre máquinas a través de la red, ya que proporciona una interfaz entre ambas máquinas. Esto lo realiza mediante un conjunto de protocolos y estándares de comunicación con la

finalidad de facilitar un servicio en Internet. Por lo tanto, es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones o sistemas. Así pues, las aplicaciones software en diferentes lenguajes de programación, y ejecutadas sobre cualquier dispositivo, pueden usar servicios web para intercambiar datos en la red.

- *WordPress*: Es un sistema de gestión de contenidos lanzado en 2003 y desarrollado en el lenguaje de programación PHP. Wordpress está diseñado para crear y gestionar cualquier tipo de página web desde blogs hasta tiendas virtuales. Es utilizado para administrar y actualizar inform@UVa por parte de los alumnos de Periodismo de la Facultad de Filosofía y Letras y a partir de ésta obtener las noticias para nuestra aplicación móvil.
- *API REST de WordPress*: Esta herramienta propia del gestor de contenidos WordPress proporciona una interfaz para que las aplicaciones interactúen con dicho gestor. En este caso las dos aplicaciones alojadas en el servidor interactúan con esta API obteniendo un objeto JSON del que se obtiene la información que se guarda en la base de datos.
- *Google Drive* [24]: es un servicio de alojamiento de archivos que permite realizar también trabajos de manera grupal, pudiendo ver, editar y comentar en un archivo en tiempo real. Es la herramienta que hemos empleado para la documentación ya que nos permitía una revisión continua de los avances por parte de los tutores.
- *Netbeans* [25]: Es un entorno de desarrollo libre diseñado principalmente para la utilización del lenguaje de programación java con cantidad de módulos y plugins que amplían sus posibilidades. Es gratuito y sin restricciones de uso.

3

Actualización e Implementación de Nuevas Funcionalidades en la Aplicación UVaNow

3.1 Introducción

La primera parte del Trabajo de Fin de Grado ha consistido en realizar cambios funcionales y de diseño sobre la aplicación Android. Los cambios fueron realizados en Android Studio, el entorno de desarrollo oficial para Android.

Dentro de estos cambios se encuentran un conjunto de mejoras, tales como ajustes para el correcto funcionamiento del buscador, problemas en la longitud de la palabra introducida o la imposibilidad de escribir ciertos caracteres. Así mismo, también se llevaron a cabo mejoras en el funcionamiento del slider principal de la aplicación puesto que sufría problemas funcionales y de diseño. También se realizaron cambios en el diseño general y en la visualización y formato de las noticias.

Por otra parte, en este capítulo se describen los cambios realizados en el proyecto para incluir las nuevas funcionalidades a la aplicación. Estas nuevas funcionalidades han sido meditadas, consultadas y puestas a la opinión del público objetivo confirmando su utilidad y diseño dentro de la aplicación. Entre las nuevas funcionalidades implementadas se encuentra el guardado automático de ciertas noticias en el calendario del móvil y la posibilidad de compartir en redes sociales las noticias.

3.2 Análisis

En este primer grupo de actividades trabajamos dentro del cliente Android con el objetivo de conseguir cumplir los requisitos establecidos en el proyecto que mejorarán la aplicación en términos de experiencia de usuario, también con el añadido de nuevas funcionalidades y mejoras en el diseño.

En la siguiente lista se especifican detalladamente los requisitos funcionales y requisitos no funcionales tratados en este capítulo:

1. RF-1. Implementar en la aplicación la funcionalidad que permita compartir cualquier noticia en redes sociales desde la pantalla que muestra el contenido de la noticia. Mas específicamente, a través de las redes sociales de WhatsApp, Facebook y Twitter.
2. RF-2. Implementar en la aplicación la funcionalidad que permita guardar los eventos fechados que se indiquen en determinadas noticias para su posterior recordatorio en la aplicación Calendario de los teléfonos móviles con sistema operativo Android.
3. RNF-1. Incorporación de las imágenes principales de las noticias en la ventana donde se muestra el listado de noticias de cada categoría, mejorando así la facilidad de uso de la aplicación.
4. RNF-2. Mejorar el formato con el que se muestran las noticias en la aplicación, incorporando la posibilidad de mostrar scripts e imágenes y video dentro del cuerpo de la noticia. y el texto de la noticia en formato justificado.
5. RNF-3. Modificar el formato con el que se muestran las noticias en la aplicación para que el texto se muestra en formato justificado.
6. RNF-4. Corregir el problema asociado al buscador por el cual no resultaba efecto en búsquedas con palabras acentuadas o mayúsculas.
7. RNF-5. Corregir el problema asociado al buscador que impedía un uso de palabras de más de 10 caracteres.
8. RNF-6. Corregir el problema de acceso a las noticias del slider, permitiendo así a los usuarios entrar en las noticias seleccionadas del mismo.
9. RNF-7. Corregir el problema de visualización de las últimas noticias en el slider que no se mostraban correctamente.

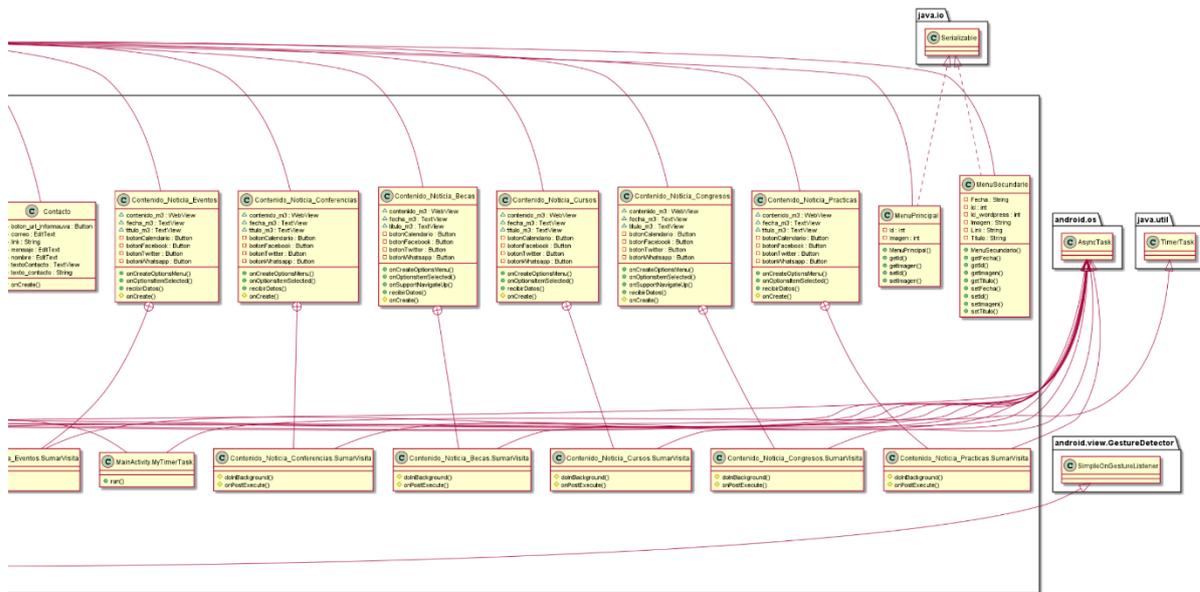


Figura 5: Segunda parte del diagrama de clases de la aplicación Android

3.4 Implementación

3.4.1 Mejoras en el buscador de la aplicación

UVaNow tenía una serie de problemas relacionadas con el buscador que hacían que no fuera plenamente funcional, por lo que uno de los objetivos de este Trabajo de Fin de Grado ha sido arreglar las deficiencias que tenía. Partimos del buscador creado en el trabajo anterior cuya base era buena y que funciona con una palabra clave con la que se busca la noticia o noticias que nos interesa ver.

El primer problema que fue solventado fue el tamaño máximo de la palabra admitida en el buscador. Este problema fue sencillo de solucionar aumentando dicho tamaño a 20 letras, ya que estaba establecido inicialmente en 10 letras, algo insuficiente para muchas palabras.

Otro defecto detectado en el buscador era que las palabras no aceptaban ciertos caracteres, tales como la “ñ” o letras acentuadas. Este problema se resolvió añadiendo esos caracteres que faltaban a la característica “android:digits” del *EditText* dentro de los layouts de cada categoría quedando el código del siguiente modo:

```
android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789áéíóúÁÉÍÓÚ"
```

Por otro lado, a las palabras que se obtienen del buscador se las efectúan dos cambios, el primero pasar a minúsculas y segundo quitar los acentos. Este proceso también se realiza con el título y el contenido de las noticias, de tal forma que no afecte a la búsqueda de la palabra que se escriba el acento y que la palabra esté en mayúsculas o minúsculas. Para poner las

palabras en minúscula se hace por medio de la función `toLowerCase()` y para quitar los acentos se hace con el método siguiente:

```
public static String cleanString(String texto) {  
    texto = Normalizer.normalize(texto, Normalizer.Form.NFD);  
    texto = texto.replaceAll("[\\p{InCombiningDiacriticalMarks}]", "");  
    return texto;  
}
```

Para finalizar, se añadió la característica de mostrar el puntero dentro del `EditText` para facilitar visualmente el uso del buscador, tal y como se muestra en la Figura 6.



Figura 6: Buscador de noticias

3.4.2 Mejoras en el slider y contenido de noticias

En este apartado comentaremos una serie de mejoras y correcciones implementadas divididas en dos bloques, el primero dirigido al contenido de las noticias y el segundo, todo lo referente al slider.

En el primer bloque se establecieron ciertos objetivos que se fueron expandiendo según fueron cumpliéndose. Para empezar, la justificación del texto en las noticias era algo importante puesto que da un nivel mayor de orden, mejor usabilidad y experiencia por parte de los usuarios. La justificación de los textos es habitual en medios informativos, dándoles un carácter más formal y serio, sumado a los factores beneficiosos descritos anteriormente. Realizar este cambio requirió cambiar la variable que representaba el contenido de tipo `TextView` a `WebView`, tanto en los archivos Java que implementan el contenido de las noticias en cada categoría como en sus respectivos layouts (Figura 7).

```
<WebView
    android:id="@+id/contenido_m3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:lineSpacingExtra="5dp"
    android:text=""
    android:layout_gravity="center_horizontal"
    android:textSize="14dp" />
```

Figura 7: Elemento WebView para mejorar el formato de las noticias

Como se puede apreciar en la Figura 4 se muestra el elemento *WebView* que se implementó con el fin de justificar el texto. Gracias a este cambio de elementos podemos aprovechar que la cadena del contenido que obtenemos de Wordpress en primer lugar y en última instancia de la base de datos está escrito en lenguaje HTML y viene justificado. *WebView* coge ese código HTML y lo plasma como si de una web se tratara, de tal modo que el texto queda justificado, además de darnos otras ventajas que se encontraban dentro de los objetivos.

Otra de las mejoras realizadas es que ahora también podemos acceder a enlaces desde dentro de las noticias, así como ver las imágenes y videos que están incluidos en dichas noticias. Esto anterior junto con la posibilidad de tener scripts y que funcionen correctamente en las noticias, se debe gracias a una serie de configuraciones que se aplican al elemento *WebView* y que se muestran en la Figura 8.

```
WebSettings webSettings = contenido_m3.getSettings();

webSettings.setJavaScriptEnabled(true);

contenido_m3.getSettings().setLayoutAlgorithm(WebSettings.LayoutAlgorithm.TEXT_AUTOSIZING);
contenido_m3.getSettings().setLoadWithOverviewMode(true);
contenido_m3.getSettings().setUseWideViewPort(true);
contenido_m3.getSettings().setMixedContentMode(WebSettings.MIXED_CONTENT_ALWAYS_ALLOW);
```

Figura 8: Configuraciones aplicadas a WebView

En la imagen de la Figura 5 vemos que la primera característica que habilitamos por medio de la función *setJavaScriptEnable(true)* es la posibilidad de ejecutar Scripts dentro de nuestro *WebView*. El siguiente ajuste es para establecer un algoritmo subyacente que provoca un

rediseño del *WebView*. En concreto, con el atributo que le añadimos, `TEXT_AUTOSIZING`, queremos que la fuente de los párrafos sea legible en ventanas amplias o en un modo de vista general. Otro ajuste, `setLoadWithOverviewMode`, se usa para reducir el contenido para que entre en el ancho de la pantalla, en el caso de que el ancho del contenido es mayor que el del *WebView*. Por último, la configuración de `setMizedContentMode`, se utiliza cuando *WebView* intenta cargar elementos desde un origen que considere inseguro por alguna circunstancia. En concreto, en nuestro caso generaba algunos problemas y por eso establecimos mediante el valor `MIXED_CONTENT_ALWAYS_ALLOW` que se permitiera todo el contenido de origen que erróneamente catalogaba de inseguro.

El segundo bloque de este apartado describe las mejoras y correcciones realizadas sobre el slider de la App. Éste presentaba principalmente algunos fallos que impedían su correcto funcionamiento, como el no poder acceder a la noticia pulsando sobre la imagen que en ese momento aparecía, convirtiendo al slider en un mero escaparate de las noticias más recientes y obligando al usuario a buscar la noticia dentro de las listas de noticias de cada categoría. También se detectaron problemas estéticos, tales como la visualización azul de las últimas posiciones del slider, no apareciendo ni imagen ni titular de noticia alguna.

El primer problema que se abordó fue este último, en que el error estaba relacionado con que estaban establecidas más vistas del slider (exactamente 10) que noticias se seleccionaban para visualizar dentro del mismo, que eran 6. Para la solución aprovechamos que queríamos tener más noticias en el slider ya que 6 noticias se quedaba algo escaso, así que aumentamos a 10, coincidiendo con el tamaño ya establecido del slider en el anterior proyecto.

El segundo punto importante estaba relacionado con solucionar la imposibilidad de acceder a la noticia de forma directa a través del propio slider, ya que no funcionaba. Para este problema se desarrolló el código que se muestra en la Figura 9. Lo que se hace en el código que se ve en la Figura 9 implementado en la clase *MainActivity.java* es una comparativa de las “coordenadas” de los píxeles en el momento en el que se pulsa en la pantalla y en el momento en el que se suelta de pulsar. De este modo, si la distancia o coordenadas entre cuando se pulsa sobre la noticia a cuando se suelta ha sido muy pequeña o nula, la aplicación avanza a la noticia del slider en que se encontraba. Por el contrario, si hay una distancia mayor significa que el usuario no tiene intención de acceder, sino que ha sido un movimiento para pasar de noticia y por lo tanto no carga la noticia. Esto se ve indicado por la variable *moved* en el código que toma su valor de la diferencia entre el primer punto y el segundo.

Por último, para finalizar con los cambios, se incluyó una animación en las transiciones del slider para hacer un efecto de alejamiento y acercamiento progresivo según se cambia de una pestaña a otra del slider.

```
viewPager.setOnTouchListener( new View.OnTouchListener() {
private boolean moved;
float currentX=0.0f;
float currentY=0.0f;
@Override
public boolean onTouch(View v, MotionEvent event) {

if (event.getAction() == MotionEvent.ACTION_DOWN) {
currentX = event.getX();
currentY = event.getY();
moved=false;
} if (event.getAction() == MotionEvent.ACTION_UP) {
float previousX = event.getX();
float previousY = event.getY();
float moveX=currentX-previousX;
float moveY=currentY-previousY;
if(moveX>=10.0f || moveX<=-10.0f || moveY>=10.0f || moveY<=-10.0f){
moved=true;
}else{
moved=false;
}
if (moved!=true) { //view.performClick();
//pasamos la fecha a caracteres alfanumericos
year1 = fecha[longitud - 1- viewPager.getCurrentItem()].substring(0,4);
mes1 = fecha[longitud - 1- viewPager.getCurrentItem()].substring(5, 7);
dial = fecha[longitud - 1- viewPager.getCurrentItem()].substring(8,10);

if(dial.equals("01")) {
dial = "1";
}
else if(dial.equals("02")) {
dial = "2";
}
}
}
}
```

Figura 9: Corrección del Slider

3.4.3 Mejoras en el diseño de visualización de las noticias

Uno de los puntos importantes del TFG era hacer un nuevo diseño para presentar la lista de las noticias de cada categoría en un formato más visual y atractivo. La aplicación presentaba las noticias al usuario meramente con su titular y fecha separados por unas líneas horizontales, tal y como se muestra en la Figura 10.



Figura 10: Presentación de las noticias en la anterior versión

Siguiendo los diseños con los que se planificó en un primer momento la aplicación, se pretendía ofrecer una lista con la imagen principal de la noticia y el titular, sin la imagen de la flecha ni la línea horizontal que inicialmente se había implementado (Figura 10).



Figura 11: Aplicación de la Universidad Complutense de Madrid

Este diseño que se pretendía aplicar surgió de un estudio realizado sobre aplicaciones similares desarrolladas para otras universidades, este es el caso de la aplicación de la Universidad de Granada o la desarrollada para la Universidad Complutense de Madrid, que tal y como se puede apreciar en la Figura 11.

Para llevar a cabo esta visualización, las noticias se obtienen a través de una petición a la API Rest implementada en el servidor remoto, y la respuesta a esta petición llega en forma de lista de elementos JSON. De estos elementos JSON sacamos los valores que nos interesan de cada noticia y son añadidos a una lista de elementos de tipo *MenuSecundario*, que es una clase que representa a cada item o elemento del menú secundario. A continuación, esta lista de elementos se pasa a la clase de java *Adaptador_menuSecundario* que sirve como adaptador para el ListView de los menús secundarios.

De esta manera todos los elementos que se necesitan para el nuevo diseño de presentación de las noticias están en la lista que se pasa a la clase llamada *Adaptador_menuSecundario*. Dentro de esta clase por medio del método *getView()*, método que se ejecuta en cada secuencia y vincula cada elemento de la lista con un título, imagen y fecha determinadas, obtenemos de

la lista de *MenuSecundario* las imágenes, titulares y fechas de cada noticia y son asociados a los respectivos elementos del layout. Los titulares y sus fechas ya se cogían en la primera versión, y como se muestra en la Figura 12, en esta versión incorporamos las imágenes principales de las noticias por medio de la librería Picasso. Esta librería para Android es utilizada para la gestión de imágenes y utiliza técnicas de cacheo, cargando y eliminando de la memoria volátil estas cuando son o no necesarias.

También se incluye una distinción para aquellas noticias que carezcan de imagen, en cuyo caso se pone una imagen predeterminada con el logo de la aplicación. Esto se hace comprobando la dimensión de String de la imagen, debido a que en la base de datos las noticias que no tienen imagen se guardan con un 0 para indicarlo. Si el tamaño del atributo imagen recuperado es mayor que uno, significa que esa noticia posee una imagen y se carga la imagen correspondiente.



Figura 12: Presentación de las noticias en la versión actual

En la Figura 12 se aprecia el diseño final de la aplicación, mostrando la imagen sobre el titular y la fecha en cada noticia, haciéndolo más atractivo e intuitivo para el usuario.

3.5 Diseño e implementación de nuevas funcionalidades

3.5.1 Guardar noticias en el calendario del móvil

Desde el comienzo del proyecto se meditó sobre la mejor manera de guardar los eventos, congresos o noticias que interesen al usuario. La primera idea que se propuso fue crear un apartado dentro de la aplicación, implementando un calendario interno donde se marcara el evento que se desea guardar junto con la fecha correspondiente al evento o suceso remarcado, tal como un congreso o una conferencia. Se consideró que la utilidad de una funcionalidad así se resiente al obligar al usuario a ver el calendario interno de la aplicación, en vez de usar el calendario propio del dispositivo móvil. Por ello, se decidió establecer un sistema por el que se cogieran las fechas y el lugar de celebración de los acontecimientos de la noticia y por medio de un botón se abriera el calendario. Cuando se abre el calendario se puede guardar el evento en el día concreto con las fechas de inicio y fin indicadas en la noticia, así como el título y el lugar de la celebración.

Esta opción parte de la idea de coger la información de la noticia para luego pasarla a la aplicación de Calendario del dispositivo Android. Para recoger los datos principales, hay que extraerlos de un String con el contenido de la noticia, por lo que se decidió que la mejor manera era establecer una plantilla en el lado de Wordpress que se pudiera detectar y sacar estos datos con facilidad. Trabajando con los alumnos de informa@UVa se realizaron una serie de pruebas para comprobar distintas versiones de plantillas que se diseñaron. Finalmente, la plantilla que quedó fijada fue la siguiente:

```
<strong>Lugar Celebración:</strong> Auditorio Miguel Delibes  
<strong>Fecha Comienzo:</strong>12-02-2020 <strong>Hora:</strong>10:40  
<strong>Fecha Final:</strong> 13-02-2020 <strong> Hora:</strong>13:50
```

Este es el código en formato HTML, tal y como se encuentra dentro del String del contenido de la noticia. De este segmento que se encontrará al final del String, se sacan algunos de los datos importantes, tales como el lugar de celebración, la fecha y hora de inicio y por último la fecha y hora final. Estos datos junto con el titular de la noticia son enviados a la aplicación del Calendario al pulsar el botón situado en la parte superior derecha, justo al lado del titular de la noticia. Parte del funcionamiento de la lectura de la plantilla era eliminar los elementos que no necesitábamos como los del lenguaje HTML, simplificando a lo mínimo el texto que forma la plantilla. Una vez hecho y sabiendo que los elementos que queremos extraer se encuentran siempre en el mismo orden, separamos las palabras mediante la función *split()*, que convierte un String en una Lista de elementos String seleccionando éstos según el carácter de separación que le indiquemos, puede ser una coma, un espacio, etc. En nuestro caso utilizamos un espacio en blanco como separador de las palabras. A parte de eliminar lo que no se necesita, se incluyó una cierta flexibilidad para no ser 100% estrictos, como el caso de los espacios en blanco que no afectan a la hora de interpretar los datos de la plantilla puesto que antes de separar las diferentes palabras, al String se le eliminan los espacios sobrantes dejando solo uno entre

palabras. Otro punto importante es una variación posible dentro de la plantilla, que es la inclusión o no del elemento “Hora”. Hay veces en que el acontecimiento anunciado no tiene una hora de inicio o de fin, sino solamente una fecha, lo que también se contempla localizando dentro de la plantilla la palabra “Hora” y modificando como es lógico los índices donde se encuentran las palabras objetivo.

El diseño del botón de guardar en el calendario en nuestra APP se realizó con la herramienta de editado de imágenes GIMP, haciendo dos imágenes de un calendario minimalista, una en azul y otra igual en rojo para cambiar el color del botón cuando está pulsado el botón. Una medida para evitar fallos graves a la hora de leer la plantilla es solo mostrar el botón en el caso de que la plantilla esté bien escrita y se pueda coger bien los datos, de esta forma ante algún despiste o errata al escribir la noticia no se cierra la aplicación. Como es lógico, en el caso de que la noticia no tenga una fecha de realización de un evento, esta función está también deshabilitada. Esta imagen aparece situada en la parte superior derecha del Titular de la noticia, tal y como se muestra en la Figura 13.



Figura 13: Imagen del contenido de una noticia y nuevo botón para guardar noticias

En la Figura 13 se muestra el botón implementado en las ventanas de contenido de las noticias, a la izquierda en azul cuando no está pulsado y a la derecha cuando se pulsa. Una vez que el usuario suelte el dedo del botón se abre en su dispositivo la aplicación del calendario con los datos del evento ya establecidos para guardarlo en caso de que así se quiera (Figura 10).

Como ejemplo, se puede apreciar en la Figura 14 que se abre la aplicación del Calendario del dispositivo móvil preparada para guardar el evento con el nombre del titular de la noticia, las fechas y horas de inicio y final, así como la localización (en este caso es Online). Ya dentro de esta ventana el usuario puede configurar como quiere ser avisado con las opciones que aporta la propia aplicación de Calendario de Android.



Figura 14: Aplicación Calendario con los datos de la noticia guardada

3.5.2 Compartir noticias a través de redes sociales

La funcionalidad de compartir en redes sociales es básica dentro del estándar de aplicaciones móviles o webs destinadas a difundir información. Por consiguiente, realizamos una revisión de cómo se implementa esta funcionalidad en numerosas aplicaciones similares para hallar el diseño más apropiado. A su vez, también había que concretar qué sería lo que se compartiría y en qué redes sociales (RRSS) concretas.

Una vez investigado se decidió por un formato minimalista con solo los iconos de las RRSS seleccionadas y una etiqueta a su izquierda indicando la función de los botones para una mayor aclaración de la funcionalidad de estos. Las redes sociales elegidas fueron Facebook, Twitter y

WhatsApp debido a sus características y formato más adecuado para la compartición de noticias mediante enlaces, y sobre todo por su uso mayoritario entre usuarios jóvenes.

Se estableció que, al compartir una noticia, se hiciera a través del enlace de la noticia que provenía de la propia página web de inform@UVA. Este procedimiento no se podía realizar con los servicios web antiguos porque el enlace no era extraído del JSON de respuesta a la WP API Rest e introducido en la base de datos. Después de solventar ese problema extrayendo ese dato y guardándolo en la base de datos en un nuevo campo, ya se pudieron usar los tres botones creados para las tres RRSS, tal y como se muestra en la Figura 15.

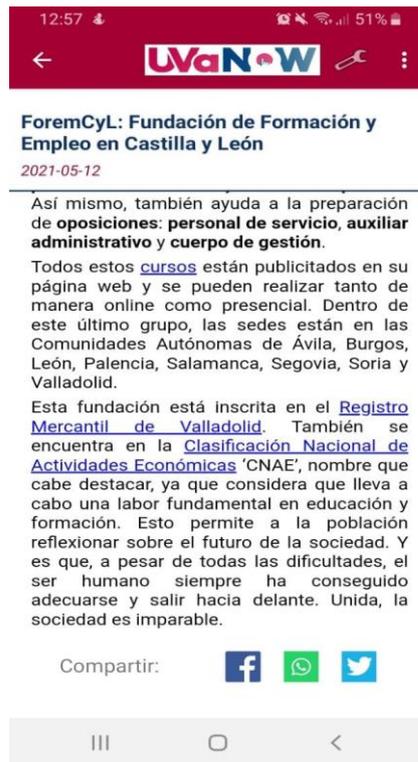


Figura 15: Nuevos botones para compartir en Redes Sociales

3.6 Pruebas

Número Prueba	Título Prueba	Resumen	Pasos	Resultado
1	Prueba del buscador	Probar con una palabra en mayúscula, con acento y con un numero de caracteres elevado en este	1-Seleccionar una categoría 2-Seleccionar el buscador e introducir la palabra "tecnologia" en	La palabra fue encontrada correctamente tanto en titulares de noticia como en noticias que

		caso “Tecnología” para buscarlo sin acento ni mayúsculas, comprobando así la solidez del buscador	minúsculas y sin acento	contenían dicha palabra a pesar de escribirla sin acentos ni mayúscula inicial.
2	Prueba slider	Comprobamos que el slider funciona correctamente, moviéndonos entre las noticias que hay en el slider y seleccionando una.	1-Movemos hacia la izquierda el slider para que nos muestre la noticia siguiente. 2-Seleccionamos dicha noticia.	Accedemos correctamente a la noticia que se seleccionó.
3	Prueba compartir RRSS	Realizamos un ejemplo de compartir una noticia en redes sociales	1-Accedemos a una noticia. 2-Deslizamos hacia abajo del todo y seleccionamos el icono de Whatsapp. 3-Seleccionamos el contacto al que enviar el enlace a la noticia	El contacto objetivo recibe el enlace a la noticia compartida desde la Aplicación
4	Prueba de funcionalidad de guardado de eventos en calendario	Probamos la nueva funcionalidad guardando en nuestra aplicación de calendario una noticia que tenga	1-Acceder a una categoría, por ejemplo “Eventos”. 2-Seleccionamos una noticia que tenga un evento, conferencia, ... con fecha concreta.	La noticia se guardó correctamente en la aplicación Calendario de nuestro móvil con datos cogidos de la

		un evento con fecha concreta	<p>3-Dentro de la noticia presionamos el botón situado a la derecha del título y se accede al Calendario de nuestra aplicación con los datos de la noticia.</p> <p>4-Guardamos el evento en nuestro calendario personal del móvil</p>	noticia correctamente.
--	--	------------------------------	---	------------------------

Tabla 2: Pruebas de la parte del Cliente Android

Para probar esta parte del desarrollo de la parte del cliente Android se entregó la aplicación a un grupo de pruebas reducido que agrupaban profesores y alumnos de la Facultad de Periodismo para obtener las opiniones al respecto. En línea general las opiniones fueron favorables y la mayoría de las opiniones que me llegaban eran proposiciones de pequeñas mejoras o pequeños fallos que fueron solventados. Estas opiniones llegaban directamente de los propios alumnos o a través de mi tutora ya que era una prueba muy cerrada no se realizó una encuesta por la poca muestra que representaría y por qué el objetivo era conocer de primera mano la visión de las personas de la Facultad de Periodismo.

3.7 Conclusiones

A lo largo de este proceso de mejora de la aplicación Android he aprendido mucho del funcionamiento interno de las aplicaciones, del diseño de sus interfaces y del manejo general de una aplicación como Android Studio, herramienta muy importante dentro del desarrollo de aplicaciones para Android. Además de estos conocimientos ganados hay que destacar la investigación realizada y las numerosas pruebas que hubo en el desarrollo tanto en las nuevas funcionalidades como en las mejoras y correcciones de la anterior versión de UVaNOW.

En este capítulo se han detallado todas las funcionalidades nuevas que se han implementado, es posible que en futuras versiones se incorporen alguna funcionalidad a mayores.

4

Migración de la Base de Datos y el Servidor Web

4.1 Introducción

Uno de los objetivos marcados para este proyecto es el desplegar un servicio web y hacer la monitorización de accesos a la base de datos. Este proceso no es posible con el alojamiento web inicialmente otorgado para esta aplicación, ya que carecemos de los permisos necesarios para acceder, instalar y administrar todo lo que requerimos para hacerlo. Por tanto, se decidió cambiar de alojamiento web y se nos proporcionó otro servidor. El servidor, con sistema operativo Debian, tenía que ser configurado con las herramientas requeridas para el proyecto y en este capítulo contaremos todo este proceso llevado a cabo. Empezando por como accedíamos al nuevo servidor, después contaremos los requisitos en el apartado de análisis junto con la descripción del alojamiento inicial. Mas adelante, contaremos en el apartado de diseño la arquitectura de la capa de datos y lógica para terminar contando la implementación de la migración y las distintas configuraciones realizadas en el servidor. Sin olvidar, un apartado para las pruebas realizadas.

4.2 Modo de acceso al Servidor

El nuevo servidor proporcionado para el proyecto se aloja en las instalaciones de la universidad con una dirección que no plasmaremos en este documento por su carácter público. Se instaló el sistema operativo Debian 10, que es un SO usual para servidores, y además se configuró el servidor en cuestión de seguridad. El resto de las herramientas y configuraciones necesarias para el proyecto comentadas en la introducción fueron realizadas a través del acceso remoto al servidor. El proceso para conseguir el acceso al servidor remoto se explicará a continuación.

Para acceder mediante consola el proceso es bastante sencillo. A través de la aplicación Putty, que utilizaremos para conectarnos en modo consola, solo necesitamos acceder mediante el comando ssh y con el nombre de usuario y contraseña que se nos concedió.

```
ssh user@XXX.XXX.XXX.XXX
```

Pero para una mayor facilidad y para configurar ciertos elementos del servidor, lo mejor es tener un acceso mediante interfaz gráfica al servidor. Para ello se tuvieron que seguir unos pasos para utilizar la VPN proporcionada por los técnicos de la ETSI Telecomunicación, ya que se necesitan unos permisos para poder acceder desde esa VPN al servidor remoto utilizando la herramienta de Windows llamada Conexión a Escritorio Remoto (Figura 16).

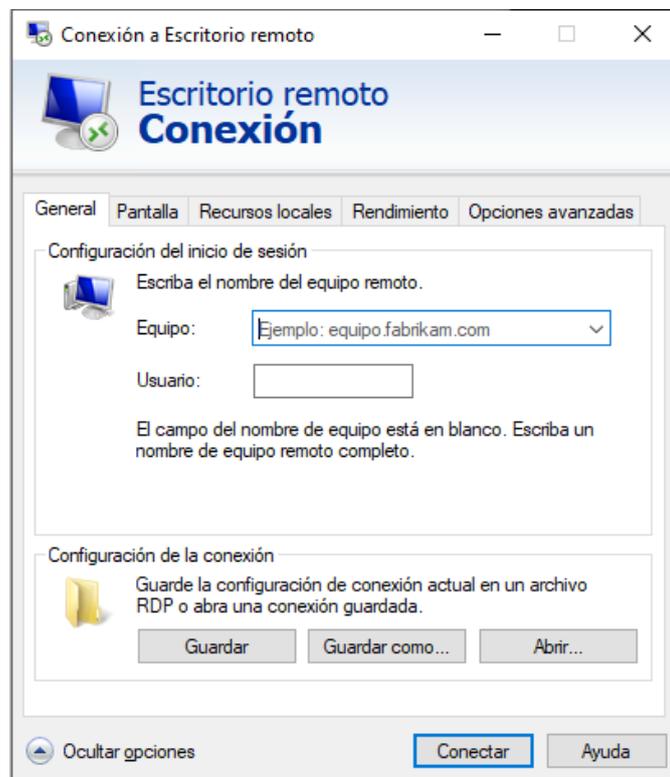


Figura 16: Acceso a escritorio remoto con interfaz gráfica

4.3 Análisis

El primer alojamiento web que poseía la aplicación y en el cual se encontraban tanto la base de datos como los servicios web escritos en PHP carecía de los permisos y posibilidades que nos permitieran realizar lo que pretendíamos en este proyecto. Básicamente constaba de un acceso limitado a dos directorios llamados /www donde se almacenan los servicios web, otro directorio /logs donde como su nombre indica se alojarán los logs correspondientes. Por último, accediendo por medio de la URL formada por la dirección del alojamiento web y el path /phpmyadmin obtenemos acceso al gestor de bases de datos de phpMyAdmin alojado en el servidor. No teníamos capacidad ni permisos para acceder a otros directorios ni para acceder a la interfaz gráfica ni instalar las herramientas que necesitábamos para el proyecto.

Los requisitos que se abordan en este capítulo son los referentes a la migración de la base de datos y la configuración del nuevo alojamiento web. Los requisitos funcionales que se tratan en este capítulo son los siguientes:

1. RF-1. Instalar y configurar JDK de Java en el nuevo alojamiento web proporcionado.
2. RF-2. Instalar y configurar MySQL para poder crear la base de datos en nuestro servidor remoto.
3. RF-3. Instalar y configurar phpMyAdmin para gestionar la base de datos de la aplicación de manera más sencilla y visual.
4. RF-4. Instalar y configurar de un servidor Apache Tomcat 7 donde se desplegará el servicio web que conecta la aplicación Android con la base de datos.
5. RF-5. Migrar la base de datos desde el antiguo alojamiento web de la aplicación al nuevo alojamiento web proporcionado, donde antes para este cometido se habría instalado y configurado tanto MySQL como phpMyAdmin para crear y gestionar la base de datos de la aplicación.
6. RF-6. Modificación de la base de datos de la aplicación para la inclusión de los nuevos atributos que permitan llevar a cabo las nuevas funcionalidades implementadas en el proyecto de compartición en redes sociales y guardado de eventos en el calendario del móvil, así como que permitan implementación de la monitorización de la aplicación.
7. RF-7. Obtener dentro del servidor remoto una copia de la base de datos generada los días 1 y 15 de cada mes de manera automática con el nombre del fichero indicando la fecha de la copia.

4.4 Diseño

El nuevo alojamiento web corresponde a un servidor alojado físicamente en la Universidad de Valladolid con un sistema operativo Debian 10, sistema operativo propio de servidores. Este alojamiento web contendrá un servidor Tomcat 7 totalmente configurado para el posterior despliegue del servidor web que conectará al cliente Android con la base de datos también alojada en el servidor remoto como se aprecia en la Figura 17.

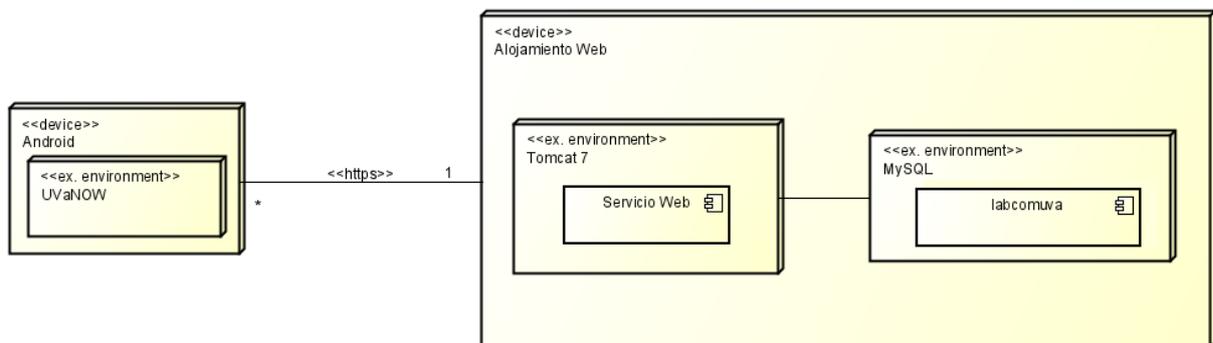


Figura 17: Arquitectura del alojamiento web

La migración de la base de datos la realizamos por medio del gestor de bases de datos phpMyAdmin instalado en ambos alojamientos, permitiéndonos exportar e importar la base de datos con todos sus datos. A través de la herramienta Filezilla podemos pasar el archivo en formato SQL que luego desde dentro del servidor importaremos.

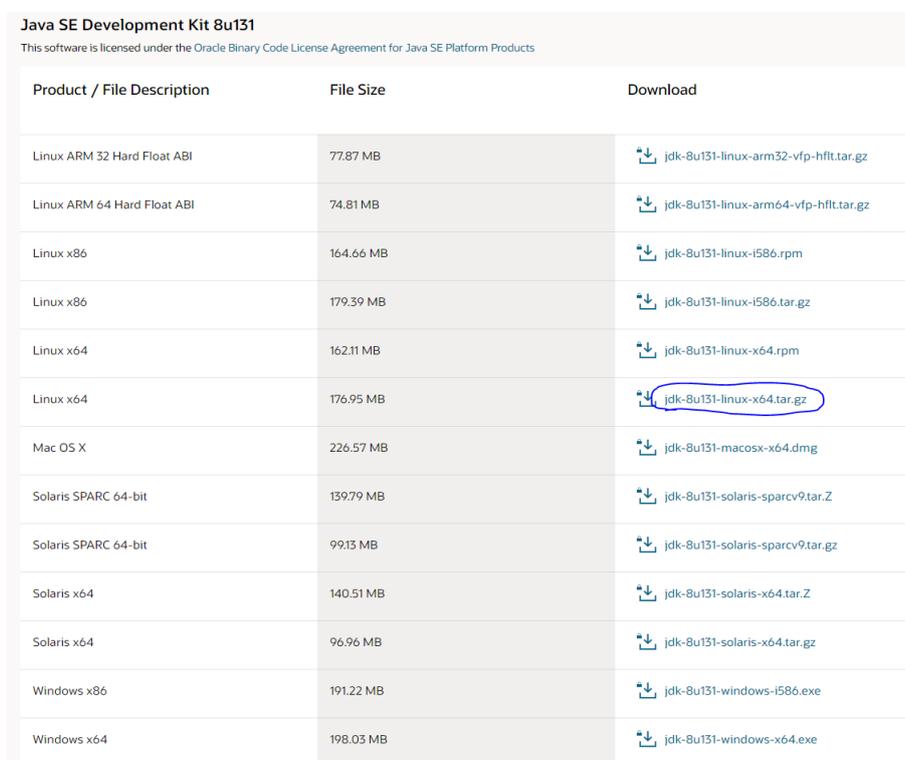
Optamos por la instalación del JDK 8 pro ser una versión estable y por ser la versión que he usado en mi paso por la universidad, más concretamente la versión JDK 8 u131 fue la elegida. Para la versión de MySQL optamos por el mismo motivo, a parte de por sus prestaciones, por la versión 8 de la herramienta que junto a phpMyAdmin eran las herramientas utilizadas para de toda la parte de la base de datos de la aplicación. En un primer momento se iba a instalar y configurar el servidor Apache Tomcat 9 pero debido a algunos problemas que se generaron al desplegar se optó finalmente por la versión 7 del servidor como lugar para desplegar nuestro servicio web.

4.5 Implementación

4.5.1 Instalación de JDK en el servidor

El primer paso de la configuración del servidor es instalar el *Java Development Kit* [26], que proporciona herramientas para el desarrollo de programas escritos en lenguaje Java. Optamos por la instalación de la versión JDK 8 por ser una versión estable y con la que se desarrollaron los proyectos que desplegamos en el servidor. Para su instalación se siguieron los siguientes pasos:

1. Se accede a la cuenta oficial de Oracle previamente registrado y buscamos la versión que nos interesa de linux para descargarlo en nuestro servidor, tal y como se muestra en la Figura 18.



Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.87 MB	jdk-8u131-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.81 MB	jdk-8u131-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.66 MB	jdk-8u131-linux-i586.rpm
Linux x86	179.39 MB	jdk-8u131-linux-i586.tar.gz
Linux x64	162.11 MB	jdk-8u131-linux-x64.rpm
Linux x64	176.95 MB	jdk-8u131-linux-x64.tar.gz
Mac OS X	226.57 MB	jdk-8u131-macosx-x64.dmg
Solaris SPARC 64-bit	139.79 MB	jdk-8u131-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.13 MB	jdk-8u131-solaris-sparcv9.tar.gz
Solaris x64	140.51 MB	jdk-8u131-solaris-x64.tar.Z
Solaris x64	96.96 MB	jdk-8u131-solaris-x64.tar.gz
Windows x86	191.22 MB	jdk-8u131-windows-i586.exe
Windows x64	198.03 MB	jdk-8u131-windows-x64.exe

Figura 18: Página web de Oracle donde descargar JDK 8u131

2. Una vez descargado el archivo `jdk-8u131-linux-x64.tar.gz` lo movemos y luego lo descomprimos en el directorio `/usr/lib/jvm/` mediante los comandos siguientes:

```
mv /home/noemer/Descargas/jdk-8u131-linux-x64.tar.gz /usr/lib/jvm
cd /usr/lib/jvm
tar xzvf jdk-8u131-linux-x64.tar.gz
```

3. A continuación, se configura la variable `JAVA_HOME`, sabiendo donde ésta instalado, escribiendo los siguientes comandos:

```
sudo vim /etc/environment
```

Este primer comando sirve para editar el archivo `environment` y allí añadir la siguiente línea al final:

```
JAVA_HOME="/usr/lib/jvm/java-8-versionQueTengamos/"
```

Después guardamos y cerramos el archivo. Para comprobar que se ha guardado la variable de entorno ejecutamos el siguiente comando:

```
echo $JAVA_HOME
```

Y debería mostrarnos la ruta correcta. En el caso en que no sepamos la dirección de la variable de entorno o queramos cambiar entre varios JDK instalados en el servidor, existe el comando:

```
sudo update-alternatives --config java
```

En este caso, se nos mostrará una lista con las rutas de las distintas instalaciones Java y podremos seleccionar cual es la que nos interesa que se establezca como variable de entorno, tal y como se muestra en la Figura 19. Como se observa en la Figura 15, aparecen tres directorios donde puedes establecer la variable de entorno en este caso de ejemplo.

```
Output
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path
-----
  0            /usr/lib/jvm/java-11-openjdk-amd64/bin/java  1111  auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java  1111  manual mode
* 2            /usr/lib/jvm/java-11-oracle/bin/java        1091  manual mode

Press <enter> to keep the current choice[*], or type selection number:
```

Figura 19: Lista de JDK instaladas en el servidor

4.5.2 Instalación y configuración de MySQL

MySQL es parte fundamental del proyecto debido al uso de una base de datos para la aplicación. Para la instalación y configuración de MySQL en un servidor Debian, se necesita seguir los siguientes pasos:

1. El primer paso es actualizar los paquetes actuales a la última versión con los siguientes comandos:

```
sudo apt update
```

```
sudo apt upgrade
```

-
2. Después descargamos el archivo necesario para su instalación y lo ejecutamos utilizando los comandos:

```
wget http://repo.mysql.com/mysql-apt-config_0.8.13-1_all.deb  
sudo dpkg -i mysql-apt-config_0.8.13-1_all.de
```

Durante la instalación se nos pedirá que seleccionemos la versión a instalar entre la 8.0 y la 5.7, en nuestro caso seleccionamos la 8.0. En el caso de que se necesite cambiar alguna configuración específica siempre se puede ejecutar comando:

```
sudo dpkg-reconfigure mysql-apt-config
```

Una vez seleccionada una versión, se guarda. Y ya está el sistema listo para instalar MySQL.

3. Instalamos MySQL mediante los siguientes comandos:

```
sudo apt update  
sudo apt install mysql-server
```

Para este proceso se solicitará la contraseña de root que quedará como predeterminada (Figura 20). Se pedirá dos veces para confirmar la contraseña, que será la contraseña del usuario root de MySQL, necesaria para iniciar sesión en el servidor MySQL.

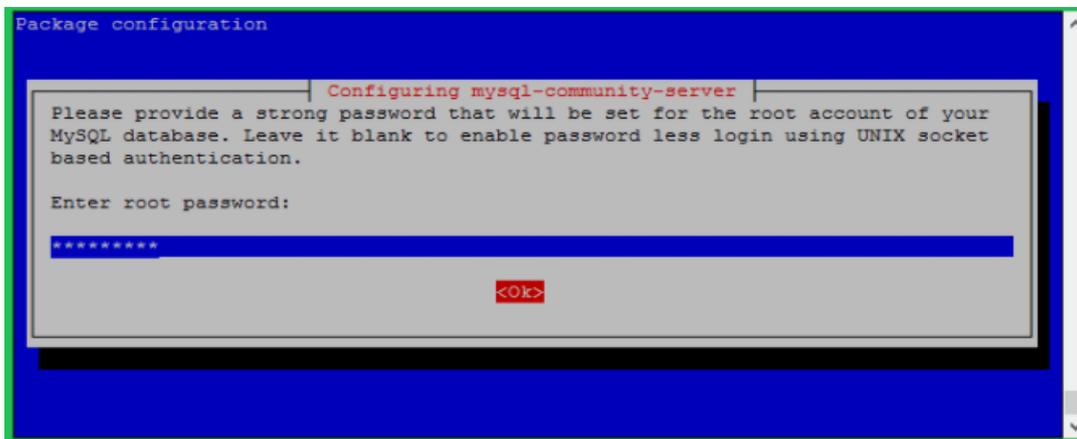


Figura 20: Configuración en instalación MySQL, petición de contraseña root

La siguiente ventana, Figura 21, nos muestra unas opciones de seguridad para las contraseñas propias de la versión MySQL 8. Optamos por seleccionar la opción que nos indica como recomendada y que aporta una seguridad extra a nuestra contraseña.

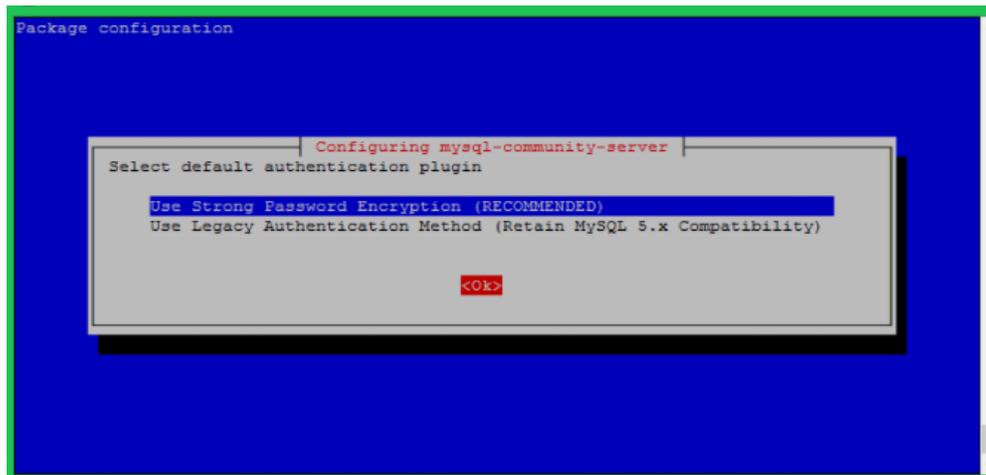


Figura 21: Elección de seguridad dentro de instalación MySQL

Para finalizar el proceso es necesario iniciar el servidor con el comando:

```
sudo systemctl restart mysql.service
```

4. El siguiente paso es conectarse al servidor MySQL, para ello utilizamos la siguiente línea de comandos:

```
mysql -u root -p contraseña
```

Con estos pasos queda instalado y configurado MySQL en nuestro servidor remoto, y podemos establecer la base datos de la aplicación, que estará gestionada mediante phpMyAdmin. En el siguiente apartado detallaremos su instalación y configuración completas.

4.5.3 Instalación y configuración de phpMyAdmin

Para la instalación de phpMyAdmin es necesario previamente tener instalado lo que se conoce como LAMP Sack [27] que es el software conformado por Linux, Apache (servidor web), MySQL y PHP. En este apartado veremos la instalación y configuración de todos los elementos necesarios para la instalación de phpMyAdmin, así como la propia configuración del administrador de bases de datos. Para ello, se tienen que seguir los siguientes pasos:

1. El primer paso sería instalar wget para poder descargar desde el terminal, pero ya lo teníamos instalado de anteriores usos de ese comando, por tanto, instalaremos el Apache2:

```
sudo apt install apache2 -y
```

Y después de instalarse, comprobamos el estado del servidor a ver si está ejecutándose (Figura 22):

```
systemctl status apache2
```

```
phoenixnap@phoenixnap:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
   Active: active (running) since Mon 2020-05-04 04:45:38 MST; 1min 47s ago
     Docs: https://httpd.apache.org/docs/2.4/
  Main PID: 11282 (apache2)
    Tasks: 55 (limit: 2347)
   Memory: 20.8M
    CGroup: /system.slice/apache2.service
            └─11282 /usr/sbin/apache2 -k start
              └─11284 /usr/sbin/apache2 -k start
                └─11285 /usr/sbin/apache2 -k start

May 04 04:45:38 phoenixnap systemd[1]: Starting The Apache HTTP Server...
```

Figura 22: Comprobación del estado del servidor apache2

2. El siguiente paso es instalar el lenguaje PHP en nuestro Debian 10 ya que es esencial para el uso de la aplicación phpMyAdmin. Para ello procederemos a instalar los paquetes principales de PHP junto con los complementos de apache y MySQL:

```
sudo apt install php php-cgi php-mysql php-pear php-mbstring php-gettext libapache2-mod-php php-common php-phpseclib php-mysql -y
```

Para comprobar la versión ejecutamos el comando:

```
php --version
```

3. Una vez instalado MySQL, solo queda realizar algunos ajustes de configuración del mismo, esto pasa por poner el siguiente comando y elegir entre las opciones que se proponen:

```
sudo mysql_secure_installation
```

De esta manera podremos realizar configuraciones como integrar el plugin de validación de contraseña, si quieres eliminar usuarios anónimos y bases de datos de prueba. También se podrá deshabilitar los inicios de sesión remotos para más seguridad y todos estos ajustes se cargarán en el MySQL de nuestro servidor. Luego ya se puede crear los usuarios que queramos para acceder a la base de datos. En nuestro caso trabajaremos con el usuario root.

4. Una vez que tenemos todos los requisitos previos para poder instalar phpMyAdmin, nos disponemos a descargar la última versión a nuestra carpeta de descargas del siguiente modo:

```
wget -P Descargas https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.tar.gz
```

Una vez descargado, necesitamos verificar la clave GPG de phpMyAdmin. Para ello descargamos el archivo siguiente en el mismo directorio que el anterior, en nuestro caso Descargas:

```
wget -P Downloads https://files.phpmyadmin.net/phpmyadmin.keyring
```

Accedemos al directorio de Descargas y ejecutamos el siguiente comando para importar el keyring:

```
gpg --import phpmyadmin.keyring
```

Después, descargamos el archivo GPG.asc correspondiente a nuestro phpMyAdmin:

```
wget -P Downloads  
https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-  
languages.tar.gz.asc
```

A continuación, se ejecuta el comando para verificar el archivo .asc que contiene las claves:

```
gpg --verify phpMyAdmin-latest-all-languages.tar.gz.asc
```

Es entonces cuando el sistema mostrará la información de la clave GPG y las comparará con las credenciales de desarrollador de la página oficial.

5. A continuación, se procede a desempaquetar y configurar nuestro administrador de base de datos y lo primero es crear un directorio en la siguiente ruta:

```
sudo mkdir /var/www/html/phpMyAdmin
```

Desde el directorio de descargas desempaquetamos el archivo que contiene la última versión de la aplicación en el nuevo directorio creado mediante el comando:

```
sudo tar xvf phpMyAdmin-latest-all-languages.tar.gz --strip-components=1 -C  
/var/www/html/phpmyadmin
```

Lo siguiente, es crear un archivo de configuración predeterminado:

```
sudo cp /var/www/html/phpmyadmin/config.sample.inc.php  
/var/www/html/phpmyadmin/config.inc.php
```

Mediante un editor de textos cualquiera, abrimos ese archivo de configuración y buscamos la siguiente línea `$cfg['blowfish_secret']=''`, tal y como se muestra en la Figura 23:

```
sudo vim /var/www/html/phpmyadmin/config.inc.php
```

```

<?php
/**
 * phpMyAdmin sample configuration, you can use it as base for
 * manual configuration. For easier setup you can use setup/
 *
 * All directives are explained in documentation in the doc/ folder
 * or at <https://docs.phpmyadmin.net/>.
 */

declare(strict_types=1);

/**
 * This is needed for cookie based authentication to encrypt password in
 * cookie. Needs to be 32 chars long.
 */
$config['blowfish_secret'] = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */

/**
 * Servers configuration
 */
$i = 0;

/**
 * First server
 */

```

Figura 23: Configuración contraseña phpMyAdmin

En esta línea se agrega una contraseña secreta y posteriormente guardamos y salimos del fichero. Lo siguiente es cambiar los permisos sobre ese fichero usando el comando `chmod`:

```
sudo chmod 660 /var/www/html/phpmyadmin/config.inc.php
```

Cambiamos también las propiedades del directorio de `phpmyadmin` y por último reiniciamos el servidor Apache con los siguientes comandos:

```
sudo chown -R www-data:www-data /var/www/html/phpmyadmin
sudo systemctl restart apache2
```

De esta forma ya podremos acceder desde nuestro navegador a `phpMyAdmin` a través de la dirección web `localhost/phpmyadmin`.

4.5.4 Migración y modificaciones de la Base de Datos

Uno de los objetivos planteados con el cambio de servidor era migrar la base de datos que tenía la aplicación al nuevo alojamiento web y realizar sobre ella algunas modificaciones para que las nuevas funcionalidades puedan realizarse correctamente, así como implementar un sistema de monitorización de la aplicación.

Para realizar el proceso de migración, el primer paso fue acceder a la base de datos antigua por medio de `phpMyAdmin` entrando en el enlace `http://albergueweb1.uva.es/phpmyadmin` y poniendo el usuario y contraseña que nos había dado. Dentro de `phpMyAdmin` hay que seleccionar la base de datos, en nuestro caso `labcomuva` y dar a la pestaña de *Exportar*. Se nos mostrará dos opciones para seleccionar el método para realizar la exportación y dejaremos seleccionada la primera opción, la opción *Rápida*. Después damos al botón *Continuar* y se nos descargará la base de datos en formato `SQL`.

El archivo descargado lo pasamos a nuestro servidor remoto por medio de la herramienta Filezilla, en la que primero habrá que configurar la conexión a nuestro servidor indicando la dirección, el usuario, la contraseña y por último el protocolo de transferencia de datos SFTP. Una vez tenemos el archivo en nuestro servidor abrimos phpMyAdmin con la ruta localhost/phpMyAdmin en el navegador y accedemos mediante el usuario y contraseña establecidos con anterioridad.

Una vez dentro, nos dirigimos a la pestaña *Importar* y seleccionamos nuestro archivo .sql para proceder a la importación de la base de datos. El archivo sql contiene tanto la creación de las tablas como toda la información contenida hasta el momento en la anterior base de datos.

Cuando se crea una base de datos en lenguaje SQL, se especifica primero las tablas y atributos que contendrá y después en el caso de poseer esa información, se mete las filas o tuplas que se tengan. Es decir, meter los datos en la base de datos y esto se realiza en SQL por medio de inserts como se muestra en la Figura 24.

```
CREATE TABLE `becas` (
  `id` int(10) NOT NULL,
  `id_wordpress` int(10) NOT NULL,
  `titulo` text CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT NULL,
  `contenido` text CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT NULL,
  `fecha` text CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT NULL,
  `imagen` text CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Volcado de datos para la tabla `becas`
--

INSERT INTO `becas` (`id`, `id_wordpress`, `titulo`, `contenido`, `fecha`, `imagen`) VALUES
(0, 4257, 'Programa de movilidad SICUE', '<p style="text-align: justify;">¿Quieres realizar parte de tus estudios universitarios en',
(1, 4342, 'Becas Obra Social La Caixa para alumnos de postgrado', '<p style="text-align: justify;">Si has finalizado tus estudios de',
(2, 4451, 'Becas para proyectos periodísticos en el MIT', '<p style="text-align: justify;"><span style="color: #000000;">¿Quieres',
(3, 4647, 'Becas para estudios de grado, máster y doctorado en China', '<p style="text-align: justify;"><span style="color: #000000',
(4, 4911, 'Becas para realizar prácticas periodísticas en el Parlamento Europeo', '<p style="text-align: justify;"><span style="col',
(5, 4770, 'Programa Prometeo para el desarrollo de proyectos', '<p style="text-align: justify;"><span style="color: #000000;">Todc',
(6, 5160, 'Becas de investigación para estudiantes de Grado', '<p style="text-align: justify;"><span style="color: #000000;">La <s',
(7, 5335, 'Becas ARGO para realizar prácticas de empresa en el extranjero', '<p style="text-align: justify;"><span style="color: #0',
(8, 5438, 'Becas de formación en el Banco Mundial', '<p style="text-align: justify;"><span style="color: #000000;"><strong>El Banc',
(9, 5752, 'La UIMP oferta 14.000 becas para realizar cursos de inmersión en lengua inglesa', '<p style="text-align: justify;"><span',
(10, 5747, 'El British Council School ofrece una beca IELTS para cursar estudios en el extranjero', '<p style="text-align: justify;',
(11, 5817, 'La Fundación Villalar - Castilla y León convoca su VIII Edición de Becas de Investigación', '<p style="text-align: justif',
(12, 5832, 'Cinco becas para participar en el Festival Internacional de Periodismo de Perugia', '<p style="text-align: justify;"><span',
(13, 5907, 'SE ABRE LA CONVOCATORIA DE BECAS DE MOVILIDAD ENTRE UNIVERSIDADES DE ESPAÑA (SICUE)', '<p align="JUSTIFY"><span style=',
(14, 6031, 'Fulbright ofrece 25 becas para estudiar en EEUU durante el curso 2015/2016', '<p style="text-align: justify;"><span styl',
(15, 6353, 'Ya puedes solicitar la Beca ELLE 2015', '<p style="text-align: justify;"><span style="color: #000000;">Si estás intere',
INSERT INTO `becas` (`id`, `id_wordpress`, `titulo`, `contenido`, `fecha`, `imagen`) VALUES
(16, 6857, 'Plazo abierto para solicitar la beca de periodismo Carlos M. Castañeda', '<p style="text-align: justify;"><span style=',
(17, 7888, '1.500 becas para los Cursos de Verano 2015 de la Universidad Internacional Menéndez Pelayo', '<p style="text-align: just',
(18, 7992, 'Abierto el plazo para solicitar la ayuda económica de la beca Erasmus', '<p style="text-align: justify;"><span style="c'
```

Figura 24: Introducción de datos en la BD por medio de Inserts

Al final los *inserts* se tuvieron que borrar, debido a que el anterior servicio web, escrito en PHP, no funcionaba correctamente y no guardaba todas las noticias de las categorías que en realidad había en la página web. Esto se comentará con más detalle en el Capítulo 5 donde explicaremos la corrección de este error, pero lo destacado es que una vez importado se eliminó el contenido de las tablas para que nuestro proyecto Java de actualización de base de datos la rellenara automáticamente. Para borrar los *inserts* se fue tabla por tabla y abajo se selecciona “mostrar todo” y después abajo también le damos a “seleccionar todo” y al botón “borrar”.

Antes de proceder con el relleno automático de la base de datos importada, ésta necesitaba cambios en las tablas para que incorporaran dos nuevas columnas, ambas pueden verse en la Figura 25. La primera columna que se incluyó era la columna *link* en la que se encuentra el enlace a la noticia en la página *inform@UVa*. El enlace era necesario para la funcionalidad de compartir en redes sociales, y se obtiene su valor del propio JSON de respuesta a nuestra petición a la WP API Rest de donde obtenemos el resto de información de las noticias de Wordpress. El enlace luego es enviado como respuesta a la aplicación donde si el usuario quiere compartir la noticia, se comparte el enlace de la página web *inform@UVa*.

La segunda columna añadida a todas las tablas es el campo *visitas*, apartado importante para poder procesar la información de la monitorización llevada a cabo. Los valores de esta columna aumentan en uno cada vez que alguien entra en la noticia debido a que se realiza una llamada al Servicio Web implementado con el valor de su *id_wordpress*, valor único que poseen las noticias y ese servicio se encarga de sumar uno en la columna de visitas. De esta forma podemos tener una monitorización de los accesos a las noticias desde la aplicación y saber qué categorías y noticias son las más relevantes para los usuarios. También en el Capítulo 5 se detalla más el proceso llevado a cabo para la monitorización tanto desde la aplicación como desde el Servicio Web.

Tabla	id	id_wordpress	titulo	contenido	fecha	imagen	link	visitas
labcomuva cursos	int	int	text	text	text	text	text	int
labcomuva conferencias	int	int	text	text	text	text	text	int
labcomuva eventos	int	int	text	text	text	text	text	int
labcomuva becas	int	int	text	text	text	text	text	int
labcomuva congresos	int	int	text	text	text	text	text	int
labcomuva practicas	int	int	text	text	text	text	text	int

Figura 25: Esquema de la Base de Datos actual

Para crear estas nuevas columnas nos dirigimos a las tablas, desplegamos la sección de “Columnas” y seleccionamos “Nueva”, como se aprecia en la Figura 26 en la que ya se ve las nuevas columnas creadas en nuestra base de datos. Se nos mostrará a la derecha un formulario donde detallaremos los ajustes propios de la nueva columna, tales como tipo de dato, si puede ser o no *nulo* o el valor inicial que tendrá al crearse.

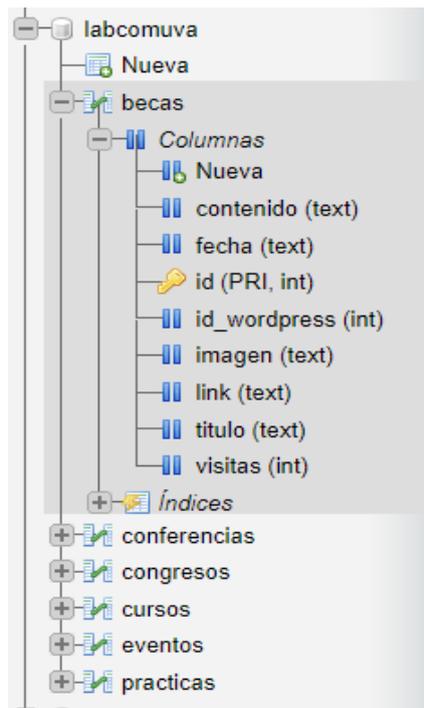


Figura 26: Desglose de las tablas y columnas de la Base de Datos

4.5.5 Copias de seguridad Base de Datos

Una de las acciones a realizar en ese TFG era la creación de copias de seguridad de la base de datos de manera totalmente automática cada cierto periodo de tiempo para darle seguridad a los datos guardados en caso de pérdida. Con este objetivo se creó un script que realiza la copia de seguridad de la base de datos cada 15 días, ya que nos pareció el rango de tiempo más razonable para este objetivo. Para ello usamos el siguiente comando dentro de un script (Figura 27):

```
#!/bin/bash
mysqldump -h localhost -u root -p0pclbpJ2021 labcomuva > /home/noemer/Escritorio/CopiasBaseDatos/labcomuva_$(date +%Y-%m-%d).sql
~
~
```

Figura 27: Script que genera en el directorio seleccionado una copia de la Base de Datos fechada

Como se observa en la Figura 22, el comando *mysqldump* se encarga de hacer la copia de seguridad dando como parámetros la dirección, el usuario MySQL, su contraseña y el nombre de la base de datos de la que queremos hacer la copia. Además, le tenemos que indicar en qué directorio generar esas copias de seguridad y mediante el comando *\$(date +%Y-%m-%d).sql* le añadimos al nombre del fichero generado la fecha y el tipo de fichero, en nuestro caso SQL.

Una vez creado el script que genera copias de la base de datos, nos falta que esta tarea se ejecute de forma autónoma cada 15 días. Para ello, hacemos uso de Cron [28] que es un proceso que se ejecuta en segundo plano desde el arranque del sistema y comprueba si tiene alguna tarea

que tiene que ser ejecutada en una fecha y hora especificada y con la periodicidad que tenga impuesta. Esto como nos permite ejecutar el script con la periodicidad que queramos lo establecimos para que las copias se realizarán los días 1 y 15 de cada mes.

Para introducir la nueva tarea en Cron, lo primero que tenemos que hacer es pasar a usuario root, aunque también es posible hacerlo para otros usuarios, pero en nuestro caso usamos el usuario raíz. Después se ejecuta el comando:

```
crontab -e
```

Y se abrirá el fichero donde se escriben las tareas que se deben ejecutar en Cron. En el final del fichero tenemos que incluir la nueva tarea. Para incluir la tarea que queríamos hay que escribir en el fichero Cron lo siguiente:

```
0 7 1,15 * * /home/noemer/Esitorio/CopiasBaseDatos/copyDB.sh
```

El primer número indica los minutos, el segundo las horas, el tercero indica los días del mes, el quinto son los meses del año y el último es para indicar los días de la semana. Como se puede apreciar la copia de la base de datos se genera a las 7 de la mañana los días 1 y 15 de cada mes. Los asteriscos puestos en meses y días de la semana son para indicar que no especificamos el valor. Después de esto solo queda poner la ruta donde se encuentra el script que queremos ejecutar. Tal y como se observa, se ha colocado un directorio en el escritorio del servidor titulado “CopiasBaseDatos” donde se irán almacenando dichas copias.

En caso de querer ver solo qué tareas hay implementadas en Cron se puede escribir el siguiente comando:

```
crontab -l
```

4.5.6 Instalación y Configuración Apache Tomcat

El último paso de la configuración del servidor remoto es la instalación de Apache Tomcat para poder desplegar el servicio web. Como requisitos previos es necesario tener instalado el JDK que como se detalla en anteriores puntos de este capítulo ya estaba instalado y configurado. Por tanto, procedemos a explicar los pasos a seguir para la instalación de Tomcat en el servidor. Para ello se siguen los siguientes pasos:

1. El primer paso es descargarse el Apache Tomcat, que en nuestro caso nos decantamos por el Apache Tomcat 7 con el que había trabajado con anterioridad por ejemplo en la carpeta Descargas:

```
cd /home/noemer/Descargas
```

```
wget http://apache.rediris.es/tomcat/tomcat-7/v7.0.35/bin/apache-tomcat-7.0.35.tar.gz
```

2. El siguiente paso es descomprimir el paquete descargado con el comando *tar* y seguidamente mover esa carpeta a su localización final en */usr/local/*:

```
tar xvzf apache-tomcat-7.0.35.tar.gz
```

```
mv apache-tomcat-7.0.35 /usr/local/
```

3. A continuación, se crea un enlace simbólico de la siguiente manera:

```
ln -s /usr/local/apache-tomcat-7.0.35 /usr/local/tomcat
```

4. Para hacer la administración de Apache Tomcat más sencilla añadimos al *path* el script catalina. También añadimos catalina al *runlevel* siguiendo la siguiente secuencia de comandos:

```
ln -s /usr/local/tomcat/bin/catalina.sh /usr/local/cantina
```

```
ln -s /usr/sbin/catalina /etc/rc1.d/K99catalina
```

```
ln -s /usr/sbin/catalina /etc/rc2.d/S99catalina
```

5. El siguiente paso es crear un usuario con el que poder acceder al *Manager* de Tomcat. Para hacer esto accedemos con el siguiente comando al fichero a modificar:

```
vi /usr/local/tomcat/conf/tomcat-users.xml
```

Y dentro de este fichero, justo antes del final creamos tres roles que son necesarios como *manager-gui* para hacer uso de la interfaz HTML del Manager de Tomcat. También añadiremos el rol de *admin* y el rol *manager* de la siguiente manera:

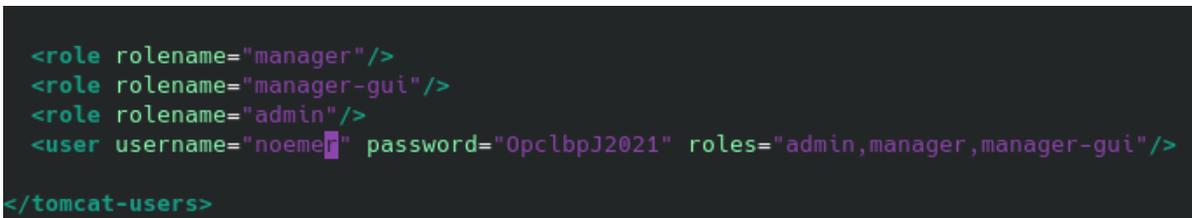
```
<role rolename="nombreRol"/>
```

Aparte de la creación de esos roles se necesita crear un usuario a quien asignar dichos roles. Al crear el usuario se le dota de un nombre de usuario, una contraseña y unos

roles, donde introduciremos el nombre de los creados anteriormente separados por comas, siguiendo el siguiente comando:

```
<user username="nombreUsuario" password="contraseña" roles="roles"/>
```

El fichero debería quedar tal y como se ve en la imagen de la Figura 28.



```
<role rolename="manager"/>
<role rolename="manager-gui"/>
<role rolename="admin"/>
<user username="noeme" password="0pclbpJ2021" roles="admin,manager,manager-gui"/>
</tomcat-users>
```

Figura 28: Creación de roles y usuario para Apache Tomcat

6. Finalmente, creamos un script al *init.d* con el objetivo de hacer más sencillo el arranque y el parar el servidor. El script lo llamaremos Tomcat y escribiremos lo siguiente:

```
#!/bin/bash
### BEGIN INIT INFO
# Provides: tomcat7
# Required-Start: $network
# Required-Stop: $network
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start/Stop Tomcat server
### END INIT INFO
PATH=/sbin:/bin:/usr/sbin:/usr/bin
start() {
sh /var/opt/tomcat/bin/startup.sh
} stop() {
sh /var/opt/tomcat/bin/shutdown.sh
} case $1 in
start) start;;
stop) stop;;
restart) stop; start;;
*) echo "Run as $0 "; exit 1;;
esac
```

Para finalizar el proceso comprobamos que funciona iniciando Tomcat de la siguiente manera:

```
/etc/init.d/tomcat start
```

Una vez realizados todos estos pasos, ya podremos acceder desde el navegador al Tomcat introduciendo la dirección en la url <http://157.88.130.185:8080>, tal y como se muestra en la Figura 29.

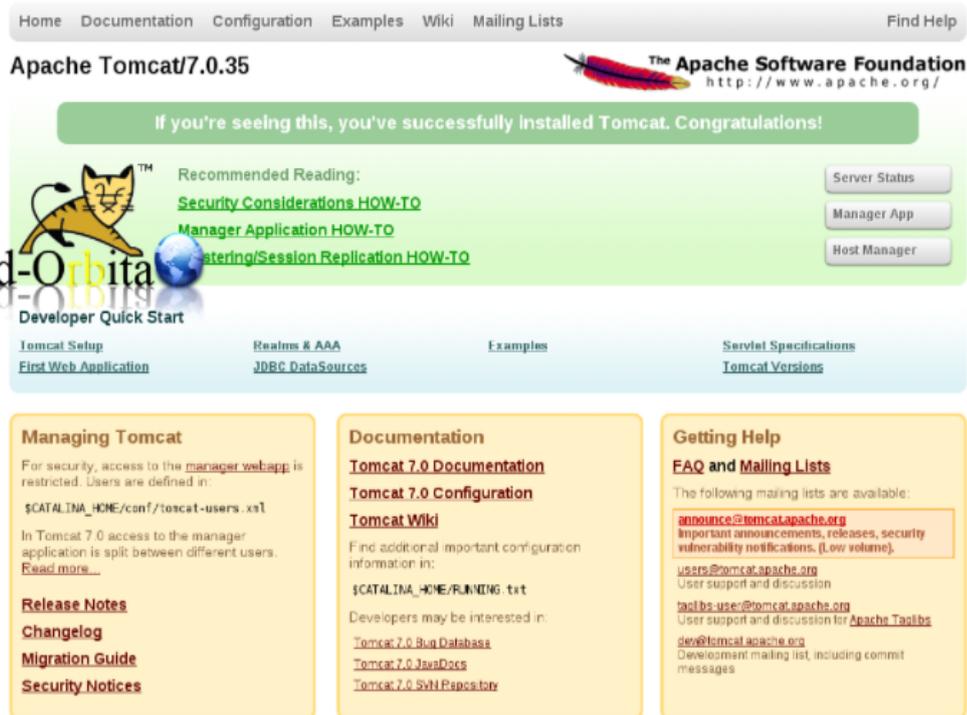


Figura 29: Página principal de Apache Tomcat

Finalmente, solo nos queda desplegar el servicio web y para ello accedemos al *Tomcat Web Application Manager* de Tomcat pulsando sobre el botón *Manager App*. En este momento, se nos abrirá una ventana donde veremos las aplicaciones desplegadas en ese momento en el servidor Tomcat y si seguimos bajando encontramos un apartado titulado “*WAR file to deploy*”. En este apartado, presionamos sobre el botón “*Choose File*” y buscamos dentro de nuestro servidor el fichero War generado de nuestro servicio web. Y para terminar damos al botón “*Deploy*” y nuestro servicio web, si no hay ningún inconveniente, se desplegará en nuestro servidor y estará lista para recibir las peticiones correspondientes (Figura 30).

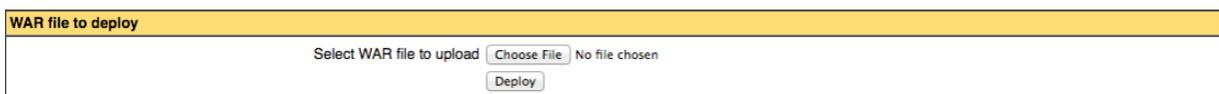


Figura 30: Apartado donde seleccionar y desplegar aplicación en formato War

Aparte, también podremos ver nuestra aplicación en la lista anteriormente comentada de servicios desplegados dentro del Tomcat, junto con una serie de opciones como parar el servicio o quitar el despliegue, además del path principal del servicio.

4.6 Pruebas

Número	Título de Prueba	Resumen	Pasos	Resultado
1	Prueba de Java	Comprobamos si esta instalado y configurado java en el servidor	1-Accedemos a la terminal 2-Escribimos el comando "java -version"	Se nos muestra la versión de java que instalamos, demostrando la correcta configuración de Java
2	Prueba de MySQL	Comprobamos al correcta instalación y configuración de MySQL	1-Accedemos a la terminal. 2-Escribimos el comando "mysql-version"	Se nos muestra la versión de MySQL y por tanto la instalación fue correcta
3	Prueba de servidor Tomcat 7	Comprobamos al correcta instalación y configuración de Tomcat 7	1-Accedemos a la dirección web en el puerto 2-Nos pide usuario y contraseña 3- Accedemos Manager App	Nos deja acceder al "Tomcat Web Application Manager" y por tanto esta bien configurado el Tomcat 7
4	Prueba de phpMyAdmin	Comprobamos al correcta instalación y configuración de phpMyAdmin	1-Accediendo a la dirección web firmada por la ip del servidor y el path /phpmyadmin se abre la página de phpMyAdmin 2-Ponemos el usuario y contraseña	Podemos acceder a phpMyAdmin satisfactoriamente

Tabla 3: Pruebas de las configuraciones e instalaciones realizadas en el servidor

4.7 Conclusiones

Al principio de la planificación del proyecto no contábamos con tener que cambiar de servidor, pero fue algo más que necesario. En el anterior alojamiento no podíamos realizar muchas de las cosas que se plantearon al comienzo, como la instalación de los nuevos servicios web escritos en lenguaje Java que teníamos que desarrollar. Pero este cambio en el rumbo del Trabajo de Fin de ha sido beneficioso para el proyecto, gracias a ello en lo personal he aprendido a configurar muchas herramientas importantes como es la instalación de un servidor Tomcat, la instalación de MySQL y de un administrador para las bases de datos como phpMyAdmin y demás configuraciones realizadas. Y para el proyecto también es beneficioso, ya que ahora la aplicación tiene un alojamiento propio con más permisos y posibilidades para cambiar los servicios web o las bases de datos que se creen con más facilidad y comodidad.

Para este trabajo de migración a un nuevo servidor y la configuración de éste, realicé un proceso de investigación, además del seguimiento de múltiples guías para hacer funcionar las herramientas que necesitábamos en un servidor Debian 10 como es nuestro servidor.

Como se comenta en el siguiente capítulo, desde el nuevo alojamiento web también desarrollamos la mayor parte del servicio web que conecta la aplicación con la base de datos. Para ello se instalaron una serie de programas para elaborar en ellos el servicio web, aunque finalmente se utilizó la herramienta Spring Tool Suit.

5 Implementación de los Servicios Web en Java

5.1 Introducción

En este capítulo trataremos todo lo referente al desarrollo de los servicios web para la aplicación. El objetivo principal que se propuso era la sustitución de los actuales servicios web de la aplicación desarrollados en PHP por unos servicios web en un lenguaje menos obsoleto que proporcionara mejores prestaciones y seguridad a la aplicación. Este capítulo está dedicado a mostrar todo el proceso seguido para la creación de estos servicios y para ello comenzaremos viendo en el análisis como se implementaba anteriormente los servicios en la aplicación y los requisitos nuevos del proyecto. Después, veremos el diseño tanto del servicio web como los programas java para después explicar de forma más detallada la implementación de los requisitos. También se incluirá todo lo referente al desarrollo del sistema de monitorización de la aplicación, objetivo principal dentro de este TFG y que abarca modificaciones tanto en la aplicación, como en los servicios web y en la base de datos. Por último, se mostrará una tabla de las pruebas realizadas y las conclusiones.

5.2 Análisis

Los requisitos funcionales y no funcionales que se tratarán en este capítulo son los siguientes:

1. RF-1. Creación de un nuevo servicio web para la aplicación que sustituyan a los anteriormente utilizados escritos en PHP. El servicio web será una aplicación Rest cuyos servicios conectan al cliente Android con la base de datos alojada en el mismo servidor remoto que el servicio web.
2. RF-2. El nuevo servicio web contendrá unos servicios que permitan obtener la lista de noticias de cada categoría y las noticias del slider que se encuentran almacenadas en la base de datos.
3. RF-3. EL nuevo servicio web contendrá unos servicios que permitan la monitorización de la aplicación, guardando en la base de datos el número de accesos de cada noticia.

4. RF-5. Desarrollar un programa encargado de la obtención de las noticias de la página web inform@Uva (<http://www.informauva.com/>) de las categorías que aparecen en la aplicación para guardarlas en la base de datos de forma automática y diaria.
5. RF-6. Desarrollar un programa cuyo objetivo es la actualización de toda la base de datos para casos particulares, como la necesidad de editar una noticia ya publicada.
6. RF-7. Creación de un script que ejecute de manera diaria la aplicación encargada de actualizar la base de datos.
7. RF-8. Desarrollo de un sistema de monitorización de la aplicación que nos permita obtener el número de visualizaciones de cada noticia.

bd_becas.php	1.377	Archivo de...	06/10/2019 10:...	-rw-----	55194 55194
bd_conferencias.php	1.412	Archivo de...	10/07/2019 17:...	-rw-----	55194 55194
bd_congresos.php	1.397	Archivo de...	10/07/2019 17:...	-rw-----	55194 55194
bd_cursos.php	1.383	Archivo de...	10/07/2019 17:...	-rw-----	55194 55194
bd_eventos.php	1.387	Archivo de...	10/07/2019 17:...	-rw-----	55194 55194
bd_practicas.php	1.397	Archivo de...	10/07/2019 17:...	-rw-----	55194 55194
bd_slider.php	5.455	Archivo de...	01/12/2019 17:...	-rw-----	55194 55194
escribir_bd_becas.php	6.467	Archivo de...	31/10/2019 20:...	-rw-----	55194 55194
escribir_bd_conferenc...	6.385	Archivo de...	31/10/2019 20:...	-rw-----	55194 55194
escribir_bd_congresos...	6.508	Archivo de...	31/10/2019 20:...	-rw-----	55194 55194
escribir_bd_cursos.php	6.312	Archivo de...	31/10/2019 20:...	-rw-----	55194 55194
escribir_bd_eventos.p...	6.425	Archivo de...	03/11/2019 9:4...	-rw-----	55194 55194
escribir_bd_practicas...	6.327	Archivo de...	31/10/2019 20:...	-rw-----	55194 55194

Figura 31: Servicios web empleados en la anterior versión de UVaNOW

En la Figura 31 se muestra en el antiguo alojamiento web el directorio donde se encontraban los programas PHP llamados por nuestra aplicación y también los que se ejecutaban de una manera manual para rellenar la base de datos (con prefijo la palabra “escribir”).

5.3 Diseño

En la primera versión de UVaNow, el trabajo de traer las noticias desde la base de datos a la aplicación era desempeñado por siete programas PHP, como anteriormente se comentaba, pero para este nuevo proyecto se decidió que fuera sustituido por un único proyecto Java que desempeñará las mismas funciones de forma integrada. Esto es debido a las posibilidades que nos ofrece este lenguaje de programación, la mejores prestaciones y seguridad que proporciona.

Para el desarrollo de este proyecto y aprovechando la posibilidad de programar desde dentro del propio servidor remoto, instalamos el programa Spring Tool Suite en nuestro alojamiento web. Spring Tool Suite es una herramienta basada en eclipse diseñada para la programación con el framework Spring que será usado para este desarrollo. Esto nos permitía de una manera más cómoda probar la aplicación en el servidor Apache Tomcat 7 que tenemos instalado allí.

El servicio web desarrollado es una API Rest alojada en nuestro servidor remoto, desplegada en el Tomcat y que responde a ciertas peticiones lanzadas por la aplicación devolviendo un JSON con las noticias que se piden en la petición o también aumentando el valor de visitas de una noticia como parte del sistema de monitorización implementado en nuestra aplicación. Toda la implementación se explica con más detalle en los siguientes apartados del capítulo, ya que desglosamos el proceso llevado a cabo para el desarrollo de los servicios web y se especificarán cada uno de los métodos implementados que se encargan de responder a las diferentes peticiones lanzadas desde UVaNow. En la Figura 32 se muestran los servicios proporcionados por la aplicación Rest creada.

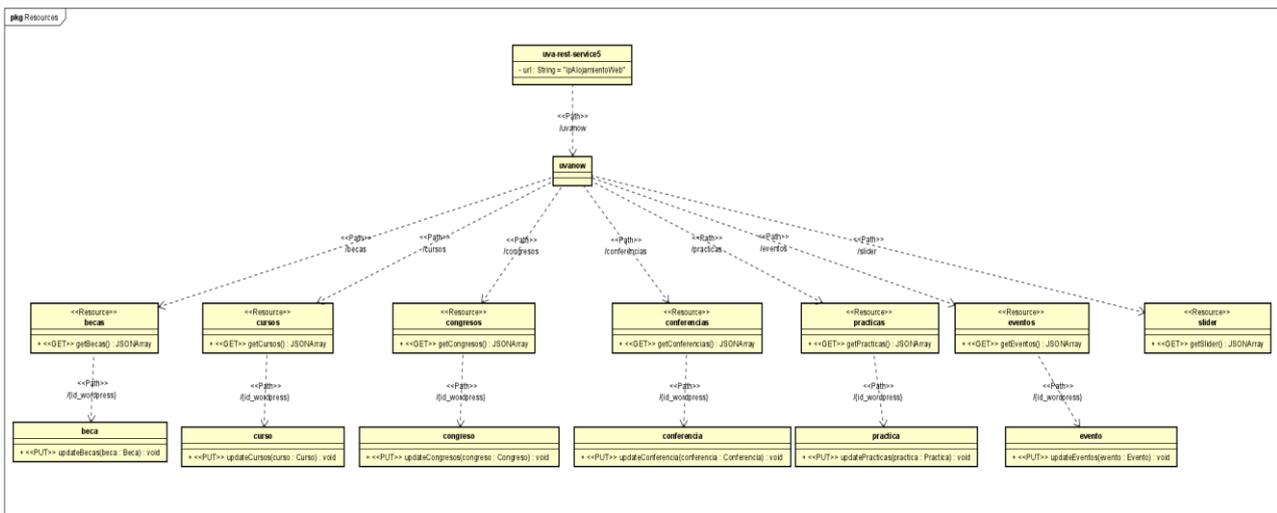


Figura 32: Diagrama de los recursos del servicio web desarrollado

Por otra parte, otros elementos que tenían que modificarse eran un conjunto de programas previos desarrollados en lenguaje PHP que se encargaban de rellenar la base de datos intermedia mediante peticiones a la WP API Rest de Wordpress, y de este modo sustituirlos por programas en Java que realizaran esa misma función. Así pues, se decidió implementar dos nuevos programas en Java para sustituir a los seis existentes programados PHP. El primer programa es el encargado de la actualización diaria de la base de datos con las nuevas noticias subidas por los alumnos de Periodismo, ya que se ejecuta mediante un script programado para lanzarse todos los días mediante la herramienta de linux Cron. El segundo programa ha sido desarrollado para casos excepcionales en los que se requiera la actualización del contenido, imagen, título u otro elemento de alguna noticia. Esto se debió a que el procesado de las peticiones y las respuestas, sumado al desgranado de los JSON para obtener los valores que necesita la aplicación no es un proceso muy ligero y por tanto una vez la base de datos está rellena, la actualización diaria añadiendo solamente las últimas noticias aligera mucho dicho proceso de actualización. Si por alguna circunstancia se requiere cambiar una noticia o varias noticias ya existentes, bastaría con que se cambiasen en la web inform@UVa (mediante WordPress) y

después se ejecutaría este segundo programa. Al acabar el proceso de cierta duración, todas las noticias habían sido actualizadas en la base de datos.

5.4 Implementación

5.4.1 Servicio Web que conecta con la Aplicación Android

Este apartado está dedicado a explicar detalladamente el diseño y creación del servicio web encargado de recibir las peticiones de la aplicación de UVaNOW y proporcionar una respuesta o un efecto sobre la base de datos alojada en nuestro servidor web. En primer lugar, se expondremos cómo eran los anteriores servicios web, seguido de cuales han sido las herramientas y tecnologías empleadas. Más adelante, se describe el funcionamiento de la aplicación y de todos sus métodos, para terminar, describiendo cómo se desplegó en el servidor.

5.4.1.1 Herramientas y tecnologías usadas en el desarrollo del servicio web

El desarrollo del Servicio Web se realizó desde el propio servidor remoto, ya que tenemos acceso gráfico directo al servidor y se pueden instalar todas las herramientas necesarias, lo que nos permite desplegar y probar la aplicación directamente de manera muy sencilla. Tal y como se estableció como objetivo del Trabajo de Fin de Grado, el Servicio Web se ha elaborado en el lenguaje de programación Java. Este objetivo se propuso debido a que el lenguaje PHP está cada vez más en desuso y Java ofrece muchas más posibilidades en todos los aspectos y sobre todo en el ámbito de la seguridad.

El programa inicial que se propuso para el desarrollo de los Servicio Web fue Netbeans, debido a que es la herramienta usada en la asignatura del Grado “Desarrollo Basado en Componentes y Servicios”. Así pues, se empezó desarrollando la aplicación en Netbeans, pero por problemas a la hora de desplegar la aplicación y ante la imposibilidad de encontrar la solución y al mismo tiempo cumplir con los plazos establecidos, se decidió por consejo de mis tutores probar el framework Spring Boot, y para ello cambié la plataforma de desarrollo a Spring Tool Suite. Esta herramienta es un IDE basado en Java EE de Eclipse muy personalizado para el uso del framework Spring. A partir de dicho despliegue, se volvió a hacer la aplicación reutilizando todo lo que servía de la parte desarrollada en Netbeans y de esta forma programar y desplegar todo el servicio web correctamente.

Este framework fue elegido por su gran versatilidad para hacer diferentes programas ya sea servicios web o microservicios. Y dentro de este framework está Spring Boot que nos permite gracias a sus múltiples automatismos desarrollar una aplicación y desplegarla de manera rápida y sencilla. Además, es uno de los frameworks cada vez más utilizados para el desarrollo de aplicaciones. Lo que hice en términos generales fue un CRUD (*Create, Read, Update and*

Delete) con SpringBoot y Java Persistence API o por sus siglas JPA que es la API de persistencia de la plataforma Java EE. JPA facilita el acceso a la base de datos ya que será el JPA de Spring el encargado de gestionar la comunicación con la capa de datos. A esta abstracción se le denomina DAO o Data Access Object. En definitiva, JPA es un conjunto de clases y métodos que permite comunicarnos con la capa de persistencia. Hay que aclarar que un servicio CRUD se usa para referirse a las funciones básicas de la capa de persistencia de una aplicación que son creación, lectura, actualización de valores y borrado.

Otro elemento que destacar fue el uso de Maven [29] para la obtención de ciertas dependencias necesarias para que funcionara bien. Maven es una herramienta software que sirve para gestionar y elaborar proyectos Java, similar a Ant pero más simple y basado en el formato XML.

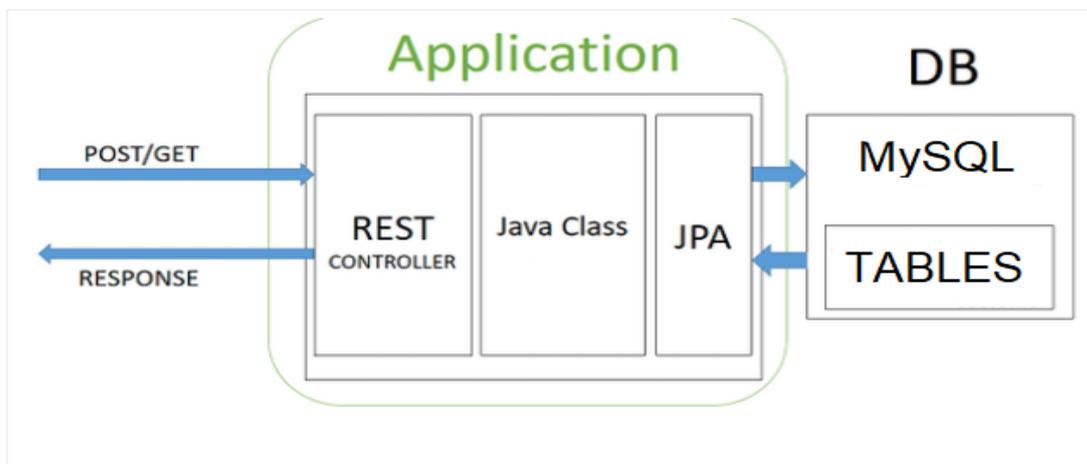


Figura 33: Esquema de funcionamiento de la aplicación Rest

En la Figura 33 se presenta un esquema del funcionamiento de la aplicación Rest, donde se observa cómo le llegan peticiones desde la aplicación UVaNOW. Luego dentro de la aplicación tenemos lo referente al controller, las entidades y las clases de conexión con la base de datos. Y vemos cómo se comunica con la base de datos y responde las peticiones. En nuestro caso las peticiones que realizamos son GET y PUT por lo que los accesos a la base de datos serán solo de lectura y actualización de datos. Las peticiones GET para obtener las noticias de la base de datos y las peticiones de tipo PUT se usan en un método encargado de actualizar una noticia aumentando en uno el número de visitas.

5.4.1.2 Descripción del funcionamiento y métodos del servicio web

En primer lugar, se describirán los servicios web previos (desarrollados en PHP) para basarnos en cómo obtienen los elementos de la Base de Datos. Para el primer prototipo de UVaNOW se empleaban siete servicios web para obtener las noticias de la base de datos, seis para cada una de las categorías de la aplicación (tipos de noticias categorizadas) y una adicional para las noticias del slider. Los servicios web de las categorías se llamaban al acceder a estas y eran bastante sencillos. En primer lugar, se conectan a la base de datos anterior (antes de la migración al nuevo servidor), y desde allí obtenía de la consulta *SELECT * FROM*

nombreTabla todos los elementos de la tabla de la categoría en cuestión. Una vez tomados esos datos seleccionaba para el JSON de respuesta los datos importantes en ese momento. Estos datos eran el id, el título, contenido, fecha e imagen de cada noticia formando un Array de JSON con estos datos de cada noticia. Por otro lado, el Servicio Web encargado de responder con las noticias destinadas a formar parte del Slider era algo más complejo, básicamente hacia lo mismo que para obtener todas las noticias de una categoría, pero solo cogiendo las tres últimas añadidas y de todas las categorías. Luego se creaba el array de JSON y se respondía a la petición.

Para el nuevo Servicio Web se optó por un único proyecto que realizara todas esas funciones y alguna más mediante una aplicación API Rest que responda dependiendo de la llamada concreta. Como comentamos en anteriores apartados se realizó un CRUD Rest utilizando Spring Boot y JPA para desarrollar la aplicación. El proceso no fue tan directo, es decir, no programamos y realizamos las pruebas directamente como queríamos hacerlo, debido a la espera que tuvimos que hacer hasta tener nuestro nuevo alojamiento web. Con el objetivo de avanzar lo máximo posible, se desarrolló un proyecto de prueba en el que ir programando la forma en que la aplicación cogería los datos de la base de datos y los enviaría en formato JSON para que la aplicación pudiera manejarlos. Y este proyecto de prueba se hizo en Netbeans y utilizando una base de datos local que simulaba la original. Al final este proyecto en Netbeans avanzó lo suficiente para ser un proyecto final y fue desplegado en el servidor para comprobar su funcionamiento. Debido a los múltiples problemas que surgieron con su despliegue y funcionamiento se tomó la decisión de volver a hacerlo usando Spring Boot.

Así pues, se instaló Spring Tool Suit como framework en que desarrollar la aplicación dentro del servidor remoto y creamos el proyecto Spring que utilizará las siguientes dependencias propias del framework:

- Spring-boots-started-web: Para crear aplicaciones web como una aplicación Rest como es nuestro caso y utiliza Spring MVC.
- Spring-boot-started-data-jpa: Esta dependencia proporciona la posibilidad de conectar la aplicación con la base de datos de manera más eficiente por medio del uso de JPA.
- MySql-connector-java: Es un controlador JDBC de tipo 4 (Driver protocolo nativo de Java) desarrollado como conector JDB para uso en servidores de base de datos tanto MariaDB [30] como MySQL como la que usamos en nuestra aplicación.

Una vez creado el proyecto Spring se debe configurar el acceso a la base de datos mediante el archivo `Application.properties`. Este archivo, propio de las aplicaciones Spring, se utiliza para mantener unas propiedades en un solo archivo y poder ejecutar la aplicación en diferentes entornos.

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/labcomuva?autoReconnect=true&useSSL=false&useUnicode=yes&characterEncoding=UTF-8
2 spring.datasource.username=root
3 spring.datasource.password=0pClbpJ2021
4 spring.datasource.driver-class-name: com.mysql.jdbc.Driver
5 spring.jpa.hibernate.ddl-auto=none
6 spring.datasource.tomcat.connection-properties=useUnicode=true;characterEncoding=utf-8;
```

Figura 34: Propiedades establecidas en el archivo Application.properties

En la Figura 34 se muestran las propiedades establecidas en la aplicación en el archivo Application.properties que pasaremos a explicar:

- `Spring.datasource.url`: en esta propiedad indicamos la dirección URL de la base de datos, además de especificar el Encode, es decir, la codificación de los caracteres y también indicar la propiedad `useSSL` como falsa como en numerosas páginas se recomendaba por problemas que podría generar al conectar a la base de datos.
- `Spring.datasource.username`: aquí se especifica el nombre de usuario para conectarnos a la base de datos.
- `Spring.datasource.password`: propiedad que establece la contraseña para poder acceder a la base de datos.
- `Spring.datasource.driver-class-name`: indicamos el driver utilizado para la base de datos. En nuestro caso, al ser una base de datos MySQL y haber usado el driver JDBC, establecimos el valor `com.mysql.jdbc.Driver` para ello.
- `Spring.jpa.hibernate.ddl-auto`: esta propiedad específica de Spring Data JPA que controla el comportamiento de la base de datos según el valor que le establezcamos. Por ejemplo, si es `update` intentará agregar nuevas columnas o restricciones después de consultar la API del controlador JDBC, pero nunca eliminará columnas, restricciones, etc. En caso de usar `create-drop`, cuyo uso es más típico en para casos de pruebas, los datos se eliminan del esquema dejándola vacía después de ejecutar las pruebas. Para esta aplicación establecimos el valor `none` que básicamente es no especificar esta propiedad porque no era necesaria, también por ser lo más recomendado y lo que menos afecta a la base de datos.

Ahora procederemos a explicar el servicio web, tanto su estructura como su funcionamiento de manera detallada apoyándonos en imágenes del código y del esquema de paquetes de la aplicación.



Figura 35: Estructura de la aplicación y package Core

Como se observa en la Figura 35, tenemos el proyecto abierto dentro de Spring Tool Suit y se muestra dentro de la ruta src/main/java el primer paquete llamado com.example.core, que contiene el *Main* de la aplicación. Se aprecian dos clases, la primera, Application que contiene el *Main* de la aplicación y la clase ServlerInicializer que sirve para poder empaquetar nuestra aplicación como un archivo de extensión *War*.

```
package com.example.core;

import org.springframework.beans.factory.annotation.Autowired;

@SpringBootApplication
public class Application {

    @Autowired
    BecasRepository repositorio;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Figura 36: Clase Application, clase Main del proyecto Java

Toda aplicación tiene su método *Main* para ejecutarse, en el caso de las aplicaciones con el framework Spring se llama al método run de la clase *SpringApplication*. Este framework se basa en principalmente en las anotaciones es decir en palabras claves que empiezan por un @, en el caso del *Main* de la aplicación, necesita una serie de anotaciones que indiquen las funciones que tiene esta clase. Estas anotaciones son @Configuration, que indica que la clase contiene la configuración principal del servicio web, @EnableAutoConfiguration que indica que se aplicará la configuración automática del starter utilizado y @ComponentScan que se necesita para localizar los elementos etiquetados cuando se necesite. Todas estas anotaciones se pueden sustituir por @SpringBootApplication que realiza las funciones de las tres anteriores nombradas y que se puede ver en la Figura 36.

Los siguientes conjuntos de clases que vemos son los controladores y las implementaciones de éstos. Como podemos observar en la Figura 37, hay un controlador y una implementación por cada categoría de noticia y una más para el slider. Esto se hizo para una mayor comodidad a la hora de realizar posibles cambios en el futuro en determinadas categorías o en el slider así como para ser más sencillo usar la conexión a la base de datos por medio de JPA.



Figura 37: Paquetes Controllers y ControllersImpl

Las clases dentro de Controllers son básicamente interfaces para cada una de las 6 categorías de noticias más una para el slider.

```
package com.example.core.controllers;

import java.util.List;

public interface BecasController {

    public List<Becas> getBecas();

    // public Optional<Becas> getBecasById(Long id);
    //
    // public Becas addBecas(Becas becas);
    //
    // public String deleteBecas(Long id);

    public String updateBecas(int id_becasNew);

}
```

Figura 38: Interfaz del controlador Becas del paquete Controllers

En la Figura 38 observamos que los métodos de la interfaz corresponden a 2 de las funciones CRUD explicadas anteriormente. Concretamente es una llamada de solo lectura, lo que dentro de un servicio Rest es una llamada de tipo GET y luego tenemos lo que sería una llamada de tipo PUT en la que se modifican datos de la base de datos sin borrar ni incluir nuevas tuplas. El

primer método, el método GET, corresponde en este caso a la obtención de todas las noticias de la base de datos de la tabla Becas. El segundo método realiza la función de monitorización, actualizando un atributo de la noticia en la base de datos cada vez que accede un usuario a su lectura. Se profundizará más en esto mientras seguimos avanzando con la explicación de la implementación de los controladores, de servicios y accesos a la base de datos. A modo de resumen, las clases del paquete `Controllers` definen los controladores que se implementan en el paquete `controllersImpl`. En el caso del slider solo hay un método de tipo GET para traer las noticias que representa.

```
package com.example.core.controllersImpl;

import java.util.List;

@RestController
@RequestMapping(value = "/uivanow")
public class BecasControllerImpl implements BecasController {

    @Autowired
    BecasService becasService;

    // http://localhost:8080/becas (GET)
    @RequestMapping(value = "/becas", method = RequestMethod.GET, produces = "application/json")
    @Override
    public List<Becas> getBecas() {
        return becasService.findAllBecas();
    }

    @RequestMapping(value = "/becas/{id_wp}", method = RequestMethod.PUT)
    @Override
    public String updateBecas(@PathVariable(value="id_wp") int id_wp) {

        return becasService.updateBecas(id_wp);
    }
}
```

Figura 39: Clase que implementa interfaz del controlador Becas

La implementación del controlador nos permite realizar los *ends points* (URLs de un servicio que responde a las peticiones) para responder a las peticiones, y tal como se muestra en la Figura 39 se observa la implementación de los métodos de la interfaz. El primer punto a destacar dentro del código son las anotaciones de la propia clase. La primera anotación es `@RestController`, que es una anotación que facilita la creación de servicios Rest ya que combina las anotaciones `@Controller` y `ResponseBody` lo que elimina la necesidad de anotar cada método que maneje solicitudes con la anotación `@ResponseBody`. Después tenemos una anotación muy común en una aplicación Rest como es `@RequestMapping` que sirve para establecer la ruta URL a los diferentes servicios aportados por la Aplicación Rest. Lo siguiente que se aprecia en el código es la anotación `@Autowired` sobre una variable de tipo `BecasService`. `@Autowired` es una anotación muy habitual cuando se trabaja con Spring Framework ya que permite “conectar” elementos dentro del proyecto, es decir, nos permite inyectar unas dependencias con otras.

Los métodos que se implementan corresponden a una llamada GET y a una llamada PUT, cuyo *path* dentro de la aplicación es el nombre de la categoría de noticia o es */slider* según el servicio al que se quiera llamar, y se establece a la vez que el tipo de *Request* (Petición) del método y el tipo de respuesta proporcionada, dentro de la anotación *@RequestMapping*. Hay que comentar que la anotación *@Override* es propia del lenguaje Java e indica que sobrescribe o implementa un método de una clase de la que extiende o de una interfaz. El primer método, es un *get* que nos responde con la lista de noticias de la categoría seleccionada en formato JSON. Para ello, realiza la llamada al método *findAll* de la interfaz de servicios propia de la categoría de noticia. El segundo método, el *update*, recibe desde la URL el *id_wordpress* de la noticia, llama al método *update* de los servicios para actualizar un atributo de la noticia indicada, en concreto el atributo *visitas*, que aumentará en uno su valor para así realizar la monitorización de la aplicación.

La implementación del controlador del slider es diferente al descrito anteriormente debido a que consiste en traer las tres últimas noticias de cada una de las categorías. Por ello se realiza en el método *get* que se implementa el mismo proceso que para traer todas las noticias de cada categoría (Figura 40) y posteriormente se seleccionan las tres últimas. Se diseñó el servicio de forma que se diferencian las noticias según su categoría así si en un futuro requiere cambios en alguna categoría en concreto serán más sencillos de realizarse, a parte de que de este modo es más sencilla la conexión con la base de datos via JPA. Por este motivo, en el caso especial del slider transformamos las últimas 3 noticias de todas las categorías a noticias de tipo *Becas* con el objetivo de hacer de una forma más simple la respuesta con las noticias que corresponden al slider (Figura 41).

```
@Autowired
BecasService becasService;

@Autowired
CongresosService congresosService;

@Autowired
ConferenciasService conferenciaService;

@Autowired
CursosService cursosService;

@Autowired
EventosService eventosService;

@Autowired
PracticasService practicasService;

@RequestMapping(value = "/slider", method = RequestMethod.GET, produces = "application/json")
public List<Becas> getSlider() {
    List<Becas> becas=becasService.findAllBecas();
    List<Conferencias> conferencias=conferenciaService.findAllConferencias();
    List<Congresos> congresos=congresosService.findAllCongresos();
    List<Cursos> cursos=cursosService.findAllCursos();
    List<Eventos> eventos=eventosService.findAllEventos();
    List<Practicas> practicas=practicasService.findAllPracticas();

    List<Becas> slider= new ArrayList<Becas>();
```

Figura 40: Implementación controlador del Slider, parte uno

```

for(int i=becas.size()-1;i>=becas.size()-3;i--) {
    slider.add(becas.get(i));
}
for(int i=conferencias.size()-1;i>=conferencias.size()-3;i--) {
    Becas beca=new Becas();
    beca.setID(conferencias.get(i).getID());
    beca.setID_WORDPRESS(conferencias.get(i).getID_WORDPRESS());
    beca.setTITULO(conferencias.get(i).getTITULO());
    beca.setIMAGEN(conferencias.get(i).getIMAGEN());
    beca.setFECHA(conferencias.get(i).getFECHA());
    beca.setCONTENIDO(conferencias.get(i).getCONTENIDO());
    beca.setLINK(conferencias.get(i).getLINK());
    beca.setVISITAS(conferencias.get(i).getVISITAS());

    slider.add(becca);
}
for(int i=congresos.size()-1;i>=congresos.size()-3;i--) {
    Becas becal=new Becas();
    becal.setID(congresos.get(i).getID());
    becal.setID_WORDPRESS(congresos.get(i).getID_WORDPRESS());
    becal.setTITULO(congresos.get(i).getTITULO());
    becal.setIMAGEN(congresos.get(i).getIMAGEN());
    becal.setFECHA(congresos.get(i).getFECHA());
    becal.setCONTENIDO(congresos.get(i).getCONTENIDO());
    becal.setLINK(congresos.get(i).getLINK());
    becal.setVISITAS(congresos.get(i).getVISITAS());

    slider.add(becal);
}
}

```

Figura 41: Implementación controlador del Slider, parte dos

Después de ver los controladores y su función dentro del servicio web, vamos a comentar cómo funciona la conexión del controlador con los demás paquetes de clases de la aplicación y sus funciones. Dentro del servicio web tenemos el paquete Repository que será la capa donde implementaremos las conexiones mediante JPA. Después tenemos las clases del paquete de Dominio que contiene las entidades con los campos que queremos que mapee JPA y acabamos con los servicios. Dentro de los servicios tenemos con los controladores una interfaz servicios, que contendrá la definición de los servicios existentes y en ServiceImple las clases que implementan estos servicios. Los servicios son los encargados de desempeñar la capa de negocio. Estos paquetes descritos se muestran junto con sus clases en la Figura 42.

Este reparto de paquetes y estructura atiende a una buena práctica de programación en la que el repositorio tiene que pasar por los servicios antes de que el controlador los devuelva y que además tanto los controladores como los servicios cuando se comuniquen con ellos sea a través de las interfaces. A continuación, se explicarán más detalladamente estas clases, así como sus métodos y funciones dentro de la aplicación.



Figura 42: Paquetes Dominio, Repository, Service y ServiceImple del esquema de paquetes

En el paquete dominio están las entidades de cada uno de los tipos de noticias que tenemos en la aplicación, esto se hizo para adecuarse a las tablas de base de datos y así poder utilizar JPA que establece conexión con la base de datos promedio de objetos, es decir, JPA mapea los objetos de una base de datos o asigna objetos a las tablas de la base de datos y viceversa. Esto reduce la carga de interactuar con la BBDD de forma notable.

Las Entidades como hemos comentado están creadas para ser iguales que las tablas de la base de datos, con los mismos atributos. Esto quiere decir que tendremos los atributos guardados en la base de datos en la propia entidad. Estos atributos son:

- Id: atributo de tipo Integer que representa el identificador dentro de la tabla correspondiente.
- Id_wordpress: atributo de tipo Integer que identifica de manera inequívoca a la noticia dentro de las noticias de wordpress.
- Título: como su propio nombre indica, es un atributo String que tiene el valor del titular de la noticia.

-
- Fecha: tiene el valor de la fecha en que se publica la noticia y es representada mediante un atributo de tipo String.
 - Contenido: String que contiene el interior de la noticia, todo el cuerpo e imágenes internas que tiene, así como las posibles plantillas para el guardado de la noticia que podría incluir.
 - Imagen: también es un atributo de tipo String que contiene la URL de la imagen principal de la noticia.
 - Link: enlace de la noticia en la página web inform@Uva que se representa mediante un String.
 - Visitas: Este atributo tiene como objetivo la monitorización de la aplicación y por medio su valor de tipo Integer sabemos el número de visitas obtenidas por una noticia.

```
package com.example.core.dominio;

import javax.persistence.Column;

@Entity
@Table(name = "becas")
public class Becas {
    @Id
    @Column(name="id")
    private int ID;

    @Column(name="id_wordpress")
    private int ID_WORDPRESS;

    @Column(name="titulo")
    private String TITULO;

    @Column(name="fecha")
    private String FECHA;

    @Column(name="contenido")
    private String CONTENIDO;

    @Column(name="imagen")
    private String IMAGEN;

    @Column(name="link")
    private String LINK;

    @Column(name="visitas")
    private int VISITAS;
}
```

Figura 43: Entidad del Dominio con sus atributos

Como se aprecia en la Figura 43, se observan los atributos antes descritos como una serie de anotaciones. La anotación `@Entity` sirve para indicarle al JPA que esa clase es una entidad. Luego está la anotación `@Table` que nos permite especificar los detalles de la tabla que se utilizará para conservar la entidad en la base de datos. Por último, encontramos dos anotaciones sobre los atributos de la clase que son, `@Id` que identifica el atributo es clave primaria de la

entidad y también tenemos la anotación `@Columns` que simplemente nos indica el nombre de la columna que corresponde al atributo dentro de la base de datos.

Lo siguiente que vamos a comentar son la función de las clases dentro del paquete `Repository`. En estas clases donde añadiremos el JPA contamos con una por cada categoría de noticia como en el resto de los paquetes.

```
package com.example.core.repository;

import java.util.Optional;

@Repository
public interface BecasRepository extends JpaRepository<Becas, Integer> {
    void save(Optional<Becas> becasToUpdate);
}
```

Figura 44: Clase Repository capa donde se integra el JPA

Como podemos observar en la Figura 44, estas clases extienden de `JpaRepository` que incluye muchos de los métodos de acceso a la base de datos que necesitamos como son `findAll()`, o `findById(ID id)` entre otros. A parte de esto, mapeamos la capa correspondiente, en este caso `Becas` y le indicamos que la clave primaria es de tipo `Integer`. También destacar que añadimos un método (el método `save()`) para cuando actualicemos la base de datos guardar el elemento actualizado. Los métodos que incluye la clase `JpaRepository` son los que utilizamos de forma directa para obtener la información de la base de datos y para modificar la base de datos.

La clave de esta estructura de paquetes es respetar las buenas prácticas de programación y por eso el repositorio pasa por el servicio y luego ya el controlador devuelve la respuesta. Ahora veremos la parte de los servicios, empezando por la interfaz.

```
package com.example.core.service;

import java.util.List;

public interface BecasService {
    public List<Becas> findAllBecas();
    public Optional<Becas> findBecasById(int id);
    // public Becas saveBecas(Becas becasNew);
    // public String deleteBecas(Long id);
    public String updateBecas(int becasNew);
}
```

Figura 45: Clase interfaz Servicio

En la Figura 45 se muestra la interfaz de servicio, en este caso como en el resto de los ejemplos que se han mostrado en las figuras es de la categoría `becas`, pero es extrapolable al resto de las categorías presentes. Como se observa, tenemos tres métodos en la interfaz. El

primero, *findAllBecas()* obtiene en una lista todas las noticias de la tabla Becas de la base de datos, mientras que el segundo método, llamado *findBecasById(int id)*, lo utilizamos para obtener una noticia en concreto según el id. Y el tercer y último método es el utilizado para modificar la base de datos pasándole como argumento el id de wordpress de la noticia y después, en la implementación veremos cómo se busca la noticia y se le modifica su atributo Visitas sumándole uno.

Para finalizar, explicaremos como se implementan los servicios. Como vemos en la Figura 46, tenemos la anotación `@Service` que es una especialización de la anotación `@Component`, que indica que la clase es un servicio dentro de la aplicación.

Lo primero que se observa dentro de la clase es el `@Autowired` sobre un atributo Repositorio para conectar con la clase Repositorio que se comunica con la base de datos. Lo siguiente es el primer método implementado de los servicios, *findAllNoticia()* que básicamente devuelve una lista de las noticias de la categoría en cuestión realizando una llamada a la función de la clase `Repository findAll()` que coge de la base de datos la información requerida.

```
import java.util.List;

@Service
public class BecasServiceImpl implements BecasService {

    @Autowired
    BecasRepository becasRepository;

    // private static final Logger logger = LoggerFactory.getLogger(BecasServiceImpl.class);

    @Override
    public List<Becas> findAllBecas() {
        return becasRepository.findAll();
    }

    @Override
    public String updateBecas(int becasNew) {
        List<Becas> becas=becasRepository.findAll();

        boolean esta=true;
        int i =0;
        Becas becaUpdate=new Becas();
        while(esta) {
            int id_wp=becas.get(i).getID_WORDPRESS();
            if(id_wp==becasNew) {
                becaUpdate=becas.get(i);
                esta=false;
            }

            i++;
        }

        if(becasRepository.findById(beccaUpdate.getID()).isPresent()) {

            becaUpdate.setVISITAS(beccaUpdate.getVISITAS()+1);

            becasRepository.save(beccaUpdate);
            return "Beca modificada";
        }

        return "Erro al actualizar";
    }

    @Override
    public Optional<Becas> findBecasById(int id) {
        Optional<Becas> beca = becasRepository.findById(id);
        return beca;
    }
}
```

Figura 46: Clase `BecasServiceImpl`, ejemplo de implementación de servicio

El siguiente método es el que se encarga de manejar la monitorización de la aplicación, y funciona de la siguiente manera. En primer lugar, obtenemos la lista de todas las noticias de la categoría que sea, después buscamos dentro de esta lista la noticia que contenga el `id_wordpress` que llega como argumento en el método. Una vez encontrado, obtenemos esa noticia y cambiamos el valor de su atributo `Visitas` sumándole uno. Después se llama a la función `save(noticia)` con la noticia a actualizar como argumento y de esta forma se actualiza el valor en la base de datos. La monitorización como se puede comprobar consiste en sumar uno al valor `Visitas` de una noticia cada vez que un usuario accede a ésta, dándonos la información de qué noticias son las más visitadas y cuales las que menos.

Por último, el método `findbyId(int id)` se encarga de obtener una noticia en concreto según el `id` que se pasa como argumento, y este argumento se pasa directamente al método `findById(id)` de `Repository` que busca la noticia en la base de datos.

```
HttpClient httpClient = new DefaultHttpClient();
String url = "http://157.88.130.185:8080/uva-rest-service5/uvanow/cursos";
HttpGet del = new HttpGet(url);

del.setHeader( name: "accept", value: "application/json");

try
{
    /**
     * El servidor nos da una respuesta y nos devuelve un objeto JSON, el cual t
     * y dentro de este array tenemos información como id, titulo noticia, fech
     */
    HttpResponse resp = httpClient.execute(del);
    String respStr = EntityUtils.toString(resp.getEntity());

    byte[] bytes = respStr.getBytes(StandardCharsets.ISO_8859_1);

    String utf8respStr = new String(bytes, StandardCharsets.UTF_8);
    JSONArray jsonArray = new JSONArray(utf8respStr);

    longitud = jsonArray.length();
    id = new int[longitud];
    id_wordpress = new int[longitud];
    titulo_noticia = new String[longitud];
    fecha = new String[longitud];
    contenido = new String[longitud];
    imagen = new String[longitud];
    link = new String[longitud];
    dia = new String[longitud];
    mes = new String[longitud];
    year = new String[longitud];
}
```

Figura 47: Llamada al servicio Rest para obtener todas las noticias de una categoría

En la figura 47 se muestra la llamada realizada desde la aplicación UVaNOW al servicio web desplegado en nuestro servidor remoto. Como se aprecia, realizamos una llamada `HttpGet` con la URL del servicio web, para luego con la clase `HttpResponse` obtener la respuesta a la llamada que mediante la función `EntityUtils.toString(resp.getEntity())` obtenemos en un valor de tipo `String` la respuesta. El siguiente paso es aplicar dos funciones para solucionar el

problema con el Encode, debido a que la respuesta llega en una codificación diferente a la deseada y esto se resolvió con estas dos líneas:

```
byte[] bytes = respStr.getBytes(StandardCharsets.ISO_8859_1);  
String utf8respStr = new String(bytes, StandardCharsets.UTF_8);
```

Por último, antes de empezar a manejar la información obtenida es necesario transformar el *String* de la respuesta, después de cambiar el Encode al formato UTF-8, en un *JSONArray* que contenga la lista de noticias cada una siendo un *JSONObject* con el que podemos trabajar. De estos elementos JSON sacamos los datos como el id, la fecha, el título de la noticia, etc.

Para terminar este apartado, vemos en la Figura 48 como creamos y ejecutamos una clase creada llamada SumarVisita.

```
SumarVisita sumarvisita=new SumarVisita();  
sumarvisita.execute();
```

Figura 48: Clase SumarVisita

Esta clase extiende de *AsyncTask*, para que se ejecute en segundo plano y no detenga el funcionamiento normal de la aplicación por algún fallo de la llamada. Dentro de la aplicación cuando se accede al contenido de una noticia, esta clase se “ejecuta” en segundo plano, realizando una llamada al servicio Rest. Esta llamada es de tipo PUT ya que consiste en modificar algo de la base de datos, en este caso el atributo Visitas de una noticia sumándole uno. Por ello como se ve en la Figura 49, en la llamada al final de la URL se le añade el *id_wordpress* de la noticia con lo que los servicios web saben qué noticia modificar.

```
public class SumarVisita extends AsyncTask<String,Integer,Boolean> {  
  
    /**  
     * Método donde se ejecutan las acciones en segundo plano  
     * @param params indica información de entrada a la tarea.  
     */  
    protected Boolean doInBackground(String... params) {  
        Bundle extras = getIntent().getExtras();  
  
        int id_wordpress=extras.getInt( key: "id_wordpress");  
  
        HttpClient httpClient = new DefaultHttpClient();  
        String url ="http://157.88.130.185:8080/uva-rest-service5/uvanow/becas/"+id_wordpress  
        HttpPut del = new HttpPut(url);  
  
        boolean resul = true;  
        try {  
            HttpResponse response = httpClient.execute(del);  
        } catch (IOException e) {  
            Log.e( tag: "ServicioRest", msg: "Error!", e);  
            resul = false;  
        }  
  
        return resul;  
    }  
}
```

Figura 49: Implementación de la clase SumarVisita

5.4.2 Aplicaciones Java que actualizan la Base de Datos

Los otros servicios que desarrollar eran los encargados de coger los datos de la página web `inform@Uva` (en WordPress), que estaban escritos en el lenguaje PHP, sustituyéndoles por un programa escrito en java que se ejecutara de manera diaria para ir actualizando al base de datos de una manera más cómoda. Al estar la página desarrollada en Wordpress tanto los anteriores servicios como los nuevos obtienen las noticias de la página haciendo uso de la WP API Rest. Al final en vez de un proyecto que sustituyera a los siete servicios PHP anteriores se desarrollaron dos proyectos, uno para la actualización diaria de las noticias y otro para cuando alguna noticia o varias requieren alguna modificación. Se explicarán más en detalle en los siguientes apartados.

5.4.2.1 Descripción de las aplicaciones Java

En este apartado se describirán las herramientas empleadas para su desarrollo y subida al nuevo servidor desplegado. Lo primero que se hizo fue ver cómo funcionaban los anteriores servicios y lo elementos que traían desde Wordpress. Una vez analizados los puntos claves se creó un programa en Netbeans que fue el IDE donde se desarrollaron ambas aplicaciones Java. La primera aplicación que describiremos es la más importante, que es la que se encarga de actualizar la base de datos todos los días.

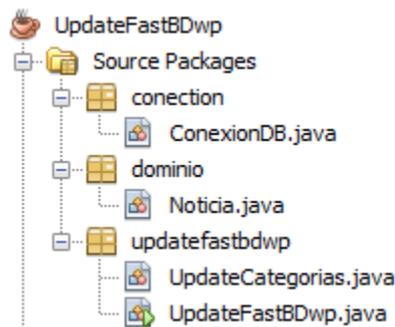


Figura 50: Estructura de paquetes de la aplicación de actualización de la Base de Datos

Como se ve en la Figura 50, se muestra la estructura de paquetes formada por el paquete conexión que contiene la clase dedicada a realizar la conexión con la base de datos llamada `UpdateFastBDwp`, por ser la aplicación más ágil para actualizar las noticias pues solo coge de Wordpress las noticias que no están en la base de datos aún. Otro paquete es el de dominio que contiene la entidad `Noticia` con sus atributos, getters (métodos para obtener atributos de un objeto) y setters (métodos para dar valor a un atributo del objeto). Y por último tenemos el paquete `updatefastbdwp`, en el que están las dos clases encargadas del funcionamiento principal de la aplicación.

Empezaremos hablando de la clase dedicada a la conexión a la base de datos, que como se observa en la Figura 51 el primer caso es crear las variables de tipo Connection para la conexión, Statement para las consultas SQL y ResultSet para obtener los resultados de las consultas. Lo siguiente es crear las variables String que contendrán los valores del nombre de la base de datos, su usuario, su contraseña y la URL de la base de datos. Esta URL está formada por la dirección del localhost, más el puerto 3306 de MySQL y se le añaden unos atributos a la URL para que el Encode sea UTF-8.

Después, damos valor a la variable de conexión por medio de DriverManager.getConnection() más las variables String con los datos de conexión a la base de datos. Con esto se realiza la función de conectarse dentro del método setDBConnection. Para estos métodos hay que hacer que tengan las excepciones de SQLException.

```
public class ConexionDB {
    public static Connection dbCon;
    public Statement dbStmt;
    public ResultSet dbRst;

    public static Connection setDBConnection() throws SQLException{

        String db="labcomuva";
        String login="root";
        String password="OpclbpJ2021";
        // String url="jdbc:derby://localhost:1527/"+db;//+"?useUnicode=true&characterEncoding=utf8_spanish_ci
        String url="jdbc:mysql://127.0.0.1:3306/"+db+"?useUnicode=true&characterEncoding=utf8&useSSL=false";
        try{

            //Class.forName("com.mysql.jdbc.Driver");
            dbCon=DriverManager.getConnection(url,login,password);

        }catch(Exception e){
            System.out.println(e);
        }

        return dbCon;
    }
}
```

Figura 51: Clase conexionDB de la aplicación Java

En el dominio se encuentra la entidad Noticia que utilizamos para “guardar” los datos que nos proporciona la API Rest de Wordpress. Esta entidad tiene como atributos, el Id, el Id_wordpress, titulo, contenido, fecha e imagen, e incluimos el nuevo valor que cogemos de la base de datos, que es el link de la noticia en la página web.

Ahora explicaremos cómo funciona la aplicación. Para empezar, tenemos la clase UpdateFastBDwp del mismo nombre que la aplicación que se encarga de ejecutar una por una las actualizaciones de las distintas tablas de la base de datos. Para ello como se ve en la Figura

52, lo primero es crear dos variables, una variable de tipo String con la URL con la petición API Rest a nuestra página web inform@Uva sin completar ya que luego dentro de la clase UpdateCategorias se completa. Y la otra variable que se crea es de la clase UpdateCategorias que acabamos de nombrar, clase donde se desarrolla la mayor parte de la funcionalidad del servicio. Y después se va llamando uno por uno a los métodos dentro de esa clase que actualizan cada tabla de la base de datos mientras se muestra por pantalla el proceso de actualización.

```
public class UpdateFastBDwp {  
  
    /**  
     * @param args the command line arguments  
     * @throws java.net.MalformedURLException  
     */  
    public static void main(String[] args) throws MalformedURLException, JSONException, IOException {  
  
        String urlApi="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=";  
  
        // String urlBecas="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=100&categories=3";  
        // String urlConferencias="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=100&categories=5";  
        // String urlCongresos="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=100&categories=4322";  
        // String urlCursos="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=100&categories=4";  
        // String urlEventos="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=100&categories=19";  
        // String urlPracticas="http://www.informauva.com/wp-json/wp/v2/posts/?per_page=100&categories=7";  
  
        UpdateCategorias update= new UpdateCategorias();  
  
        System.out.println("Actualizando base de datos. Espere a que termine el proceso...\n");  
  
        update.updateBecas(urlApi);  
        System.out.println("\nCategoria Becas actualizada\n");  
        update.updateConferencias(urlApi);  
        System.out.println("\nCategoria Conferencias actualizada\n");  
        update.updateCongresos(urlApi);  
        System.out.println("\nCategoria Congresos actualizada\n");  
        update.updateCursos(urlApi);  
        System.out.println("\nCategoria Cursos actualizada\n");  
        update.updateEventos(urlApi);  
        System.out.println("\nCategoria Eventos actualizada\n");  
        update.updatePracticas(urlApi);  
        System.out.println("\nCategoria Prácticas actualizada\n");  
  
        System.out.println("\n *** TODO ACTUALIZADO ***\n");  
    }  
}
```

Figura 52: Clase UpdateFastBDwp

Para terminar la explicación del funcionamiento, hablaremos de la clase principal de la aplicación donde esta mayor complejidad de la aplicación. En esta clase hay una serie de métodos, exactamente seis, uno por cada categoría de noticia de la base de datos. Este método que se muestra en la Figura 53 se encarga de realizar la petición a la API Rest de Wordpress para saber cuántas noticias hay de una categoría. Esto se consigue en la petición indicando que el atributo “categoría” tiene el valor de la categoría que queremos y en la cabecera se nos devuelve el número de noticias total de la categoría en el atributo “X-WP-Total”. Una vez obtenido este valor sabemos cuántas peticiones debemos hacer a la API. Esto es importante conocerlo porque la WP API Wordpress solo puede devolver un máximo de 100 elementos por petición y hay categorías con más noticias que 100, problema que no fue resuelto en el anterior proyecto y por ello solo se traían a la base de datos las últimas 100 noticias cuando se ejecutaban los programas PHP. Ahora se realizarán tantas llamadas como sean necesarias para traer todas

las noticias, con un bucle *for* que las realiza y crea *JSONArray*s de las noticias que se van concatenando para obtener el *JSONArray* completo con todas las noticias.

```
void updatePracticas(String urlPracticas) throws MalformedURLException, JSONException, IOException {
    String urlBecasl=urlPracticas+"&categories=7";

    //obtenemos el total de elementos para poder hacer el bucle de peticiones ya que Api Wordpress solo permite recuperar un maximo de 100 recursos por peti
    HttpClient client = HttpClientBuilder.create().build();
    HttpGet request = new HttpGet(urlBecasl);
    HttpResponse response = client.execute(request);

    Header[] headers = response.getAllHeaders();

    int numNoticias = Integer.parseInt(response.getFirstHeader("X-WF-Total").getValue());//Con esa clave viene el valor del total de recursos de esa categoria de
    System.out.println("Noticias en Wordpress: "+numNoticias);

    int numPeticiones=(int) Math.ceil((double)numNoticias/100);
    String value="";

    int marcador=0;

    JSONArray jsonarray=new JSONArray();

    int numNotiBD=getNumeroNoticias("practicas");
    System.out.println("Noticias en base de datos: "+numNotiBD);

    if(numNoticias-numNotiBD>0) {
        for(int i=0;i<numPeticiones;i++){
            URL url = new URL(urlPracticas+"100&offset="+marcador+"&categories=7");
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setRequestProperty("Accept", "application/json");

            BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputStream())));

            marcador=marcador+100;

            value = readAll(br);
            JSONArray jsenal = new JSONArray(value);

            jsonarray=concatArray(jsonarray,jsenal);
        }
    }
}
```

Figura 53: Primera parte del método updatePrácticas

El siguiente paso es llamar al método `saveJsonToList` para transformar ese *JSONArray* en una *ArrayList* de noticias. En la Figura 54 se observa cómo funciona este método, que básicamente consiste en un bucle *for* que coge cada elemento *JSONObject* que forma parte del *JSONArray* que obtuvimos de las llamadas a la API de Wordpress. Y de cada uno de estos elementos obtenemos los atributos de la clase *Noticia* uno a uno del JSON mediante los nombres establecidos para ellos dentro de este JSON. Algo importante a destacar y que es uno de los motivos por el que se realizaron dos aplicaciones es que esta aplicación solo introduce las nuevas noticias a la base de datos, ahorrando mucho tiempo y siendo más eficaz para el día a día ir actualizando la base de datos. Para este proceso es importante el momento en que se resta el número de noticias que hay en la base de datos con el número de noticias que hay en Wordpress y así solo introducir las últimas noticias de cada categoría. En el caso de las imágenes sucede un caso especial, esto es porque se tiene en consideración varios casos, desde que la noticia contenida en ese JSON tenga la imagen en una resolución alta hasta el caso en el

que no tenga ninguna imagen. En este último caso se establece la variable con un valor de 0, indicando así la ausencia de imagen. Por último, comentar la extracción del *link* de las noticias para poder implementar la función de compartir en redes sociales.

```
private ArrayList<Noticia> saveJsonToList(JSONArray jsonarray, int numNoticiasWordpress, int numNotiBD) throws JSONException, IOException {  
    ArrayList<Noticia> noticias = new ArrayList<>();  
    Noticia noti;  
    JSONObject jsonNoticia;  
  
    if(numNotiBD<numNoticiasWordpress){  
        for (int i = 0; i < (numNoticiasWordpress-numNotiBD) && (jsonarray.length()-i-1)>=0; i++) {  
            System.out.println("Noticia "+(jsonarray.length()-i-1)+" de "+numNoticiasWordpress);  
            jsonNoticia = jsonArray.getJSONObject(i);  
  
            noti = new Noticia();  
            noti.setID(jsonarray.length()-i);  
  
            noti.setID_WORDPRESS(jsonNoticia.getInt("id"));  
  
            JSONObject titulo=jsonNoticia.getJSONObject("title");  
            noti.setTITULO(titulo.getString("rendered"));  
  
            JSONObject contenido=jsonNoticia.getJSONObject("content");  
            noti.setCONTENIDO(contenido.getString("rendered").replaceAll("'", " "));  
  
            noti.setFECHA(jsonNoticia.getString("date").replaceAll("T", " "));  
  
            String imagenF=null;  
            JSONObject imagen2 = null;  
            try{  
                String uriImagen=jsonNoticia.getJSONObject("_links").getJSONArray("wp:featuredmedia").getJSONObject(0).getString("href");  
                imagen2 = readJsonFromUrl(uriImagen);  
            }catch (Exception e){  
                imagenF=null;  
            }  
            try{  
                imagenF =imagen2.getJSONObject("media_details").getJSONObject("sizes").getJSONObject("medium_large").getString("source_url");  
            }catch (Exception e){  
                imagenF=null;  
            }  
            if (imagenF==null){  
                try{  
                    imagenF=imagen2.getJSONObject("media_details").getJSONObject("sizes").getJSONObject("large").getString("source_url");  
                }catch (Exception e){  
                    imagenF=null;  
                }  
            }  
            if (imagenF==null){  
                try{  
                    imagenF=imagen2.getJSONObject("guid").getString("rendered");  
                }catch (Exception e){  
                    imagenF=null;  
                }  
            }  
  
            if (imagenF==null){  
                imagenF="0"; //No hay imagen  
            }  
            noti.setIMAGEN(imagenF);  
            noti.setLINK(jsonNoticia.getString("link"));  
            noticias.add(noti);  
        }  
    }  
}
```

Figura 54: Método para pasar el JSONArray a un ArrayList de Noticias

El siguiente paso y último es introducir todas las noticias de la lista que hemos obtenido después de todo el proceso en la base de datos. En la Figura 55 se aprecia claramente el proceso con el que esto se lleva a cabo. El primer paso es realizar un bucle *for* para ir cogiendo noticia a noticia y después mediante la consulta SQL, que obtienen los valores de los atributos de la noticia y los introducimos en la base de datos.

Hay que comentar que para todas las noticias que se introducen, ya que es un programa cuyo objetivo es introducir nuevas noticias, se introducen con el número de visitas a 0. Al final

del todo se muestra un mensaje del total de noticias que han sido introducidos en la base de datos.

```
//Almacenamos la información del json en una lista de noticias
ArrayList<Noticia> listNoti = saveJsonToList(jsonarray,numNoticias,numNotiBD);

if(listNoti!=null){
    for(int i=0; i<listNoti.size();i++){

        dbCon=new ConexionDB();

        String SSQL = "INSERT INTO practicas (id, id_wordpress, titulo, contenido, fecha, imagen,link,visitas) "
+ "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        try{
            conn=(Connection) ConexionDB.setDBConnection();
            PreparedStatement psql = conn.prepareStatement(SSQL);
            psql.setInt(1, listNoti.get(i).getID());
            psql.setInt(2, listNoti.get(i).getID_WORDPRESS());
            psql.setString(3, listNoti.get(i).getTITULO());
            psql.setString(4, listNoti.get(i).getCONTENIDO());
            psql.setString(5, listNoti.get(i).getFECHA());
            psql.setString(6, listNoti.get(i).getIMAGEN());
            psql.setString(7, listNoti.get(i).getLINK());
            psql.setInt(8, 0);

            psql.executeUpdate();

            psql.close();

        }catch(SQLException e){
            e.printStackTrace();
        }finally{
            if(conn!=null){
                try{
                    conn.close();
                }catch(SQLException e){
                    e.printStackTrace();
                }
            }
        }

        System.out.println("Total de noticias insertadas en base de datos: "+(numNoticias-numNotiBD));
    }
}
else{
    System.out.println("No hay nuevas noticias que insertar en la base de datos");
}
}
```

Figura 55: Segunda parte método updatePracticas

Finalmente, se pasa a explicar el segundo programa, muy similar al primero pero donde solo cambia el funcionamiento en un ciertos de puntos de la aplicación. El primer cambio es el punto en el que en la primera aplicación se tomaba el número total de noticias de una categoría en Wordpress y le restábamos en cierto momento los elementos de esa categoría que estaban en la base de datos y así obtener el número de noticias nuevas que hay que incorporar. En este segundo siempre cogeremos todas las noticias de Wordpress debido a que la función de esta aplicación es actualizar una o varias noticias e introducir nuevas noticias en la base de datos si es el caso. Como se puede ver en la Figura 56, en la parte del método donde se realiza la introducción de elementos a la base de datos se incluye una comprobación de si esa noticia existía ya en la base de datos y de ser el caso en vez de introducir una nueva noticia lo que hace el programa es actualizar la noticia con los elementos traídos de Wordpress. Este procedimiento

como hemos explicado es largo y requiere bastante tiempo, de ahí la importancia de tener una aplicación más ligera como la primera que realice la tarea solo de incorporar nuevas noticias a la base de datos y que se ejecute de manera diaria.

```
if(!estaEnBaseDatos(listNoti.get(i).getID_WORDPRESS(),"becas")){

    dbCon=new ConexionDB();

    String SSQL = "INSERT INTO becas (id, id_wordpress, titulo, contenido, fecha, imagen,link,visitas) "
+ "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
    try{
        conn=(Connection) ConexionDB.setDBConnection();
        PreparedStatement psql = conn.prepareStatement(SSQL);
        psql.setInt(1, listNoti.get(i).getID());
        psql.setInt(2, listNoti.get(i).getID_WORDPRESS());
        psql.setString(3, listNoti.get(i).getTITULO());
        psql.setString(4, listNoti.get(i).getCONTENIDO());
        psql.setString(5, listNoti.get(i).getFECHA());
        psql.setString(6, listNoti.get(i).getIMAGEN());
        psql.setString(7, listNoti.get(i).getLINK());
        psql.setInt(8, 0);

        psql.executeUpdate();

        psql.close();
    }catch(SQLException e){
        e.printStackTrace();
    }finally{
        if(conn!=null){
            try{
                conn.close();
            }catch(SQLException e){
                e.printStackTrace();
            }
        }
    }
}
}

dbCon=new ConexionDB();

String SSQL = "UPDATE becas set titulo=?, contenido=?, fecha=?, imagen=?,link=? where id_wordpress=? ";
try{
    conn=(Connection) ConexionDB.setDBConnection();
    PreparedStatement psql = conn.prepareStatement(SSQL);
    psql.setInt(6, listNoti.get(i).getID_WORDPRESS());
    psql.setString(1, listNoti.get(i).getTITULO());
    psql.setString(2, listNoti.get(i).getCONTENIDO());
    psql.setString(3, listNoti.get(i).getFECHA());
    psql.setString(4, listNoti.get(i).getIMAGEN());
    psql.setString(5, listNoti.get(i).getLINK());

    psql.executeUpdate();

    psql.close();
}catch(SQLException e){
```

Figura 56: Introducción y actualización de noticias en el segundo programa

Para terminar este apartado, en la Figura 56 se observa que cuando se actualiza una noticia porque ésta ya está en la base de datos, los valores que se cambian son todos menos los del Id y el de Visitas. Este último, con el objetivo de actualizar la noticia pero no eliminar el número de visitas que ya tenía antes y así no entorpecer la monitorización de la aplicación.

5.4.3 Script de ejecución diaria de aplicación.

La aplicación móvil preliminar tenía también el problema de que los servicios web PHP debían de ejecutarse todos los días de manera manual para actualizar el contenido de la base de datos. Para este proyecto se encargó no solo de la creación de unos programas que actualizaran el contenido de dichas tablas de la base de datos, sino que lo hiciera de manera automática todos los días. Por ello se procedió a la creación de un Script que ejecute el primer programa, el programa más ligero de los dos que creamos, todos los días a cierta hora (Figura 57).

```
#!/bin/bash
java -jar UpdateFastBDwp.jar
```

Figura 57: Script de ejecución de la aplicación Java de actualización diaria

Este script es ejecutado como en anteriores apartados hemos comentado usando la herramienta Cron, que se encargará de ejecutar el script todos los días a las siete y media de la mañana y de guardar en la misma carpeta un documento de texto con el resultado de la ejecución tal y como se aprecia en la Figura 58.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
30 7 * * * java -jar /home/noemer/Esitorio/ActualizacionDiaria/UpdateFastBDwp.jar > /home/noemer/Esitorio/ActualizacionDiaria/output.txt
```

Figura 58: Fichero Cron donde se incluye la línea para la ejecución diaria del Script

5.5 Pruebas

Número	Título prueba	Resumen	Pasos	Resultado
1	Prueba del servicio web, traer las noticias de una categoría	En esta prueba comprobaremos los 6 servicios que proporciona el servicio web dedicados a traer las noticias de las categorías.	1-Acceder a la aplicación. 2-Acceder a las noticias de una categoría.	Las noticias aparecen correctamente listadas en la pantalla, demostrando que el servidor trae las noticias de la Base de Datos correctamente.
2	Prueba del servicio web, traer noticias del slider.	En esta prueba comprobaremos el servicio que trae las noticias del slider	1-Acceder a la aplicación. 2-Comprobar todas las noticias del slider	El slider contiene las noticias que le corresponden.
3	Prueba del servicio web, prueba de monitorización	En esta prueba se comprueba el correcto funcionamiento del sistema de monitorización implementado.	1-Acceder a la Aplicación. 2-Acceder a la categoría “becas” (por ejemplo) 3-Seleccionar una noticia. 4-Acceder a la base de datos. 5-Comprobar que la noticia aumento en uno el número de visitas.	La monitorización funcionó de manera correcta, aumentando uno el valor de la columna de “visitas” de la noticia seleccionada en la prueba.
4	Prueba del programa de actualización rápida de la base	Comprobamos que la base de datos se está actualizando a	1-Sabemos de la incorporación de una nueva noticia en la web.	La base de datos incorporó la nueva noticia,

	de datos que se ejecuta de manera diaria.	diario correctamente por medio del programa creado	2-Al día siguiente por la mañana cuando se ejecuta el programa accedemos a la base de datos. 3-Comprobamos que esta la noticia o no y el archivo output de la ejecución dentro del alojamiento web	confirmando el correcto funcionamiento del programa. En el output situado en la misma carpeta del programa aparece el número total de nuevas noticias de cada categoría añadidos.
--	---	--	---	---

Tabla 4: Pruebas de los servicios web implementados

Al finalizar el desarrollo de los servicios web de la aplicación se realizó una prueba con 296 usuarios del público objetivo que posteriormente fueron encuestados como se cuenta en más detalle en el capítulo 6. En estas pruebas se reafirmó el correcto funcionamiento de la aplicación Android en un grupo más amplio de dispositivos de los probados hasta la fecha.

6

Subida al Market y Requisitos de la App

6.1 Introducción

El último paso llevado a cabo en este proyecto antes de documentar todo, fue la subida al Market de Google de la aplicación. En este capítulo se ven los resultados obtenidos de la prueba realizada en la que participaron casi 300 personas para testear la aplicación y se muestran los pasos seguidos para la subida de la aplicación al Market con el objetivo de ponerlo a disposición de los usuarios esta vez de manera pública. Sin olvidar que al final del capítulo enumeramos los requisitos de la aplicación.

6.2 Encuesta y opiniones antes de subir al Market

La aplicación antes del proceso de subida fue lanzada a modo de prueba y de manera controlada a su futuro público objetivo (alumnos de la Universidad de Valladolid) y con una encuesta para obtener las opiniones sobre la aplicación. La participación fue alta, contestaron a la encuesta un total de 296 alumnos con unos resultados satisfactorios que se ven reflejados en la Tabla 1.

Pregunta realizada	Opciones	Porcentajes
¿Cómo de intuitiva es la aplicación para su uso y navegación?	Nada	3%
	Poco	27%
	Mucho	70%
En cuanto a la velocidad de acceso a los contenidos, considera que UVaNow es:	Lenta	9%
	Normal	70%
	Rápida	21%
Gráficamente, ¿cómo considera que es el sitio?	Recargado	4%
	Equilibrado	35%
	Simple	61%
¿Cree que sería útil recibir alertas que avisen de la subida de nuevos contenidos?	Si	83%
	No	17%
	Si	46%

¿Le parecería bien que la aplicación geolocalizase su ubicación para personalizar el contenido de acuerdo con el campus universitario al que usted pertenezca?	No	54%
--	----	-----

Tabla 5: Resultados de la encuesta sobre la aplicación

En la Tabla 1, podemos apreciar que la navegación y uso es mayoritariamente favorable. Apreciamos que en un 91% de los dispositivos la aplicación funcionaba a una velocidad normal o rápida lo que nos permite afirmar que la aplicación funciona correctamente en la mayoría de los dispositivos actuales. Los encuestados consideraron el diseño de la aplicación como simple y equilibrada en su gran mayoría con un 96 %, lo cual es uno de los objetivos de una aplicación de estas características. Hay que destacar que a los usuarios que respondieron a la encuesta no les importaría recibir notificaciones de nuevas noticias en la App en su gran mayoría, un 83%. Sin embargo, cabe destacar que éste no fue uno de los objetivos de este proyecto porque en primera instancia las opiniones sobre un sistema de notificaciones eran dispares. Por último, corroborar que un sistema de geolocalización por parte de la aplicación para personalizar el contenido no es una funcionalidad muy demandada. En el caso de las personas encuestadas, un mayor porcentaje afirmaba no querer esa funcionalidad en la aplicación, exactamente un 54% de los encuestados frente al 46% que afirmaban parecerle bien dicha funcionalidad.

6.3 Subida de la aplicación al Market

El primer punto, es generar el fichero que debemos subir al Google Play Store [31]. Este fichero es de tipo .apk, y para generarlo hay que navegar dentro de Android Studio a la pestaña de “Build” y de las opciones que se nos muestran seleccionamos “Generate Signes Bundle or APK”, seleccionamos “Apk” y elegimos si tenemos un fichero en formato .jks que contiene una clave necesaria para la subida de una aplicación y que debe usarse siempre la misma y si no tenemos dicho fichero por que la aplicación es la primera vez que se sube, generamos uno mediante el comando keytool. Ya solo queda realizar la subida al Google Play Store siguiendo los pasos siguientes:

1. Iniciar sesión en Google Play Developer Console con la cuenta de Google.
2. Creamos una aplicación nueva y rellenamos la información pertinente de la aplicación, como categoría, precio, icono de la aplicación, descripción, etc.
3. El siguiente paso es seleccionar el apartado “Producción” de la columna de la izquierda.
4. Una vez en esta ventana, seleccionamos el botón “Crear Versión” y se nos abrirá un formulario donde subiremos la Apk y completaremos la información con la descripción y nombre de la versión (Figura 54)

-
5. Una vez creada, seleccionamos el botón de “Iniciar lanzamiento a producción”, esto hará que pase a una fase de revisión que dura entre 24 h y 7 días. Posteriormente la aplicación ya estará publica en el Market.

Crear versión de producción

Firma de aplicaciones de Play

Versiones no firmadas por Google Play

Google no protege la clave de firma de la aplicación. Acepta el uso de Android App Bundles. [Más información](#)

[Activar](#)

App bundles y APKs



Suelta Android App Bundles (.aab) o APKs aquí para subirlos

[Subir](#) [Añadir de la biblioteca](#)

Detalles de la versión

Nombre de la versión *

Solo se usará para que puedas identificar esta versión y no se mostrará a los usuarios en Google Play. El nombre sugerido se basa en el primer app bundle o APK de esta versión, pero puedes cambiarlo. 0/50

Notas de la versión [Copiar de una versión anterior](#)

<es-ES>
Introduce o pega aquí las notas de la versión en es-ES
</es-ES>

Has incluido notas de la versión en 0 idioma

Informa a los usuarios del contenido de tu versión. Introduce notas de la versión dentro de las etiquetas de cada idioma.

Figura 59: Creación de una nueva versión de la aplicación en producción

6.4 Requisitos de la aplicación UVaNow

En la Tabla a, podemos ver las características principales de la aplicación que debemos tener en cuenta a la hora de instalar nuestra aplicación en un dispositivo.

Nombre de la aplicación	UVaNow
Tipo de dispositivo	Dispositivos móviles y tablets
Sistema Operativo	Android
Versión mínima	Android 5.0 Lollipop (API level 21)
Última versión	Android 11 (API level 30)
Espacio libre necesario de almacenamiento interno	3,7 Mb

Tabla 6: Características principales de la aplicación

La versión mínima como se observa en la tabla es la versión de Android 5.0 Lollipop que corresponde con la API 21, fue la escogida por ser la que abarca al 95 % de los dispositivos Android del mercado, es decir, la aplicación funciona para esa versión y para todas las versiones que le preceden, abarcando el 95 % de los dispositivos Android que hay actualmente.

La última versión para la que la aplicación fue optimizada fue Android 11 que corresponde con la API 30.

7

Conclusiones y líneas futuras

7.1 Conclusiones

Este Trabajo de Fin de Grado se ha desarrollado según lo planeado, salvando el cambio de servidor que en un principio no contamos con ello. Esto fue una oportunidad de mejorar y establecer una mejor base para futuras mejoras que se puedan implementar en la aplicación, ya que ahora tenemos más control sobre el este nuevo servidor, así como ahora tenemos los nuevos servicios escritos en Java y la monitorización implementada, que nos permite obtener una información crucial para la toma de decisiones en el proyecto ahora y en el futuro. También en lo personal me permitió aprender a configurar numerosas herramientas y aprender mucho sobre el despliegue de servicios web en servidores remotos, así como el manejo de bases de datos. La aplicación fue corregida de problemas que dificultaban tener un uso plenamente satisfactorio al usuario y era uno de los objetivos de este proyecto.

Por otra parte, otro objetivo era la ampliación de las funcionalidades de la aplicación para que fuera más competitiva ante otras alternativas de carácter informativo por parte de los alumnos. La funcionalidad de compartir en redes sociales y la de guardar las noticias en el calendario en la aplicación del Calendario del dispositivo móvil han dado una gran potencialidad a la aplicación móvil.

Todo el trabajo en Android Studio me permitió aprender mucho sobre el desarrollo en esta plataforma ya que no había programado antes y era una de mis motivaciones a la hora de elegir este proyecto. Y no solo Android, en este Trabajo de Fin de Grado he aprendido sobre la configuración de servidores remotos, la creación de servicios web y bases de datos.

Por último, es destacable indicar los buenos resultados que obtuvo la aplicación en la encuesta realizada por la Facultad de Periodismo de Valladolid que permitió afirmar que la aplicación alcanzaba las cuotas de satisfacción que buscábamos cuando iniciamos el proyecto.

7.2 Líneas futuras

Después de todo el proceso de evolución de la aplicación, a pesar de ser ya plenamente funcional, este Proyecto establece una base en la que la aplicación puede ser mejorada en un futuro con mucha más facilidad. Esto es gracias a los nuevos servicios web desarrollados en Java y a tener un servidor remoto propio, con más posibilidades y control por parte de los responsables de la aplicación. El sistema de monitorización nos permite ver lo que los usuarios más les interesa lo que puede ser una fuente de nuevas ideas para implementar en la aplicación.

A pesar de la integración de nuevas funcionalidades, además de las correcciones y mejoras realizadas, siempre existe un amplio abanico de posibilidades de mejora, tanto en diseño, como en la experiencia de los usuarios finales. Así, mostramos a continuación una serie de posibles mejoras para futuras versiones de la aplicación:

- Mejora del buscador integrado en la aplicación, mejorando por ejemplo el aumento del número de palabras permitidas.
- Ampliación de dispositivos móviles en los que se puede desplegar UVaNOW. Esto se conseguiría consiguiendo la compatibilidad de la aplicación con dispositivos con el sistema operativo iOS que abarcan una importante parte del mercado de smartphones.
- Mejoras en la fluidez y velocidad de la aplicación, así como en la presentación de la funcionalidad del slider.
- Mejora a la hora de presentar la lista de noticias de una categoría integrando alguna animación de “cargando” para los dispositivos con peores características.
- Inclusión de un sistema de notificaciones con opciones de personalización es una funcionalidad que debería estudiarse según los datos de la encuesta realizada en las pruebas de la aplicación, donde un 83% consideró que estaría bien implementarlo.

8

Bibliografía

- [1] InfromaUVa, "Inform@UVA | El periódico de los estudiantes de la UVA". [Online]. <https://www.informauva.com> [26 de Junio de 2021]
- [2] WordPress, "WordPress.com: crea un sitio web o blog gratuito". [Online]. <https://wordpress.com/es/> [26 de Junio de 2021]
- [3] WordPress Developers, "REST API Handbook". [Online]. <https://developer.wordpress.org/rest-api/> [26 de Junio de 2021]
- [4] Página web de MySQL, "MySQL" [Online]. <https://www.mysql.com/> [26 de Junio de 2021]
- [5] Página web de Java, "Java | Oracle". [Online]. <https://www.java.com/es/> [26 de Junio de 2021]
- [6] Página de web de Spring. [Online]. <https://spring.io/> [26 de Junio de 2021]
- [7] Página web de Tomcat 7, "Apache Tomcat® - Apache Tomcat 7 Software Downloads". [Online]. <https://tomcat.apache.org/download-70.cgi> [26 de Junio de 2021]
- [8] PHP, "PHP: ¿Qué es PHP? - Manual". [Online]. <https://www.php.net/manual/es/intro-what-is.php> [26 de Junio de 2021]
- [9] Android Developer, "Desarrolladores de Android | Android Developers". [Online]. <https://developer.android.com/?hl=es-419> [26 de Junio de 2021]
- [10] Stack Exchange Inc. "Stack Overflow - Where Developers Learn, Share, & Build Careers". [Online]. <https://stackoverflow.com/> [26 de Junio de 2021]

-
- [11] Página web de Microsoft Teams, “Videoconferencia, reuniones, llamadas | Microsoft Teams”. [Online]. <https://www.microsoft.com/es-es/microsoft-teams/group-chat-software> [26 de Junio de 2021]
- [12] Página web Talent, “Salario para Full Stack en España - Salario Medio” [Online]. <https://es.talent.com/salary?job=full+stack> [26 de Junio de 2021]
- [13] Página web de Android Studio, “Download Android Studio and SDK tools”. [Online]. <https://developer.android.com/studio> [26 de Junio de 2021]
- [14] Página web de JetBrains, “JetBrains: Essential tools for software developers and teams”. [Online]. <https://www.jetbrains.com/> [26 de Junio de 2021]
- [15] Página web de phpmyadmin, “phpMyAdmin”. [Online]. <https://www.phpmyadmin.net/> [26 de Junio de 2021]
- [16] Página web de Visual Studio Code, “Visual Studio Code - Code Editing. Redefined”. [Online]. <https://code.visualstudio.com/> [26 de Junio de 2021]
- [17] Spring Tool Suite, “Spring | Tools”. [Online]. <https://spring.io/tools> [26 de Junio de 2021]
- Página web de Android Studio, “Download Android Studio and SDK tools”. [Online]. <https://developer.android.com/studio> [26 de Junio de 2021]
- [18] Página de Wikipedia sobre MySQL AB. [Online]. https://es.wikipedia.org/wiki/MySQL_AB [26 de Junio de 2021]
- [19] Página web de GIMP, “GIMP Descargas, tutoriales y foros. Alternativa a Photoshop gratis y libre”. [Online]. <http://www.gimp.org.es/> [26 de Junio de 2021]
- [20] Página web de PuTTY, “Download PuTTY - a free SSH and telnet client for Windows”. [Online]. <https://www.putty.org/> [26 de Junio de 2021]
- [21] Página web de FileZilla, “FileZilla: la solución FTP gratuita”. [Online]. <https://filezilla-project.org/> [26 de Junio de 2021]

-
- [22] Página web de Postman, “Postman | The Collaboration Platform for API Development”. [Online]. <https://www.postman.com/> [26 de Junio de 2021]
- [23] Baquero Garcia, J.María, “¿Qué son los web services y qué tecnología usar en su desarrollo?”. [Online]. <https://www.arsys.es/blog/programacion/web-services-desarrollo/> [26 de Junio de 2021]
- [24] Drive, “Almacenamiento en la nube para casa y el trabajo - Google Drive”. [Online]. <https://drive.google.com/> [26 de Junio de 2021]
- [25] Página web de Apache Netbeans, “Welcome to Apache NetBeans”. [Online]. <https://netbeans.apache.org/> [26 de Junio de 2021]
- [26] Java Development Kit, “Java SE Development Kit 8 - Downloads | Oracle España”. [Online]. <https://www.oracle.com/es/java/technologies/javase/javase-jdk8-downloads.html> [26 de Junio de 2021]
- [27] Erika Heidi, “Cómo instalar la pila Linux, Apache, MySQL y PHP (LAMP) en Ubuntu 20.04”. [Online]. <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04-es> [26 de Junio de 2021]
- [28] Sergio Gonzalez D., “Manual básico de cómo usar Cron”. [Online]. https://www.linuxtotal.com.mx/?cont=info_admon_006 [26 de Junio de 2021]
- [29] Página web de Maven, “Maven – Welcome to Apache Maven”. [Online]. <https://maven.apache.org/> [26 de Junio de 2021]
- [30] Página web de MariaDB, “MariaDB Foundation - MariaDB.org”. [Online]. <https://mariadb.org/> [26 de Junio de 2021]
- [31] Google Play Store. [Online]. <https://play.google.com/store?gl=ES> [26 de Junio de 2021]

-
- [32] Brian Hogan y Brian Boucheron, “How To Install Java with Apt on Debian 10”. [Online]. <https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-debian-10> [26 de Junio de 2021]
- [33] Rahul, “How To Install MySQL on Debian 10”. [Online]. <https://tecadmin.net/install-mysql-on-debian-10-buster/> [26 de Junio de 2021]
- [34] CRUD REST con Spring Boot y JPA. [Online]. <https://javadesde0.com/crud-rest-con-spring-boot-y-jpa/> [26 de Junio de 2021]
- [35] Baquero Garcia, J.María, “¿Qué son los web services y qué tecnología usar en su desarrollo?”. [Online]. <https://www.arsys.es/blog/programacion/web-services-desarrollo/> [26 de Junio de 2021]