Grado en Ingeniería Informática

Trabajo Final de Grado

Creación de una aplicación móvil para Android e iOS con Xamarin

Autor:
Cristian Ionut Valeanu

Supervisor:
Jose Luis Guillén Blasco
Tutor académico:
Juan Miguel Vilar Torres

Fecha de lectura: 17 de septiembre de 2018
Curso académico 2017/2018

Resumen

El objetivo de este documento es mostrar el trabajo realizado por el alumno durante su estancia en prácticas. Está dividido en seis capítulos: Introducción, Descripción del proyecto, Planificación del proyecto, Análisis y diseño del sistema, Implementación y pruebas y Conclusiones.

En el primer capítulo se describe la empresa, la motivación del proyecto, sus objetivos y la estructura de la memoria.

En el capítulo 2 se describe en detalle el proyecto propuesto - una **aplicación móvil multiplataforma para Android e iOS desarrollada con Xamarin para una escuela de música** - y las herramientas usadas durante el desarrollo - como IDE se ha usado Visual Studio Community 2017 y el lenguaje de programación de Xamarin es C#.

El capítulo 3 describe la planificación del proyecto, la metodología usada, la estimación de recursos y costes y seguimiento del proyecto.

El análisis y diseño del sistema están reflejados en el capítulo 4, mientras que en el capítulo 5 se describen detalles de implementación y pruebas. Finalmente, en el capítulo 6 aparecen las conclusiones del alumno.

Palabras clave

Xamarin, Android, iOS, aplicación móvil, escuela de música.

Keywords

Xamarin, Android, iOS, mobile app, music school.

Índice general

Índice de figuras

Índice de cuadros

Listings

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Californiadrims es una pequeña empresa que se dedica a proporcionar servicios y soluciones informáticos. Ofrece todo tipo de aplicaciones y servicios a los clientes que necesitan alguna ayuda, ya sea para informatizar sus instalaciones, incorporar nuevos elementos informáticos o simplemente necesitan soporte relacionado con la informática.

Este proyecto surge de la necesidad de una escuela de música, que actualmente cuenta con aproximadamente 300 alumnos, de centralizar toda su gestión y mantener a todos los implicados informados de todo lo ocurrido. Decidieron hacer esto a través de dos proyectos. Uno de ellos es una aplicación web que se centra en la parte de gestión y administración de la empresa, desde la inscripción de los alumnos y la gestión de todos los trabajadores hasta la gestión económica y administrativa de la escuela y de su equipamiento. El otro proyecto tiene como objetivo mantener informados a profesores, alumnos y seguidores de la escuela de música en todo momento. Se optó por una aplicación móvil multiplataforma, para Android e iOS, que pueda ser usada por cualquier persona. La aplicación móvil muestra a todo el público las últimas novedades de la escuela a través de su blog y página web y sus actividades planeadas para el futuro, a través de una agenda de Google Calendar.

Una **novedad** es cualquier cosa que recientemente ha pasado o que está a punto de pasar y de la que no se ha informado al público hasta ahora. Una novedad puede ser un anuncio informando que se ha abierto el plazo de matrícula, o bien, una noticia o un artículo que expone las actividades recién realizadas por la escuela, e.g., "El viernes pasado se ha celebrado el concierto de fin de curso en el que participaron todos los alumnos." Además, si el usuario de la aplicación es un alumno de la escuela, puede hacer un seguimiento de los correos que recibe por parte de los profesores de la escuela de música. Hasta ahora, se trataba de correo electrónico ordinario, que los profesores enviaban a través de Gmail. En el proyecto de gestión, se ha implementado una funcionalidad que permite enviar correos directamente desde la aplicación web de gestión. De esta forma,

además de recibir el mensaje por correo electrónico, éste se guarda en la base de datos de la escuela de música y permite a la aplicación móvil acceder a los mensajes de cada alumno y mostrarlos en ella.

1.2. Objetivos del proyecto

El objetivo de este proyecto es la creación de una aplicación móvil para Android e iOS con las herramientas de Xamarin, que muestre información desde otra aplicación de gestión de una escuela de música creada con ASP.NET CORE. En particular, la aplicación móvil debe mostrar información sobre las novedades de la escuela de música y los datos de cada alumno en su área privada.

Se espera que un alumno pueda iniciar sesión en su cuenta desde la aplicación móvil con credenciales proporcionadas por la misma escuela, pueda ver las novedades de la escuela de música y su agenda, además de tener acceso a su área personal y a mensajes/correos electrónicos recibidos de parte de los profesores. Asimismo, un usuario de la aplicación que no forma parte del alumnado de la escuela de música, debe poder registrarse en la aplicación y, posteriormente, iniciar sesión y acceder a las novedades y a la agenda de la escuela.

1.3. Estructura de la memoria

La memoria se estructura en seis capítulos. En el capítulo 1 se expone el contexto y motivación del proyecto, describiendo, *grosso modo*, la empresa y la necesidad que llevó a proponer este proyecto, los objetivos del proyecto así como las expectativas del mismo. En el capítulo 2 se describe en detalle el proyecto. El capítulo 3 se centra en la planificación del proyecto, incluyendo la metodología usada, la estimación de recursos y costes del proyecto y el seguimiento del mismo. El análisis y el diseño del sistema se detallan en el capítulo 4, mientras que en el capítulo 5 se describe la implementación y las pruebas hechas. Finalmente, en el capítulo 6 aparecen las conclusiones personales.

Capítulo 2

Descripción del proyecto

El proyecto consta de dos partes: una de gestión y administración de la escuela de música y otra de mantener informados a todos los implicados, ya sea de forma directa o indirecta. La parte de gestión y administración de la escuela forma un proyecto, una aplicación web con su API, en el que trabaja una alumna y que es desarrollado, a partir de un *template*, al mismo tiempo que la parte que debe mantener informados a los implicados. Esta última parte es una aplicación móvil multiplataforma, para Android e iOS que, a diferencia de la anterior, se desarrolla desde cero.

Como trabajo en la estancia, el alumno debe determinar los requisitos del proyecto, establecer una estimación de costes y de recursos y mantener un seguimiento adecuado en cada momento, además de desarrollar la aplicación.

La aplicación debe permitir al usuario nuevo registrarse y posteriormente iniciar sesión en ella. Si se trata de un usuario alumno de la escuela de música, se le proporciona nombre de usuario y contraseña para iniciar sesión. Dicho esto, está claro que hay dos tipos de usuarios: usuario alumno y usuario no alumno. El usuario no alumno puede ver las últimas novedades y actualizaciones de carácter público de la escuela de música así como su agenda y actividades abiertas al público. El usuario alumno tiene acceso a todo lo que tiene acceso el usuario no alumno, pero además, también puede hacer un seguimiento de los mensajes que recibe por parte de los profesores de la escuela. También se espera que la aplicación permita el uso de notificaciones *push* para informar acerca de novedades, actualizaciones y mensajes enviados por la escuela de música. Teniendo todo esto en cuenta, está claro que la aplicación móvil debe comunicarse con la aplicación de gestión y administración de la escuela de música, ya que debe acceder a su base de datos y debe hacerlo mediante *endpoints*[1].

Por decisión de la empresa, las tecnologías usadas en el desarrollo del proyecto perte-

[1]Se considera *endpoint* cualquier dirección URL disponible de forma pública en la API del proyecto de gestión y administración de la escuela de música y que ofrezca cualquier tipo de servicio: acceso a datos y opciones de modificarlos, borrarlos o añadir otros nuevos.

necen a Microsoft. La aplicación móvil ha sido desarrollada con herramientas proporcionadas por Xamarin, usando como IDE Visual Studio Community 2017. La definición de los *sprints*, su seguimiento y la colaboración entre los miembros del equipo se ha hecho a través de VSTS (Visual Studio Team Services), que proporciona un entorno integrado y colaborativo con soporte para Git, que se usa para el control de versiones del código fuente, integración continua y herramientas ágiles para la planificación y el seguimiento del trabajo. En la implementación de notificaciones *push* se ha hecho uso del servicio Azure Notification Hubs, que permite el envío de millones de notificaciones a dispositivos Android, iOS, Windows o Kindle, trabajando con APNs (Apple Push Notification service), GCM (Google Cloud Messaging), WNS (Windows Notification Service) o MPNS (Microsoft Push Notification Service). Para la aplicación móvil solamente ha sido necesario trabajar con APNs para el envío de notificaciones a dispositivos iOS y con GCM y Firebase para el envío de notificaciones a dispositivos Android.

Xamarin [2] es una empresa que fue adquirida por Microsoft y proporciona distintas herramientas para el desarrollo de aplicaciones móviles y de escritorio multiplataforma con código compartido. Dos de las tecnologías que ofrece para el desarrollo de aplicaciones móviles son Xamarin.Forms y Xamarin.Native. En Xamarin.Forms, se diseña una sola interfaz gráfica para todas las plataformas móviles, ya sean Android, iOS o Windows. En cambio, con Xamarin.Native se diseñan las interfaces gráficas para cada plataforma de manera individual. Independientemente de la plataforma usada para programar, el lenguaje utilizado es C#.

Al hacer uso de código compartido, no hace falta escribir dos veces la misma funcionalidad y, de esta forma, se ahorra tiempo. Dicho esto, al crear un proyecto en Xamarin, lo que se crea en realidad es una solución con varios proyectos dentro. En primer lugar, se crea un proyecto en el que se incluye todo el código compartido entre Android e iOS. En segundo lugar, se crea un proyecto para Android que se usa para el desarrollo de la interfaz gráfica de la aplicación y código que no puede ser compartido con otro sistema. Finalmente, se crea el proyecto para iOS, que es el que se usa para crear la interfaz gráfica para dispositivos con sistema operativo iOS. Estos dos proyectos también contienen todos los recursos que se usan para el diseño de la interfaz gráfica, como pueden ser imágenes, o texto informativo que se le muestra al usuario y que depende de cada sistema operativo. Se podría haber optado por el desarrollo nativo de la aplicación para cada sistema operativo, pero eso requería, o bien de más tiempo, o bien de otro equipo de desarrollo para llevar a cabo el proyecto en el plazo establecido.

Se ha optado por usar Xamarin.Native y no Xamarin.Forms porque permite diseñar las interfaces gráficas de las aplicaciones de forma individual para cada tipo de sistema operativo móvil. De esta forma, las interfaces gráficas de la aplicación han sido diseñadas de forma independiente para Android y para iOS, pero compartiendo funcionalidad. El lenguaje de programación que usa Xamarin es C#. En Xamarin.Forms, la interfaz gráfica se diseña haciendo uso de un lenguaje etiquetado, muy parecido a xml, pero que tiene como extensión de archivo *.xaml*. En cambio, en Xamarin.Native, la interfaz gráfica para iOS se diseña de forma nativa, al igual como se hace desde Xcode, haciendo uso de un storyboard. En Android, la interfaz puede diseñarse desde un diseñador o directamente

haciendo uso del lenguaje etiquetado de Android, el xml, adaptado para su uso en Xamarin y con C#, que tiene como extensión de fichero *.axml*. Independientemente del sistema operativo, Xamarin también proporciona la opción de crear las interfaces gráficas de forma programable, directamente desde el controlador de la interfaz, es decir, usando C# y sin necesidad de usar el storyboard en iOS o el diseñador en Android.

Para entender mejor cómo se diseña una interfaz gráfica mediante el uso del lenguaje etiquetado o directamente desde C#, el listado 2.1 y el listado 2.2 muestran el ejemplo de la vista con la declaración de privacidad de datos de la aplicación, en C# y en código etiquetado respectivamente. El código etiquetado pertenece a la vista de la aplicación Android, mientras que el código C# pertenece a la vista de la aplicación iOS. El resultado es casi el mismo y se puede ver en las figuras 2.1 y 2.2.

```
1  using Foundation;
2  using System;
3  using UIKit;
4  using WebKit;
5
6  namespace WaterPCL.iOS
7  {
8      public partial class MoreDataViewController : UIViewController
9      {
10         public string Html;
11
12         public MoreDataViewController (IntPtr handle) : base
               (handle) { }
13
14         public override void ViewDidLoad()
15         {
16             base.ViewDidLoad();
17             WKWebView webView = new WKWebView(View.Frame, new
                   WKWebViewConfiguration());
18             View.AddSubview(webView);
19
20             webView.LoadHtmlString(Html, null);
21         }
22     }
23 }
```

Listing 2.1: Diseño de una interfaz nativa para iOS en Xamarin.Native usando el lenguaje C#.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent">
6      <Toolbar
7          android:minHeight="?android:attr/actionBarSize"
8          android:background="?android:attr/colorPrimary"
```

```
9           android:minWidth="25px"
10          android:layout_width="match_parent"
11          android:layout_height="wrap_content"
12          android:id="@+id/toolbarPrivacidad"
13          android:title="Privacidad"
14          android:theme="@android:style/
15                  ThemeOverlay.Material.Dark.ActionBar"
16          android:elevation="4dp" />
17      <RelativeLayout
18          android:minWidth="25px"
19          android:minHeight="25px"
20          android:layout_width="match_parent"
21          android:layout_height="match_parent"
22          android:id="@+id/privacidadLayout"
23          android:background="@android:color/white"
24          android:layout_centerHorizontal="true"
25          android:layout_below="@id/toolbarPrivacidad">
26          <ProgressBar
27              style="?android:attr/progressBarStyleSmall"
28              android:layout_width="wrap_content"
29              android:layout_height="wrap_content"
30              android:id="@+id/progressBarPrivacidad"
31              android:layout_centerVertical="true"
32              android:layout_centerHorizontal="true"
33              android:visibility="invisible" />
34          <android.webkit.WebView
35              android:layout_width="fill_parent"
36              android:layout_height="fill_parent"
37              android:id="@+id/webViewPrivacidad"
38              android:background="@android:color/background_light"
39              android:layout_gravity="center"
40              android:layout_below="@id/relative_buttons"
41              android:layout_centerInParent="true"
42              android:layout_centerHorizontal="true"
43              android:layout_centerVertical="true"
44              android:textAlignment="center" />
45      </RelativeLayout>
46  </LinearLayout>
```

Listing 2.2: Diseño de una interfaz nativa de Android en Xamarin.Native usando el lenguaje de etiquetas *xml*.

Figura 2.1: Pantalla con la declaración de privacidad de datos para la aplicación Android.

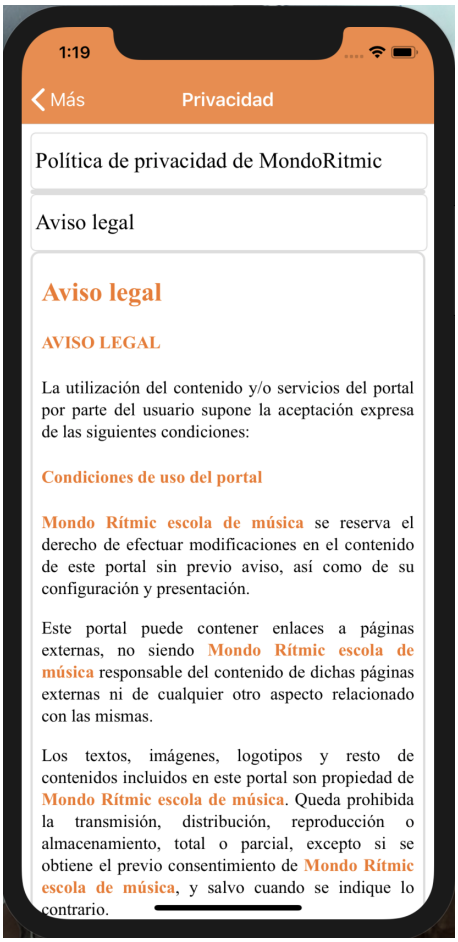

Figura 2.2: Pantalla con la declaración de privacidad de datos para la aplicación iOS.

Capítulo 3

Planificación del proyecto

3.1. Metodología

La metodología usada por la empresa en el desarrollo de sus proyectos es la ágil, definiendo *sprints* y haciendo reuniones formales con el cliente, quien en este proyecto es el director de la escuela de música, aproximadamente cada quince días, exponiéndole los avances hasta el momento, los problemas surgidos, la solución planteada para resolverlos y lo que se espera conseguir hasta la siguiente reunión. Es muy parecida a la metodología Scrum, exceptuando el hecho de que no hay reuniones diarias del equipo de trabajo. De vez en cuando, se hacen reuniones informales del equipo de trabajo y, en ocasiones, también participa el cliente.

Para la planificación del proyecto, la empresa dispone de una serie de requisitos proporcionados por el cliente. Estos requisitos son evaluados y redefinidos en detalle para asegurar qué es lo que el cliente quiere realmente. En este paso, pueden surgir dudas por parte del equipo de trabajo que se deben solucionar junto con el cliente, que al mismo tiempo, puede querer introducir requisitos nuevos o modificar alguno de los existentes. Una vez acordados los requisitos del proyecto entre el cliente y el equipo de trabajo, se procede a la planificación del proyecto.

3.2. Planificación

En la figura 3.1 podemos ver la lista de tareas y la planificación temporal del proyecto. Se puede observar que la fecha de finalización es el día 8 de junio de 2018. Esto se debe a que este proyecto debe coordinar su desarrollo con otro proyecto del que depende (aplicación de gestión de la escuela de música en ASP.NET CORE). El horario usado para el cálculo del tiempo es de cuatro horas diarias de lunes a viernes con dos períodos

festivos: el primero desde el 5 de marzo 2018 hasta el 11 de marzo de 2018 (fiestas de la Magdalena) y el segundo desde el 29 de marzo 2018 hasta el 4 de abril de 2018, fechas de inicio y fin incluidas.

La planificación temporal del proyecto se ha hecho a petición de la coordinadora de estancias en prácticas y ha sido revisada por el tutor de la universidad. Esta planificación está presente en la mayoría de los Trabajos de Final de Grado y tiene como objetivo estructurar, de algún modo, las tareas que el alumno debe realizar durante su estancia en prácticas y tener el control sobre el número total que horas que el alumno debe cumplir, que son trescientas. En la empresa, esto no se tiene mucho en cuenta, pues la metodología usada para el desarrollo de los proyectos es la ágil.

Tratándose de una metodología ágil, los requisitos pueden cambiar a petición del cliente, siendo éste el director de la escuela de música. Para prevenir que ocurran eventos inesperados, se hace un seguimiento de calidad y de recursos humanos durante todo el ciclo del proyecto. Hacer el seguimiento de los recursos humanos es muy fácil, ya que el equipo de desarrollo está formado solamente por un programador, que es el alumno, y por el supervisor de la empresa, que tiene el papel de consultor. En cambio, hacer el seguimiento de calidad es algo más complicado. En la empresa decidimos que el control de calidad se hace con el cliente, a quien se le muestra el progreso de la aplicación y según el caso, el cliente actúa como usuario y prueba la aplicación, aportando *feedback*.

En el capítulo 2, aparecen descritos los requisitos del producto. Inicialmente, mostrar la agenda de actividades de la escuela de música no formaba parte de los requisitos. Se ha incorporado posteriormente, en la fase de desarrollo, a petición del cliente. Esto supuso un cambio en la planificación temporal ya que no estaba previsto ni contabilizado en la duración de la etapa de desarrollo del producto. Para poder incorporar este cambio, hubo que reducir la duración de otras tareas, entre ellas, lo relacionado con la implementación de notificaciones *push* y las pruebas de usuario. Debido a este cambio, no se ha podido completar la implementación de notificaciones *push* en ambos sistemas operativos, Android e iOS, y solamente quedó implementada de forma parcial para la aplicación móvil de Android. Del mismo modo, no se realizó la entrega formal al cliente, debido a que el producto no está completamente finalizado, quedando este último paso para ser realizado en el futuro, por parte de la empresa.

En la figura 3.2 se puede observar la estimación temporal inicial y las dependencias entre tareas. Como es normal, se hace un seguimiento de gestión de la calidad y de los recursos humanos toda la duración del proyecto. Otra cosa a tener en cuenta es que los subapartados de la **Fase de construcción: a)** Desarrollo del software y **b)** Pruebas, coinciden en duración porque a medida que se va desarrollado el software se van creando y ejecutando pruebas.

Se ha intentado planificar la duración de las tareas de tal forma que el tiempo dedicado a cada una sea el adecuado y el inicio de algunas de las fases del proyecto pueda coincidir con las reuniones quincenales con el cliente. De esta forma, la **Fase de comunicación** tiene una duración de quince días laborables, con la mayoría del tiempo dedicado a la

formación de la plataforma Xamarin. Para la segunda reunión con el cliente se agruparon las fases de planificación y modelado del proyecto, con una duración de tres semanas. La fase de planificación está formada por la de gestión del tiempo y de costes, mientras que la fase de modelado consiste en establecer los requisitos del producto, hacer su análisis (diagrama de clases, etc.) y el diseño de la base de datos local y de prototipos de interfaces gráficas. Durante la fase de construcción del producto se celebraron más reuniones, en las que el cliente ya podía ver la forma que estaba tomando la aplicación.

Como se ha comentado anteriormente, esta planificación no llegó a completarse al 100 % debido a cambios imprevistos en los requisitos por parte del cliente. También tuvo influencia el tiempo estimado para la implementación de las notificaciones *push*, ya que se demostró no ser el que realmente se necesitaba, siendo la estimación del tiempo necesario más pequeña. Esto pudo ser causado por la falta de experiencia en la planificación y porque no se tuvo en cuenta desde el principio que dichas notificaciones deben ser implementadas de forma separada para cada sistema operativo móvil. A pesar de eso, se estima que el producto fue completado en un 90 %.

3.3. Estimación de recursos y costes del proyecto

Teniendo en cuenta que el equipo de desarrollo estaba formado por el alumno, quien era el programador y por el supervisor de la empresa, que tenía el rol de consultor, el coste del proyecto se ha calculado de la siguiente manera:

1. El número total de horas, que son trescientas, se ha multiplicado con el salario medio de un programador junior, que está alrededor de los 7 euros por hora. De este modo se consigue un total de 2100 euros.

2. Al valor anterior, se le suma el 10 % del número total de horas, que asciende a 30 horas, multiplicado por el salario de un consultor, que puede ascender a los 12 euros la hora, obteniendo así, 360 euros. Por tanto, el importe total asciende a 2460 euros.

3. A cualquier proyecto informático hay que sumarle entre el 20 % y el 30 % del importe para cubrir posibles desviaciones en la planificación. Entonces, tratándose de un proyecto pequeño, se asume el 20 % de desviaciones y se llega a un importe total de 2952 euros.

4. Para que una aplicación móvil sea usada por cualquier persona, hay que publicarla en las distintas tiendas online, en este caso, la tienda de Google, Play Store, y la de Apple, App Store. Esto supone aún más costes, pues para publicar nuestra aplicación en Play Store tenemos que realizar un pago único de 25 dólares, que son aproximadamente 22 euros, mientras que la publicación en App Store nos cuesta 99 dólares al año, que son aproximadamente 85 euros al año. Con estos costes adicionales, el presupuesto del proyecto asciende a 3060 euros.

5. Servicios contratados a Microsoft: Notification Hubs, Visual Studio Team Services y Application Insights para el desarrollo de aplicaciones móviles asciende a 10 euros al mes la versión más ligera, siendo el total de 40 euros para los cuatro meses. Sumando este importe al presupuesto, éste asciende a 3100 euros.

6. El software usado en el desarrollo, en concreto Visual Studio Community 2017 y Xamarin es gratis, no suponiendo coste alguno.

7. Además de los servicios contratados, hay que sumar el coste del hardware. Un ordenador portátil de gama media como el que se usó durante el desarrollo cuesta alrededor de 800 euros. Este importe se debe distribuir entre todos los proyectos, pues no solamente se ha usado para este proyecto. Para decidir el importe que se le suma a cada proyecto por el uso del hardware, se ha optado por calcularlo usando la duración de cada proyecto, durante un año. De esta forma, el coste mensual de uso del ordenador es de aproximadamente 67 euros. Como el proyecto tuvo una duración de cuatro meses a media jornada, el importe total que hay que sumarle al proyecto es de 134 euros, pues cuatro meses de trabajo a media jornada equivalen a dos meses de trabajo a jornada completa. En este momento el presupuesto asciende a 3234 euros.

8. Las pruebas de usuario durante el desarrollo se han hecho en los emuladores de Android e iOS, mientras que las pruebas hechas durante las reuniones con el cliente se han hecho en su terminal móvil, por lo que esto no supone coste alguno para el proyecto. De lo contrario, si hubiera sido necesario adquirir los terminales para hacer las pruebas en Android y en iOS, el coste íntegro de estos terminales se añadirían al proyecto. Un terminal con sistema operativo Android de gama media-baja cuesta alrededor de 200 euros, mientras que un terminal móvil con sistema operativo iOS nos cuesta unos 300 euros.

3.4. Seguimiento del proyecto

Como se ha indicado anteriormente, se trata de un proyecto cuyo desarrollo ha seguido la metodología ágil. Para tener control sobre el proyecto y las desviaciones que han podido aparecer, se han definido *sprints* con duraciones de entre dos y tres semanas según la carga de trabajo y se han realizado reuniones con el cliente al finalizar cada *sprint*. De este modo, el equipo de desarrollo controlaba en todo momento la dirección en la que iba el proyecto y el cliente estaba informado del progreso. En las reuniones con el cliente se trataba de ponerlo al día del progreso realizado hasta ese momento incluyendo información acerca de los *sprints*, en concreto, el número de tareas completadas, las tareas que no se pudieron completar y el por qué, los problemas surgidos y las medidas llevadas a cabo para solucionarlas, si fuera el caso y se le mostraba el estado de la aplicación, que en ocasiones podía probar él mismo. Además, se le informaba de las tareas del siguiente *sprint* y lo que se pretendía conseguir.

La primera reunión tuvo lugar el día 16 de marzo de 2018. En ella se han expuesto el

estado y el cronograma del proyecto. La segunda reunión se celebró el día 29 de marzo de 2018, teniendo como objetivo informar de los costes del proyecto y decidir sobre el diseño de la aplicación usando *mocks*. A partir de la tercera reunión que tuvo lugar el día 16 de abril de 2018, las reuniones se han basado en mostrar al cliente el estado de la aplicación, lo que se ha conseguido, los problemas surgidos y las soluciones planteadas, además de informarle de lo que se espera conseguir hasta la siguiente reunión. La cuarta reunión ha sido el día 30 de abril de 2018. Finalmente, la última reunión se ha celebrado el día 30 de mayo de 2018. Aún tratándose de la última reunión, no hubo una entrega oficial del producto al cliente ni su puesta en marcha, quedando estas dos tareas pendientes, pues la aplicación no estaba completamente finalizada.

En la sección 2 de este capítulo se menciona que ha habido cambios en los requisitos del proyecto. Se trata de la introducción de una nueva vista para la agenda de la escuela de música, requisito que ha sido introducido por el cliente y director de la escuela después de ver implementada la parte de novedades. Este cambio afectó a la planificación inicial del *sprint* y del proyecto, pues no fue contemplado desde el inicio. Para ello, se tuvo que reducir el tiempo de desarrollo para la implementación de notificaciones, cosa que impidió terminar dicha implementación al 100 %, quedando implementadas solamente las notificaciones de tipo *broadcast*, es decir, notificaciones que llegan a todos los usuarios, en dispositivos Android, sin poder implementar notificaciones individuales.

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**Prácticas externas**	**300 hrs**	**Mon 12/02/18**	**Fri 08/06/18**	
2		Gestión de calidad	300 hrs	Mon 12/02/18	Fri 08/06/18	
3		Gestión de recursos humanos	300 hrs	Mon 12/02/18	Fri 08/06/18	
4		**Fase de comunicación**	**60 hrs**	**Mon 12/02/18**	**Fri 02/03/18**	
5		Estudio del caso	8 hrs	Mon 12/02/18	Tue 13/02/18	
6		Formación	40 hrs	Wed 14/02/18	Tue 27/02/18	5
7		Definir alcance y objetivos	8 hrs	Wed 28/02/18	Thu 01/03/18	6
8		Identificación de riesgos	4 hrs	Fri 02/03/18	Fri 02/03/18	7
9		**Fase de planificación**	**16 hrs**	**Mon 12/03/18**	**Thu 15/03/18**	
10		**Gestión del tiempo**	**8 hrs**	**Mon 12/03/18**	**Tue 13/03/18**	
11		Planificación temporal	8 hrs	Mon 12/03/18	Tue 13/03/18	8
12		**Gestión de costes**	**8 hrs**	**Wed 14/03/18**	**Thu 15/03/18**	
13		Aproximación del presupuesto	8 hrs	Wed 14/03/18	Thu 15/03/18	11
14		**Fase de modelado**	**44 hrs**	**Mon 12/03/18**	**Mon 26/03/18**	
15		**Establecer requisitos**	**8 hrs**	**Mon 12/03/18**	**Tue 13/03/18**	
16		Diagrama de casos de uso	8 hrs	Mon 12/03/18	Tue 13/03/18	8
17		Historias de usuario	8 hrs	Mon 12/03/18	Tue 13/03/18	8
18		**Análisis**	**8 hrs**	**Wed 14/03/18**	**Thu 15/03/18**	
19		Diagrama de clases	8 hrs	Wed 14/03/18	Thu 15/03/18	15
20		Diagrama de actividades	8 hrs	Wed 14/03/18	Thu 15/03/18	15
21		**Diseño**	**28 hrs**	**Fri 16/03/18**	**Mon 26/03/18**	
22		Diseño de bases de datos	8 hrs	Fri 16/03/18	Mon 19/03/18	18
23		Diseño de interfaces gráficas	28 hrs	Fri 16/03/18	Mon 26/03/18	18
24		**Fase de construcción**	**188 hrs**	**Tue 27/03/18**	**Wed 06/06/18**	
25		**Desarrollo del software**	**188 hrs**	**Tue 27/03/18**	**Wed 06/06/18**	
26		Implementación del diagrama de clases	16 hrs	Tue 27/03/18	Fri 06/04/18	14
27		Diseño de interfaces de comunicación RESTful	20 hrs	Mon 09/04/18	Fri 13/04/18	26
28		Implementación de las interfaces de comunicación RESTful	48 hrs	Mon 16/04/18	Tue 01/05/18	27
29		Desarrollo modular	104 hrs	Wed 02/05/18	Wed 06/06/18	28
30		**Pruebas**	**188 hrs**	**Tue 27/03/18**	**Wed 06/06/18**	
31		Creación de pruebas	188 hrs	Tue 27/03/18	Wed 06/06/18	14
32		Ejecución de pruebas	188 hrs	Tue 27/03/18	Wed 06/06/18	14
33		**Fase de puesta en marcha**	**8 hrs**	**Thu 07/06/18**	**Fri 08/06/18**	
34		Pruebas de usuario	8 hrs	Thu 07/06/18	Fri 08/06/18	24
35		Entrega	0 days	Fri 08/06/18	Fri 08/06/18	34

Figura 3.1: Desglose de tareas y planificación temporal inicial de éstas, que debe realizar el alumno durante su estancia en prácticas.

Figura 3.2: Diagrama de Gantt y la estimación temporal del proyecto.

Capítulo 4

Análisis y diseño del sistema

4.1. Análisis del sistema

Una forma excelente de hacer el análisis de un sistema informático es mediante diagrama de casos de uso, diagrama de clases, historias de usuario, etc. Aplicando esta metodología en el proyecto, se ha conseguido establecer los requisitos del producto. La figura 4.1 muestra el diagrama de casos de uso de la aplicación desarrollada. En ella aparecen tres actores: un usuario general que es capaz de registrarse, otro usuario alumno extiende al usuario general y que puede ver los mensajes que le han enviado los profesores de la escuela de música. Además de las acciones que puede hacer cada uno, ambos pueden iniciar sesión, ver las novedades y las actividades de la escuela de música y recibir notificaciones. El tercer actor no es una persona, sino un sistema externo, la aplicación de gestión y administración de la escuela de música, que se encarga de enviar notificaciones a los dispositivos móviles y que se encarga de establecer la comunicación de la aplicación móvil con la base de datos de la escuela de música, proporcionando enlaces de registro, inicio de sesión y acceso a los mensajes de cada alumno.

El diagrama de casos de uso no es suficiente por sí solo para establecer los requisitos del sistema de forma adecuada. Es por ello que el diagrama de casos de uso se amplía usando una tabla de descripción de casos de uso. Las tablas de la 4.1 a la 4.6 describen en más detalle los casos de uso que aparecen en la figura 4.1. Además del diagrama de casos de uso y de las tablas de descripción de los casos de uso se han utilizado también las historias de usuario, que aparecen en la tabla 4.7. En total hay cinco historias de usuario que expresan las principales acciones que los usuarios quieren hacer.

4.2. Diseño de la arquitectura del sistema

Al tratarse de una aplicación móvil con interfaz de usuario, la arquitectura que mejor se adapta es la de tres capas, formada por el modelo, la vista y el controlador, cuyo diseño se puede ver en la figura 4.2.

En este proyecto, la lógica del programa se encuentra en el proyecto compartido por Android e iOS, mientras que los controladores y las vistas están situadas cada una en el proyecto de Android o iOS, según corresponde.

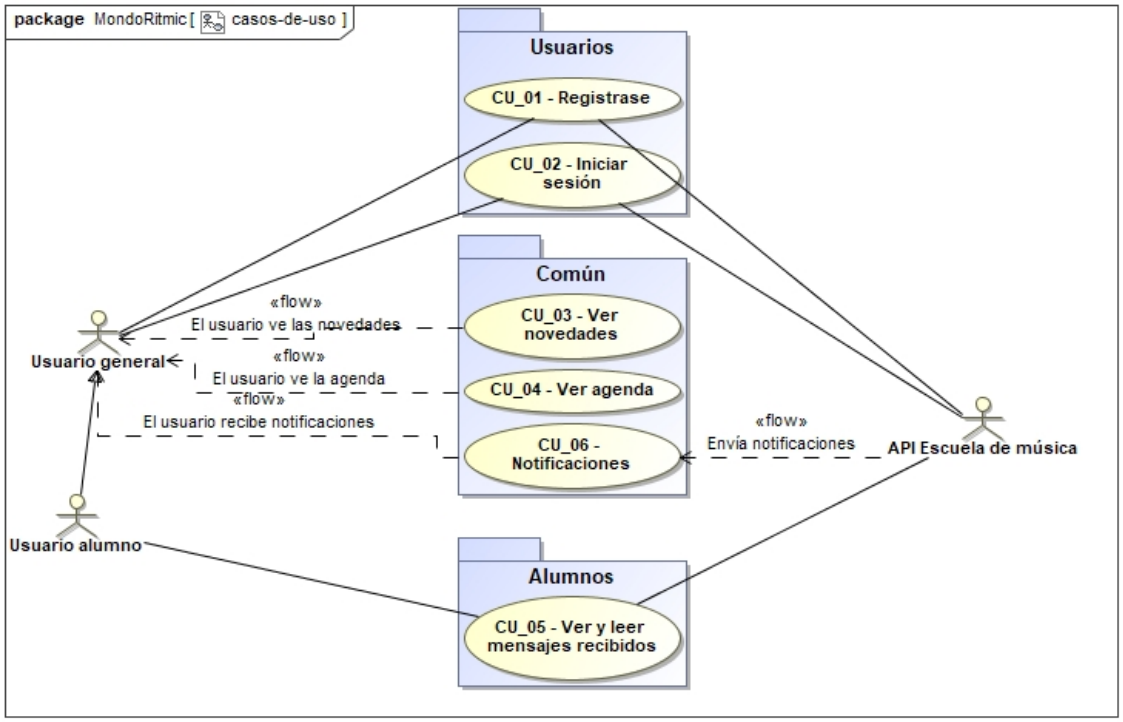

Figura 4.1: Diagrama de casos de uso.

4.3. Diseño de la interfaz

La interfaz de usuario ha sido diseñada de forma aproximada haciendo uso de *mocks*, usando la plataforma de NinjaMock [5]. En las figuras de la 4.3 a la 4.6 se pueden observar prototipos de interfaces gráficas de usuario. Estos prototipos han sido refinados y adaptados a cada plataforma móvil (Android e iOS) para tener consistencia con las directrices proporcionadas por Apple [3] y por Google [4] acerca del aspecto y diseño que deben tener las aplicaciones en sus tiendas.

Caso de uso: CU_01 - Registrarse			
Identificador:	CU_01	**Fecha creación:**	12 de marzo de 2018
Nombre:	Registrarse	**Fecha revisión:**	13 de marzo de 2018
Autor:	Cristian Ionut Valeanu	**Fecha aprobación:**	13 de marzo de 2018
Origen:	Usuario general	**Versión:**	1.0

Descripción: El usuario que no forma parte de la escuela de música se debe registrar en el sistema para poder usar la aplicación.

Secuencia de pasos:

1. Abrir la aplicación.
2. Presionar el botón "Registrarse".
3. Introducir el nombre completo.
4. Introducir una dirección de correo electrónico válida.
5. Presionar de nuevo el botón "Registrarse".

Excepción:

1. Se muestra un mensaje de error si el correo introducido no tiene un formato válido (`ejemplo@correo.es`).
2. Se muestra un mensaje de error si no hay conexión a internet.

Condición previa: Instalar la aplicación en el terminal móvil desde el cual se quiere acceder.

Comentarios: Los usuarios alumnos de la escuela de música no necesitan registrarse ya que la misma escuela les proporciona un nombre de usuario y una contraseña para iniciar sesión en el sistema.

Cuadro 4.1: Descripción del caso de uso *CU_01 - Registrarse*.

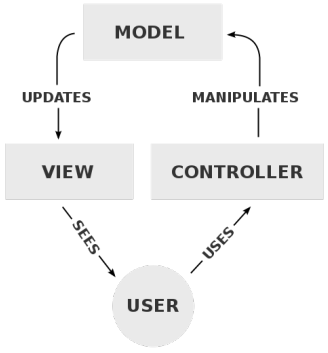

Figura 4.2: Diseño de una arquitectura MVC de tres capas [1].

Caso de uso: CU_02 - Iniciar sesión			
Identificador:	CU_02	**Fecha creación:**	12 de marzo de 2018
Nombre:	Iniciar sesión	**Fecha revisión:**	13 de marzo de 2018
Autor:	Cristian Ionut Valeanu	**Fecha aprobación:**	13 de marzo de 2018
Origen:	Usuario general y usuario alumno	**Versión:**	1.0

Descripción: Los usuarios deben iniciar sesión para poder usar la aplicación.

Secuencia de pasos:

1. Abrir la aplicación.
2. Introducir nombre de usuario y contraseña.
3. Presionar el botón "Iniciar sesión".
4. El sistema verifica las credenciales introducidas, y si son correctas, redirecciona al usuario a la pantalla principal de la aplicación.

Excepción:

1. Se muestra un mensaje de error si no hay conexión a internet.
2. Se muestra un mensaje de error si el usuario no introduce un nombre de usuario y una contraseña.
3. Se muestra un mensaje de error si las credenciales introducidas no corresponden con las de ningún usuario.
4. Se muestra un mensaje de error si el servidor falla al comprobar los datos.

Condición previa: Para poder iniciar sesión, un usuario debe haberse registrado previamente, o ser alumno con una suscripción activa de la escuela de música.

Comentarios:

Cuadro 4.2: Descripción del caso de uso *CU_02 - Iniciar sesión*.

Caso de uso: CU_03 - Ver novedades			
Identificador:	CU_03	**Fecha creación:**	12 de marzo de 2018
Nombre:	Ver novedades	**Fecha revisión:**	13 de marzo de 2018
Autor:	Cristian Ionut Valeanu	**Fecha aprobación:**	13 de marzo de 2018
Origen:	Usuario general y usuario alumno	**Versión:**	1.0

Descripción: Todos los usuarios autenticados pueden ver las novedades de la escuela de música.

Secuencia de pasos:

1. Abrir la aplicación.
2. Iniciar sesión.
3. Por defecto, una vez iniciada la sesión, la aplicación muestra la pestaña de novedades.

Excepción:

1. Se muestra un mensaje de error si no hay conexión a internet.

Condición previa: El usuario debe iniciar sesión.

Comentarios:

Cuadro 4.3: Descripción del caso de uso *CU_03 - Ver novedades*.

Caso de uso: CU_04 - Ver agenda			
Identificador:	CU_04	**Fecha creación:**	12 de marzo de 2018
Nombre:	Ver novedades	**Fecha revisión:**	13 de marzo 2018
Autor:	Cristian Ionut Valeanu	**Fecha aprobación:**	13 de marzo 2018
Origen:	Usuario general y usuario alumno	**Versión:**	1.0

Descripción: Todos los usuarios autenticados pueden ver la agenda y los eventos de carácter público de la escuela de música.

Secuencia de pasos:

1. Abrir la aplicación.
2. Iniciar sesión.
3. Seleccionar la pestaña “Agenda”.

Excepción:

1. Se muestra un mensaje de error si no hay conexión a internet.

Condición previa: El usuario debe iniciar sesión.

Comentarios:

Cuadro 4.4: Descripción del caso de uso *CU_04 - Ver agenda*.

	Caso de uso: CU_05 - Ver y leer mensajes recibidos		
Identificador:	CU_05	**Fecha creación:**	12 de marzo de 2018
Nombre:	Ver y leer mensajes recibidos	**Fecha revisión:**	13 de marzo de 2018
Autor:	Cristian Ionut Valeanu	**Fecha aprobación:**	13 de marzo de 2018
Origen:	Usuario alumno	**Versión:**	1.0
Descripción: Solamente los alumnos de la escuela de música tienen acceso a mensajes.			
Secuencia de pasos: 1. Abrir la aplicación. 2. Iniciar sesión. 3. Seleccionar la pestaña “Mensajes”.			
Excepción: 1. Se muestra un mensaje de error si no hay conexión a internet.			
Condición previa: El usuario debe iniciar sesión.			
Comentarios:			

Cuadro 4.5: Descripción del caso de uso *CU_05 - Ver y leer mensajes recibidos.*

Caso de uso: CU_06 - Ver notificaciones			
Identificador:	CU_06	**Fecha creación:**	12 de marzo de 2018
Nombre:	Ver novedades	**Fecha revisión:**	13 de marzo de 2018
Autor:	Cristian Ionut Valeanu	**Fecha aprobación:**	13 de marzo de 2018
Origen:	Usuario general y usuario alumno	**Versión:**	1.0
Descripción: Todos los usuarios que se han instalado la aplicación y han iniciado sesión reciben notificaciones relacionadas con la escuela de música.			
Secuencia de pasos: Se debe hacer al menos una vez: 1. Abrir la aplicación. 2. Iniciar sesión. **Excepción:** 1. Se muestra un mensaje de error si no hay conexión a internet. **Condición previa:** El usuario debe iniciar sesión.			
Comentarios: Una vez que el usuario haya iniciado sesión, la aplicación le muestra las notificaciones recibidas.			

Cuadro 4.6: Descripción del caso de uso *CU_06 - Ver notificaciones*.

HU-01	
COMO	Usuario de la aplicación y NO alumno de la escuela de música
QUIERO	Registrarme en la aplicación proporcionando **email** y **nombre**
PARA	Crear una cuenta.
HU-02	
COMO	Usuario de la aplicación
QUIERO	Iniciar sesión en la aplicación con mi **nombre de usuario** y **contraseña**
PARA	Ver las novedades de la escuela de música.
HU-03	
COMO	Usuario de la aplicación
QUIERO	Iniciar sesión en la aplicación con mi **nombre de usuario** y **contraseña**
PARA	Ver la agenda de la escuela de música.
HU-04	
COMO	Usuario de la aplicación y alumno de la escuela de música
QUIERO	Iniciar sesión en la aplicación con mi **nombre de usuario** y **contraseña**
PARA	Ver mis mensajes privados enviados por profesores de la escuela.
HU-05	
COMO	Usuario de la aplicación
QUIERO	Iniciar sesión en la aplicación con mi **nombre de usuario** y **contraseña**
PARA	Recibir notificaciones por parte de la escuela de música.

Cuadro 4.7: Historias de usuario.

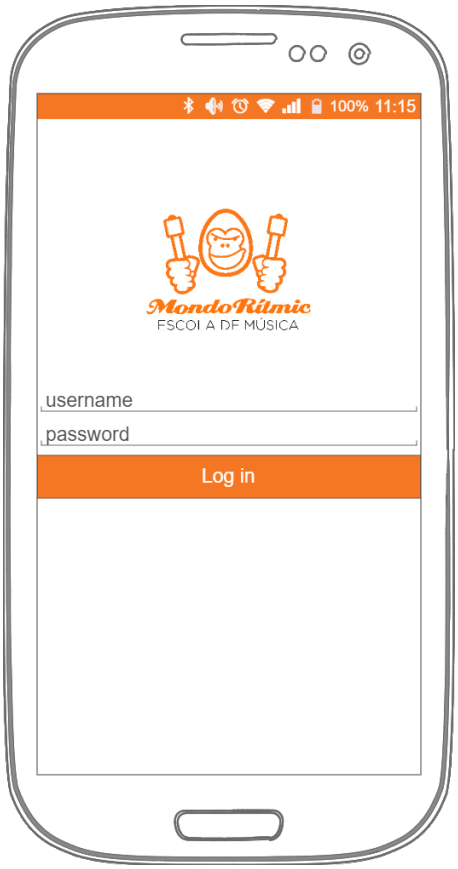

Figura 4.3: Prototipo de una pantalla de inicio de sesión para Android.

Figura 4.4: Prototipo de una pantalla que muestra las novedades de la escuela de música.

Figura 4.5: Prototipo de una pantalla que muestra la lista de mensajes de un alumno de la escuela de música.

Figura 4.6: Prototipo de una pantalla que muestra todo el mensaje recibido por el alumno.

Capítulo 5

Implementación y pruebas

5.1. Detalles de implementación

Como se ha mencionado en el capítulo 2, la aplicación móvil ha sido desarrollada usando herramientas de Xamarin, en concreto, Xamarin.Native. La decisión de usar Xamarin.Native y no Xamarin.Forms la tomó el alumno, pues era quien debía desarrollar la aplicación y quien mejor podía apreciar cuál de las dos tecnologías usar. Se ha optado por usar Xamarin.Native porque las interfaces gráficas se diseñan para cada plataforma móvil de forma nativa, permitiendo al programador adaptar la aplicación a cada sistema operativo de forma más fácil y con más consistencia.

Al comenzar un proyecto Xamarin.Native para el desarrollo de una aplicación móvil multiplataforma en Visual Studio, se crea una estructura de directorios llamada *Solución*, que contiene tres proyectos:

1. El primer proyecto almacena el código y los recursos compartidos por la aplicación en las distintas plataformas. En este proyecto se suelen guardar las clases del modelo y la lógica compartida por la aplicación Android y por la aplicación iOS.
2. El segundo proyecto es el de Android. Éste contiene los recursos específicos de una aplicación Android, como pueden ser las vistas, los controladores de las vistas y los iconos y las imágenes de la aplicación.
3. El tercer proyecto es el de iOS. En él se almacena todo el contenido relacionado con la aplicación para iOS. Suele tratarse del *Storyboard* para el diseño de las vistas, los controladores de las vistas si es el caso, y los iconos e imágenes que se usan en la aplicación.

El lenguaje de programación usado para los controladores de las vistas, las clases del modelo y la lógica de la aplicación móvil en toda la solución es C#. Todas las vistas

han sido desarrolladas de forma nativa. En iOS las vistas se han creado dentro del *Storyboard*, mientras que en Android, todas las vistas son *layouts*, ficheros con extensión *.axml*, situados dentro del directorio *Resources/layout*.

5.1.1. Proyecto compartido

Como ya se ha comentado en la sección anterior, el proyecto compartido contiene las clases del modelo y la lógica común de las aplicación para Android e iOS. Todo lo que no se puede compartir, se sitúa en el proyecto en que se usa. En la figura 5.1 aparece representado el diagrama de clases del proyecto compartido. Como se puede observar, solamente contiene clases del modelo, ya que la lógica se ha implementado en cada proyecto de forma distinta. Exceptuando las clases "Mensaje" y "DatabaseHelper", ninguna contiene métodos o funciones definidas por el programador. Los atributos que aparecen, en C# llevan el nombre de *Properties*. Se trata de atributos públicos con sus *getters* y *setters*. Los métodos públicos de la clase "Mensaje" sirven para poder comparar dos mensajes. Se han sobrescrito los métodos *Equals* y *GetHashCode* para comparar dos objetos de tipo "Mensaje" usando el método *Equals*, como se hace en todos los lenguajes de programación orientados a objetos, pero también se han sobrescrito los operadores "==" y "!=" por si alguien prefiere comparar los objetos usando este método. C# permite comparar objetos usando el método *Equals* o los operadores "==" y "!=". Para usar los operadores a la hora de comparar objetos, hay que sobrescribirlos. De lo contrario, los operadores comparan objetos por referencia. La mayoría de las clases base de C#, exceptuando la clase *object*, han sobrescrito los operadores para hacer comparaciones de esta forma. La clase "DatabaseHelper", como su nombre indica, es una clase que ofrece métodos para insertar y extraer datos de una base de datos SQLite, dentro del dispositivo móvil.

5.1.2. Proyecto Android

La estructura de un proyecto Android en Xamarin.Native es simple:

1. Contiene un directorio llamado *References* y, como su nombre indica, contiene referencias a librerías del sistema operativo Android y al proyecto compartido.

2. El siguiente directorio lleva el nombre de *Packages* y almacena todas las librerías y paquetes adicionales que se necesitan. En este directorio se incluyen las librerías Android proporcionadas por Xamarin, así como librerías de terceros, como pueden ser los servicios de Google para recibir notificaciones *push* a través de Firebase o serializadores y deserializadores de ficheros JSON.

3. El tercer directorio, llamado *Assets*, sirve para introducir recursos que serán disponibles dentro de la aplicación. Por *assets* nos referimos, por ejemplo, a ficheros con tipografías (los *.ttf*), si queremos cambiar el tipo de letra para mostrar el texto dentro de la aplicación.

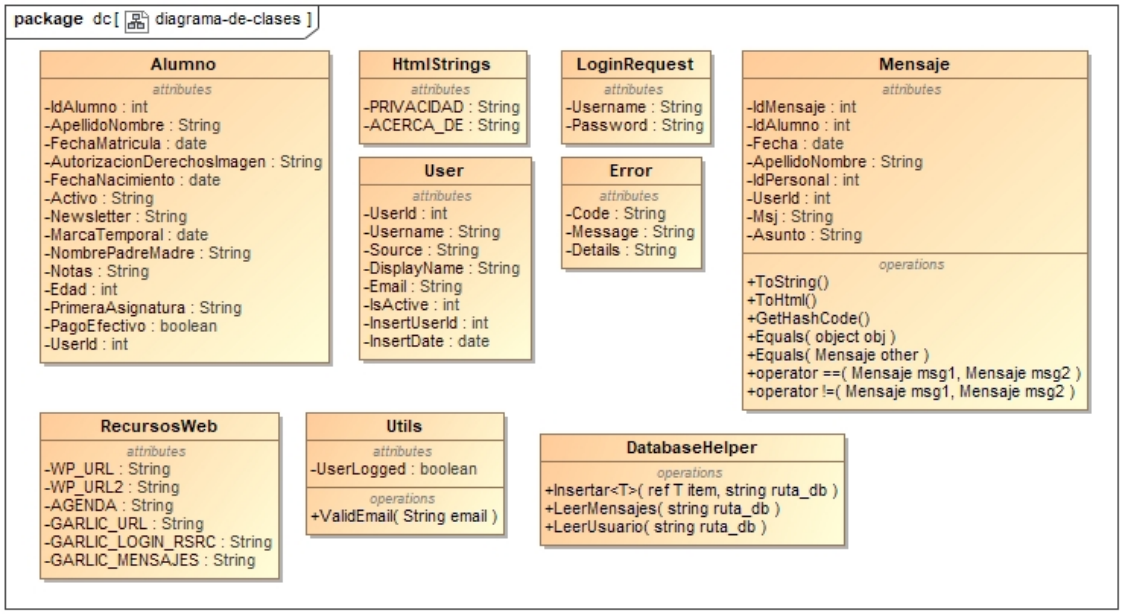

Figura 5.1: Diagrama de clases del proyecto compartido.

4. *Properties* es el nombre del cuarto directorio. En él hay dos ficheros: *AndroidManifest.xml* y *AssemblyInfo.cs*. El primer fichero contiene las especificaciones de la aplicación, indicando entre otras cosas, la versión mínima del sistema operativo en el que la aplicación se puede instalar y usar, la versión recomendada del sistema operativo y los permisos que debe tener la aplicación. El segundo fichero indica cómo compilar el proyecto y qué clases y ficheros se incluyen y los que se excluyen.

5. En el último directorio se guardan todos los recursos que se usan en el desarrollo de la aplicación. Se deben incluir en este directorio las imágenes y los iconos de la aplicación, las vistas y se pueden definir los menús y los colores que se usan.

6. Finalmente, están los controladores de las vistas, que están situados todos al mismo nivel con los directorios anteriores. También se insertan en este nivel, que el más alto dentro de la aplicación Android, otras clases que los desarrolladores necesitan y que no son necesariamente controladores de las vistas.

Dicho esto, en Android, a una vista se le conoce también bajo el nombre de *layout*, mientras que el controlador de la vista recibe el nombre de *Activity*. Todos los *layouts* se sitúan en una carpeta con mismo nombre, dentro de *Resources*, mientras que los *Activities* están en el nivel más alto, junto con los directorios de recursos, *assets* o propiedades.

La aplicación Android que se ha desarrollado está formada por siete vistas, cada una con su controlador:

1. La pantalla de registro de un nuevo usuario.
2. La pantalla de inicio de sesión.
3. La vista que muestra las novedades de la escuela de música.
4. La vista que muestra la agenda de la escuela.
5. La pantalla que muestra la lista de mensajes del alumno.
6. La pantalla que muestra en detalle un mensaje.
7. La pantalla que muestra información de privacidad de datos.

La pantalla de registro de un nuevo usuario y la de inicio de sesión son muy parecidas. Ambas tienen el logo de la escuela de música, dos campos de texto para introducir nombre y apellidos y correo electrónico si se trata de la pantalla de registro o nombre de usuario y contraseña si se trata de la pantalla de *login* y un botón para enviar los datos introducidos al servidor para crear una nueva cuenta o para iniciar sesión. La pantalla de *login* incluye un botón más que lleva al usuario a la pantalla de registro. Las figuras 5.2 y 5.3 muestran el aspecto de estas dos pantallas.

Las siguientes tres pantallas tienen el mismo nivel de importancia. Se trata de las novedades, la agenda y la lista de mensajes. Es por ello que se ha usado una vista con tres pestañas, de una forma parecida a aplicaciones como Facebook o Instagram. Para poder implementar esta vista se ha hecho uso del *FragmentPagerAdapter*. Esto permite insertar, desde los controladores, en un *layout* diferentes fragmentos, donde cada fragmento corresponde a una vista. Para que sea posible insertar los fragmentos en un mismo *layout* se usa *ViewPager*. La selección de las pestañas se hace desde el *TabLayout*. De esta forma, esta pantalla contiene un *toolbar* en la parte superior de la pantalla con dos opciones de menú: una para mostrar la declaración de privacidad de datos y otro para mostrar el información acerca de la escuela de música, un *ViewPager* en medio de la pantalla donde se insertan los *Fragments* y un *Tabbar* que permite seleccionar qué pestaña mostrar.

Para la vista con las novedades y la de la agenda se ha usado el elemento *WebView* de Android, que permite insertar contenido *html* de la misma forma que hace un navegador web. El contenido que se muestra en la pestaña "Novedades" es el de la página web de la escuela de música, mientras que la agenda ha sido introducida usando la etiqueta *iframe* de *html*, en la que se inserta el calendario de Google con las actividades de la escuela. En la figura 5.4 se muestra la pantalla con las pestañas, siendo seleccionada la de "Novedades".

La pantalla que muestra la lista de mensajes se ha implementado usando el *ListView* de Android, al que se le pasa la lista de mensajes obtenida desde la base de datos si

hay conexión a internet, o desde el mismo dispositivo si no la hay. Además de obtener la lista de mensajes desde la base de datos, también se almacena una copia en el dispositivo móvil. Esto se puede observar en la figura 5.5. En cambio, las pantallas que muestran el mensaje completo, figura 5.6, y la información de privacidad de datos e información acerca de la escuela se han hecho usando *WebView*. En los tres casos se inserta un *string* en formato *html* con el contenido del mensaje en el primer caso, con el contenido de la declaración de privacidad de datos en el segundo y con la información de la escuela de música en el tercero.

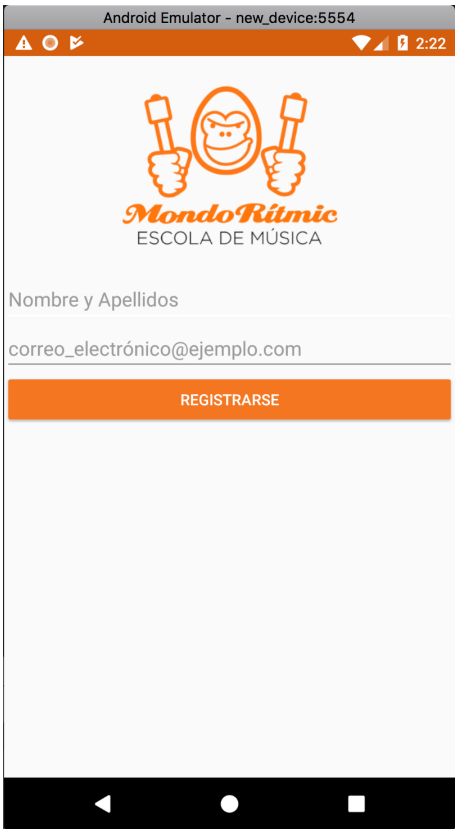

Figura 5.2: Pantalla de registro de la aplicación Android.

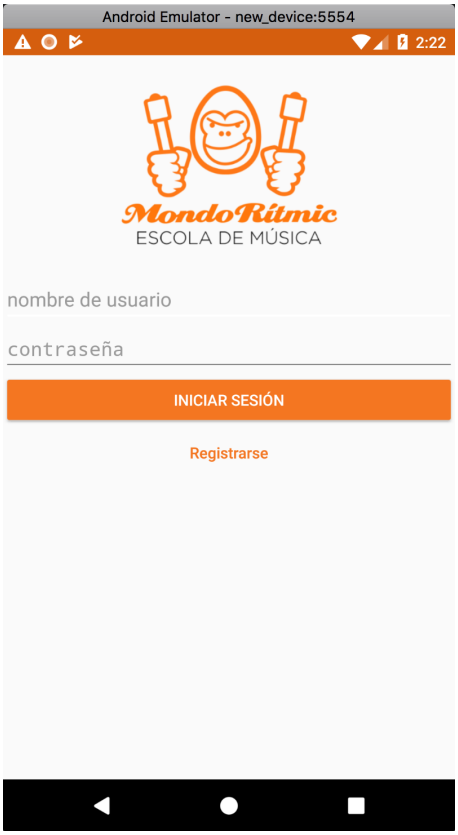

Figura 5.3: Pantalla de inicio de sesión de la aplicación Android.

5.1.3. Proyecto iOS

La estructura de un proyecto iOS dentro de Xamarin.Native es casi idéntica a la del proyecto Android, y es por eso que no entramos en detalle. Hay dos diferencias fundamentales que sí se deben mencionar.

El proyecto iOS no tiene el directorio *Properties*. Para definir las propiedades de la aplicación iOS así como los permisos y versiones del sistema operativo, Apple usa el fichero

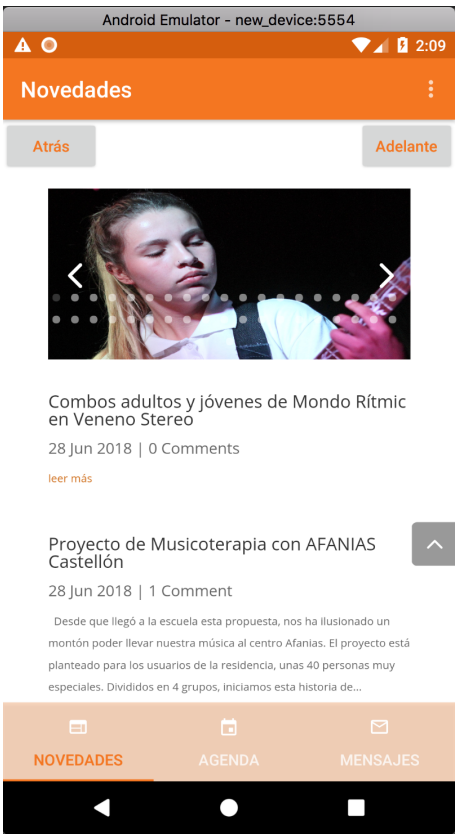

Figura 5.4: Vista que muestra las novedades en la aplicación Android.

Entitlements.plist, que contiene solamente los permisos que se deben dar a la aplicación, y el fichero *Info.plist* que contiene la información restante (versión del sistema operativo, iconos de la aplicación, *launcher*, etc.). Algunos de los permisos que necesita la aplicación se encuentran en este fichero también.

Las vistas de una aplicación iOS no se crean en ficheros separados, sino que todas ellas se diseñan dentro de un *Storyboard*, como el de la figura 5.7. En esta figura se observan diez vistas. En iOS, una vista se genera usando *Controllers*. Estos *Controllers* están separados: una parte se muestra en el *Storyboard* y sirve para crear las vistas y añadir componentes usando el diseñador; mientras que la otra parte son los ficheros *.cs*. Juntado las dos partes es cuando se crea la vista. Las clases de los ficheros *.cs* son, en realidad, *partial class*. También es posible crear una vista sin usar el diseñador, en tiempo de ejecución, desde código C# e insertarla en la pila de navegación cuando se quiere mostrar y quitarla de la pila si se vuelve a la vista anterior.

Para explicar mejor la organización de las vistas, hay que mencionar que de los diez controladores que aparecen en el *Storyboard*, solamente nueve de ellos se usan para generar vistas. El que está situado en el extremo izquierdo es un *NavigationController* y su funcionalidad es la de permitir una navegación fluida en la aplicación a través de los otros controladores. De hecho, el *NavigationController* no tiene una clase asociada en este

Figura 5.5: Pantalla con la lista de mensajes recibidos por el usuario, en la aplicación Android.

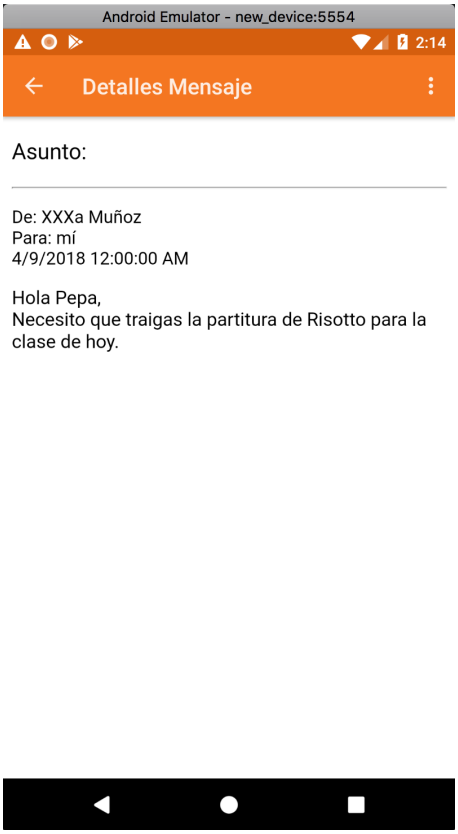

Figura 5.6: Pantalla que muestra un mensaje completo en la aplicación Android.

proyecto. Los demás controladores disponen cada uno de su clase. De los nueve controladores restantes, ocho son *ViewControllers*, mientras que al que le llegan y del cual salen más flechas en un *TabBarController*. A diferencia de la aplicación en Android, donde ha sido necesario crear el comportamiento de la navegación a través de pestañas, en iOS no hace falta hacer tal cosa, ya que Apple nos proporciona el *TabBarController*. De forma muy sencilla, simplemente creando un *ViewController* y enlazándolo desde el *TabBarController* se crea una pestaña. Es en este *ViewController* donde se diseña el aspecto que tiene la vista y se hace igual para cualquiera los controladores.

Las pantallas de registro y de inicio de sesión son idénticas a las del proyecto Android, cada una con el logo, dos campos de texto y el botón de "Registrarse" para la pantalla de registro y los botones "Iniciar sesión" y "Registrarse" en la pantalla de *login*.

Para mostrar la lista de mensajes, en Android se ha usado *ListView*, mientras que en iOS, obtener el mismo comportamiento se consigue usando *TableViewController* y asignando a cada celda el texto que debe mostrar, usando el método *GetCell*. En el listado 5.1 se muestra cómo hacerlo.

```
// ...
public override UITableViewCell GetCell(UITableView tableView,
    NSIndexPath indexPath)
{
        var cell =
            tableView.DequeueReusableCell("mensajeIdentifier",
            indexPath);
        var data = _mensajes[indexPath.Row];

        cell.TextLabel.Text = data.Asunto;
        cell.DetailTextLabel.Text = data.Fh.ToString();

        return cell;
}
// ...
```

Listing 5.1: Fragmento de código en C# que establece el contenido de una celda de *TableViewController* en iOS.

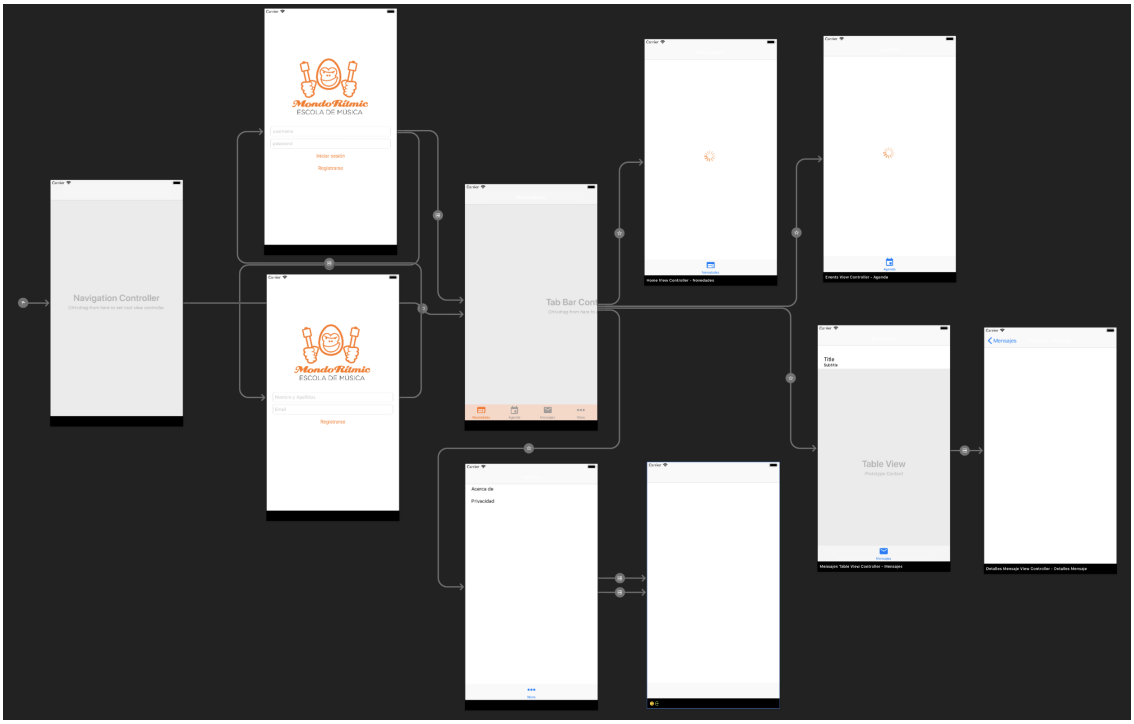

Figura 5.7: Imagen del *Storyboard* y las pantallas de una aplicación iOS.

Las vistas que muestran las novedades, la agenda, el mensaje completo y la política de privacidad de datos se han hecho como en Android, usando *WebView*, pero no el de Android, sino el de Apple, que lo usa también Safari, *WKWebView*, insertando el contenido del mismo modo.

No se ha comentado antes, pero es buena práctica incluir en el nombre del fichero su propósito, e.g., nombrar un controlador en iOS como *MainController*, nombrar un controlador en Android como *MainActivity* o una vista como *MainLayout*. Esto nos permite saber en todo momento cuál es la finalidad de cada fichero.

5.2. Verificación y validación

Para verificar y validar el producto, se han hecho pruebas de usuario durante el desarrollo del proyecto y al finalizar. Básicamente, el equipo de desarrollo ha ido probando la aplicación en emuladores de Android e iOS con diferentes tamaños de pantalla, desde pantallas de 4 pulgadas hasta pantallas de 10 pulgadas, como las de las *tablets* y distintas versiones de sistemas operativos tanto en Android como en iOS. La versión mínima del sistema operativo Android es la 5.0, tratándose de la API 21, mientras que para iOS, la versión mínima es la 10. Además de las pruebas hechas por el equipo de desarrollo, también ha probado la aplicación el cliente, tanto en los emuladores como en su propio terminal móvil, en las reuniones que se han hecho durante la etapa de construcción.

Capítulo 6

Conclusiones

Durante la estancia en prácticas he aprendido muchas cosas y he podido poner en práctica muchos de los conocimientos y competencias que he adquirido durante los cuatro años de universidad.

En primer lugar, he aprendido cómo funciona una empresa dedicada a ofrecer soluciones y servicios informáticos. Al tratarse de una empresa pequeña, se tarda poco tiempo en conocer a todas las personas que la forman y se crea un buen ambiente de trabajo. Siendo ésta mi primera experiencia en una empresa, todos los que estaban presentes ofrecían su ayuda y soporte, ayudándome así a integrarme rápidamente y sentirme cómodo.

Antes de empezar la estancia no sabía programar en el lenguaje C# ni era capaz de desarrollar una aplicación móvil. La empresa puso a mi disposición recursos para que aprendiera a programar usando el lenguaje C# y cuáles son los fundamentos de una aplicación móvil. Después de dos semanas mirando videotutoriales y documentación de Xamarin y practicando a medida que iba avanzado con la formación, ya era capaz de entender la tecnología de Xamarin y podía desarrollar una aplicación móvil sencilla. Desarrollar la aplicación móvil que ha pedido la escuela de música no ha sido fácil. Lo he conseguido gracias al esfuerzo dedicado, documentándome en todo momento acerca de cómo se hacen las cosas bien y siguiendo buenas prácticas de programación. He aprendido mucho acerca de Xamarin, C#, iOS y Android. Me he basado principalmente en la documentación oficial de Xamarin, Google y Apple para poder desarrollar una aplicación multiplataforma consistente con cada sistema operativo móvil.

Entre las cosas que he aprendido en la carrera y he podido poner en práctica dentro de la empresa se encuentran las competencias del itinerario de Ingeniería Informática del IIS01 al IIS06 [6]. Se trata de identificar problemas, analizarlos y dar la mejor solución, valorar las necesidades del cliente y especificar requisitos *software* para satisfacer dichas necesidades, identificar, evaluar y gestionar riesgos potenciales y desarrollar, mantener y evaluar sistemas *software*.

Esta experiencia me ha ayudado a mejorar tanto en el ámbito profesional como en los ámbitos personal y formativo. He mejorado en el ámbito profesional porque he tenido contacto con una empresa del sector informático, he visto de cerca cómo se trabaja y las expectativas que hay que cumplir. En el ámbito personal he mejorado gracias a las responsabilidades que he tenido durante mi estancia en prácticas, ayudándome a ver las cosas de una forma distinta a cómo lo hacía hasta ahora. También he adquirido nuevos conocimientos y esto ha mejorado mi formación.

Finalmente, he de mencionar que al finalizar mi estancia en la empresa, he mantenido el contacto con el supervisor y se me ha ofrecido la oportunidad de seguir trabajando con él, pero no en la misma empresa. Actualmente me encuentro trabajando con el supervisor de prácticas en otra empresa, en un proyecto grande, desde principios del mes de julio.

Bibliografía

[1] Model-view-controller. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Consulta: 21 de julio de 2018].

[2] Xamarin. https://en.wikipedia.org/wiki/Xamarin. [Consulta: 27 de julio de 2018].

[3] Apple Inc. Human Interface Guidelines. https://developer.apple.com/design/human-interface-guidelines/ios/overview/iphone-x/. [Consulta: 22 de julio de 2018].

[4] Google. Material Design. https://material.io/design/. [Consulta: 22 de julio de 2018].

[5] NinjaMock.com. Ninja Mock. https://ninjamock.com/. [Consulta: 23 de julio de 2018].

[6] Oficiales de Grado y Máster de la Universitat Jaume I. Sobre los itinerarios del Grado en Ingeniería Informática de la Universitat Jaume I. https://ujiapps.uji.es/ade/rest/storage/UC0I0EIJIBUXX824FOVP3RAT5HPAOYT0. [Consulta: 30 de julio de 2018].