FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Aplicación móvil para la gestión interna de la asociación Microsoft UPC

Autor: Carlos F Álamo García
Directora: Cristina Gómez Seoane
Grado en ingeniería informática
Curso 2016/17 Q2

RESUMEN

En este proyecto de Trabajo Final de Grado, se explicará el proceso de desarrollo de una aplicación desarrollada en Android con un servicio de *backend* en Azure.

La junta de la asociación Microsoft UPC necesita un software para poder gestionar los socios, poder realizar la contabilidad y tener un calendario de eventos, con la especial característica de que debería poder utilizarse en un dispositivo móvil.

El programa se desarrollará en 4 meses utilizando la metodología ágil *scrum*, dividido en 9 iteraciones. En el transcurso de este documento se explicarán los métodos de desarrollo desde toma de requisitos, planificación, presupuesto o tecnologías utilizadas para poder llevar a cabo el desarrollo de este software, además se desarrollará una reflexión para ver que la aplicación es sostenible.

RESUM

En aquest projecte de Treball Final de Grau, s'explicarà el procés de desenvolupament d'una aplicació desenvolupada en Android amb un servei de backend en Azure.

La junta de l'associació Microsoft UPC necessita un *software* per poder gestionar els socis, poder realitzar la comptabilitat i tenir un calendari d'esdeveniments, amb l'especial característica que hauria de poder utilitzar-se en un dispositiu mòbil.

El programa es desenvoluparà en 4 mesos utilitzant la metodologia àgil *scrum*, dividit en 9 iteracions. En el transcurs d'aquest document s'explicaran els mètodes de desenvolupament, desde la presa de requisits, planificació, pressupost o tecnologies utilitzades per poder dur a terme el desenvolupament d'aquest programari, a més es desenvoluparà una reflexió per veure que l'aplicació és sostenible.

ABSTRACT

In this project Final Grade Work, will explain the process of developing an application developed in Android with a backend service in Azure.

The Microsoft UPC association board needs software to be able to manage partners, to be able to perform accounting and have a calendar of events, with the special feature that it should be able to be used in a mobile device.

The program will be developed in 4 months using the agile scrum methodology, divided into 9 iterations. During this document will explain the methods of development from requirements, planning, budgeting or technologies used to carry out the development of this software, in addition will be developed a reflection to see that the application is sustainable.

AGRADECIMIENTOS

A mi directora Cristina Gómez Seoane, que me ha guiado durante todo el desarrollo y documentación del proyecto.

A la Junta de la asociación por haberme dado la oportunidad de realizar este proyecto y haber estado muy receptivos durante todo el desarrollo de la aplicación.

A mi madre, por su positividad, cariño y trabajo que han hecho que pueda haber realizado todas estas etapas de la vida.

A toda mi familia y mis amigos por el apoyo y ánimos recibidos durante todos estos años, sobre todo en los momentos más difíciles.

Índice

Índice de contenido

Índice de tablas

Índice de ilustraciones

1 Introducción

1.1 Contextualización

Las asociaciones sin ánimo de lucro, “son entidades constituidas voluntariamente por tres o más personas para cumplir una finalidad de interés general o particular, mediante la puesta en común de recursos personales o patrimoniales con carácter temporal o indefinido”, traducción de la definición de asociación de la página web gencat de la Generalitat de Catalunya [1]. Estas asociaciones están formadas por socios. Un conjunto de estos socios forma parte de la Junta. La Junta son los socios encargados de organizar y mantener la organización y el resto de socios.

La Junta como mínimo tiene que estar formada por tres socios, tal como indica el BOE Numero 73/2002 articulo 5 [2]. En Catalunya, tal como se puede comprobar en el formulario de inscripción del acta fundacional [3] y el modelo de estatutos [4], artículos 22, 23 y 24 de la página web gencat, es obligatorio que estos tres socios sean el presidente, encargado de organizar y delegar, el secretario, encargado de las actas y temas fiscales, y el tesorero, encargado de los temas económicos de la asociación. La Junta además puede contener otros encargados, como un vicepresidente, sustituto del presidente, y grupos de trabajo, conjunto de personas dentro de la Junta, encargados de temas más concretos, aunque estos últimos son comunes en asociaciones con muchos socios, y para llevar la organización es necesario una estructura jerárquica mayor.

Las tareas básicas de toda junta son las siguientes:

- Llevar un libro de cuentas de los gastos de la asociación
- Estar al día de los socios que tiene la asociación
- Convocar asambleas y reuniones y realizar el acta de estas.

La mayoría de asociaciones, para llevar la gestión, utilizan software ofimático, como tablas de Excel para la contabilidad y los socios, y utilizan el email, para informar de nuevas reuniones y asambleas.

La asociación Microsoft UPC, actualmente está en tramitación, en espera de que la UPC apruebe su creación y su nombre, por lo que la asociación actualmente tiene un nombre y una junta provisional.

En una reunión de los miembros de la Junta, se debatió cual iba a ser el método o que herramienta se iba a utilizar para poder gestionar la asociación en un futuro, y en una pequeña búsqueda, no encontraron ningún método que satisficiera todas las necesidades que tenían.

La Junta está buscando un software para ellos, que integre funcionalidades para poder realizar las tareas básicas de cualquier asociación, pero que, además, se pueda utilizar en un móvil o una tableta, ya que quieren poder utilizarlo en situaciones en las que no se dispone de ordenador, como en eventos o *stands*.

Además de las tareas descritas anteriormente, la asociación Microsoft UPC, realizará eventos o talleres, donde podrán asistir socios. La asociación necesitará saber la siguiente información de los eventos:

- Cuando se realizarán, poder informar a los socios.
- Saber que socios asistirán.
- Saber que socios han abonado el pago del evento en caso necesario.

Otra gestión de la asociación será saber que socios han abonado la cuota anual.

1.2 Formulación del problema

La Junta de la asociación tiene el problema de que no ha encontrado ningún método para realizar la gestión interna de su asociación que cubra todas sus tareas.

La asociación necesita una aplicación móvil para poder gestionar las tareas descritas en la contextualización, y tener toda la información registrada al momento, con el fin evitar que, en el futuro, a alguien se le olvide poder registrar algo o haya perdidas de información.

Otro problema que tiene la asociación es que, si se utiliza una aplicación, tiene que poder guardar la información de forma online para que todos puedan acceder a la información desde cualquier dispositivo.

1.3 Alcance

En este proyecto se desarrollará una aplicación móvil para la gestión de la asociación, y un servidor para almacenar toda la información para que la información este sincronizada en todos los terminales con la aplicación instalada.

La aplicación se llamará AgentM, y permitirá llevar un registro de:

- Socios con la información de sus datos personales y de los eventos a los que ha asistido.
- Balance de cuentas de la asociación.
- Lista de los socios que han pagado y de los que no han pagado la cuota anual.
- Calendario para asignar eventos.
- Registrar los socios que participarán y ver que socios han abonado el evento.

AgentM también permitirá exportar los datos de los socios y del balance de cuentas a formato CSV, para poder abrirlo con cualquier programa ofimático, además de permitir enviar emails a una lista de distribución de los socios.

El servidor se desarrollará con Azure, y la principal función será almacenar toda la información en una base de datos.

El programa estará diseñado para trabajar con los dispositivos móviles rápidamente en cualquier lugar. Pero no dispondrá de funcionalidades complejas, como realizar el balance anual de la empresa, para ello, la aplicación permitirá exportar los datos a formato CVS, y así, posteriormente poder tratar los datos.

1.4 Estado del Arte

La Junta de la asociación Microsoft UPC necesitaba una solución para la organización.

Una vez descartada cualquier herramienta de ofimática y cualquier método físico, como libreta de cuentas o libro de socios, se ha buscado software en la red para la gestión de ellos. Los dos programas que más se acercaban a lo que la Junta quería, son los siguientes:

- Software para asociaciones XL de Idesoft [6], este programa de pago y disponible solo para Windows, está diseñado para la gestión de grandes asociaciones, enfocada en la gestión de socios, pagos y contabilidad.
- Programa de gestión de asociaciones y generación de recibos de MDGSoft [7], esta otra aplicación de pago y solo para Windows, dispone de gestión de socios, pero principalmente está enfocada para la contabilidad. Un inconveniente de este programa es que su interfaz es rudimentaria, tal y como se puede comprobar en la siguiente ilustración.

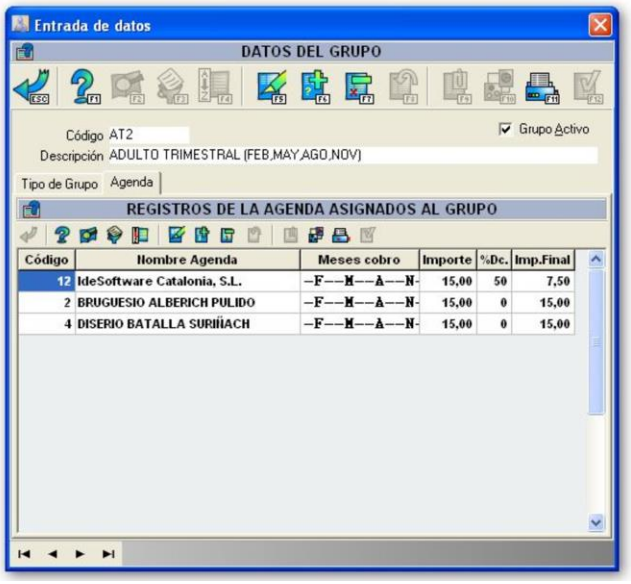


Ilustración 1. Programa MDGSoft

El problema de ambos programas es que sendos softwares solo funcionan en versiones de Windows, lo cual es un problema, porque se quería un programa para plataforma móvil. Además, estos programas contaban con características innecesarias, como poder trabajar con bancos, y le faltaban funcionalidades, como la de poder tener un calendario de eventos. A pesar de todo esto, estos programas han aportado nuevas ideas, como la posibilidad de exportar los datos.

En la siguiente tabla se comparan las funcionalidades de ambas aplicaciones con AgentM:

Funcionalidad	XL Idesoft	MDGSoft	AgentM
Plataforma móvil	No	No	Sí
Gestión de socios	Sí	Sí	Sí
Gestionar recibos	Sí	Sí	Sí
Generar recibos	Sí	Sí	No
Cobro a través de banco	Sí	Sí	No
Contabilidad	Sí	Sí	Sí
Importación de datos	Sí	No	No
Exportación de datos	Sí	No	Sí
Copias de seguridad	No	No	Sí
Sincronizado en diferentes dispositivos	No	No	Sí
Calendario	No	No	Sí
Envío de emails	Sí	No	Sí

Tabla 1.Comparativa de funcionalidades

Una vez visto que se necesitaba una aplicación móvil, se realizó una búsqueda de tienda Google Play en busca de aplicaciones. No se encontró ninguna aplicación para la gestión de asociaciones, pero se encontraron diversas aplicaciones para tareas concretas, sobre todo para el balance de cuentas.

De estas aplicaciones de Google Play, podemos destacar Mis Finanzas[8] y Meetup [9]. Mis Finanzas, es una aplicación de contabilidad, con herramientas muy potentes, en la cual podemos tener diferentes carteras, podemos organizar por categorías, y podemos

crear multitud de gráficos y estadísticas. Meetup, es una aplicación en la cual se crean eventos y los usuarios se puede registrar para asistir. Como ya se ha descrito, estas aplicaciones tienen herramientas que también tendrá AgentM, sin embargo no contienen todas las funcionalidades necesarias.

Una vez valorada la situación, se decidió diseñar y realizar un software propio para la gestión de la asociación.

1.5 Objetivo

En este proyecto podemos diferenciar un objetivo general y un conjunto de objetivos específicos.

1.5.1 Objetivo general

El objetivo de este proyecto es realizar una aplicación en Android para que la Junta de la asociación Microsoft UPC pueda realizar la gestión de los socios, la contabilidad y los eventos.

1.5.2 Objetivo especifico

La aplicación necesitara los siguientes requisitos para poder cumplir con los objetivos:

- La solución debe poder registrar nuevos socios, con toda la información necesaria de cada uno.
- Se tiene que poder realizar el balance de cuentas y añadir nuevos movimientos, con información específica de cada transacción.
- Ha de poderse llevar el control de los usuarios que han pagado y los que les falta pagar la cuota anual.
- Tiene que ser capaz de organizar eventos, saber que socios asistirán y cuales han realizado el pago del evento.
- Debe de tener un sistema de envió de información telemática
- Los datos tienen que estar sincronizados entre todos los dispositivos.

- Se ha de poder exportar datos de socios y del balance de cuentas.

1.6 Competencias técnicas

A lo largo del desarrollo de este proyecto se han utilizado un conjunto de competencias técnicas. En este apartado las explicaremos.

- **CES 1.1 Desarrollar mantener y evaluar sistemas y servicios software complejos y/o críticos.** En este proyecto se desarrolla y se evalúa, en profundidad, un sistema complejo, ya que se crea un software con almacenamiento online y con cuentas de seguridad para el acceso de los gestores de la asociación.

- **CES1.5 Especificar, diseñar, implementar y evaluar bases de datos.** Para poder realizar la aplicación, a partir de los requisitos, se ha tenido que especificar, diseñar e implementar una base de datos donde poder almacenar el contenido de la aplicación, por estos motivos esta competencia técnica se ha trabajado bastante.

- **CES1.7 Controlar la calidad y diseñar pruebas en la producción de software.** Junto a los requisitos, se especificarán unos criterios de calidad, los cuales se tendrán que cumplir o validar mediante *software* de *testing* y la supervisión de los encargados de la Junta de la asociación. Con este conjunto de procesos, se logra tratar en profundidad esta competencia.

- **CES2.1 Definir y gestionar los requisitos de un sistema software.** Para poder conocer que software necesita la asociación, se han definido todos los requisitos, tanto los funcionales como no funcionales, utilizando historias de usuario. Los requisitos se han gestionado en cada iteración, y si era necesario se han definido nuevos. Todo este trabajo realizado a supuesto trabajar esta competencia en profundidad.

- **CES2.2 Diseñar soluciones apropiadas en uno o más dominios de aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos.** AgentM la utilizará la Junta de la asociación para la gestión de socios y para llevar las cuentas económicas, por lo tanto, se

desarrollarán funcionalidades para poder realizar todas estas gestiones. Así vemos que esta competencia se desarrollara un poco.

2 Planificación

2.1 Recursos

Los recursos se pueden diferenciar en tres tipos dependiendo si son humanos, hardware o software.

2.1.1 Recursos humanos

El proyecto tendrá un desarrollador responsable, encargado de realizar el trabajo, este se guiará por los consejos de su director y los requisitos de la Junta de la asociación Microsoft UPC.

2.1.2 Recursos hardware

Para realizar el proyecto se dispondrá de los siguientes componentes:

- Portátil con sistema operativo Windows 10, con procesador i7-6700K, tarjeta gráfica Nvidea 8GB y memoria RAM de 24GB.
- Smartphone Nexus 5X con sistema operativo Android.
- Servidores de Azure, con un presupuesto máximo de 20 euros mensuales.

2.1.3 Recursos software

Los recursos software utilizados para realizar el proyecto serán los siguientes:

- Android Studio, para realizar la aplicación que se ejecutará en los terminales Android.
- SourceTree, programa gestor de repositorios GIT.
- Visual Studio 2015, para la configuración de los servidores de Azure.

2.2 Calculo del tiempo

Este proyecto se realizará en 3 meses con una jornada de trabajo de 40 horas semanales, por lo cual, el proyecto se desarrollará en un total de 503 horas tal y como podemos comprobar en la tabla del listado de tareas.

El TFG se compone de 18 créditos, y cada crédito tiene una carga de trabajo media de 30 horas, esto hace un total de 540 horas. El tiempo del TFG es un 9% superior al del estimado de este proyecto, tal y como se explica en el siguiente punto, es muy probable que haya una desviación del tiempo, sabiendo esto, para el cálculo de presupuesto también añadiremos un coste de contingencia.

Para realizar este cálculo, se ha tenido primero que realizar la toma de requisitos, y así poder enumerar las tareas a realizar.

Para contabilizar el tiempo de cada tarea se le ha añadido un tiempo extra necesario para redactar la memoria del trabajo.

En la siguiente tabla mostramos las tareas y cuántas horas se dedicará a cada una.

Tarea	Descripción	Horas(h)
TAREA #00	**Toma de requisitos de la aplicación.**	**20**
TAREA #01	**Definición de las historias de usuario.**	**40**
TAREA #02	**Formarse en desarrollo con Azure.**	**50**
TAREA #03	**Configuración del entorno de trabajo y de Azure.**	**8**
TAREA #04	**Definición y creación de la base de datos.**	**4**
TAREA #05	**Conexión de la base de datos con la aplicación.**	**16**
TAREA #06	**Crear un socio.**	**8**
TAREA #07	**Listar los socios.**	**32**
TAREA #08	**Modificar un socio.**	**6**
TAREA #09	**Eliminar un socio.**	**2**
TAREA #10	**Ver un socio.**	**8**
TAREA #11	**Buscador de socios por nombre o DNI para el listado de socios.**	**16**
TAREA #12	**Exportador de socios a formato EXCEL.**	**10**
TAREA #13	**Crear la vista de las cuotas anuales (listado de socios conforme si han pagado o no).**	**16**
TAREA #14	**Añadir pago de la cuota de un usuario.**	**16**
TAREA #15	**Modificar el año de visualización de pagos.**	**2**
TAREA #16	**Filtrar solo por impagados.**	**2**
TAREA #17	**Buscador de socios por nombre o DNI para el listado de cuotas.**	**2**
TAREA #18	**Crear un nuevo registro de cuentas.**	**8**
TAREA #19	**Listar el libro de cuentas con los registros.**	**6**
TAREA #20	**Mostrar el saldo de total.**	**2**
TAREA #21	**Ver un registro.**	**8**
TAREA #22	**Modificar un registro.**	**4**
TAREA #23	**Eliminar un registro.**	**2**
TAREA #24	**Exportar el libro de cuentas a formato EXCEL.**	**10**
TAREA #25	**Crear un nuevo evento.**	**8**
TAREA #26	**Listar los eventos.**	**16**
TAREA #27	**Ver un evento.**	**8**
TAREA #28	**Modificar un evento.**	**6**
TAREA #29	**Eliminar un evento.**	**2**
TAREA #30	**Añadir socios al evento.**	**16**
TAREA #31	**Confirmar pago de los usuarios al evento.**	**8**
TAREA #32	**Ver los eventos en un calendario.**	**16**
TAREA #33	**Crear menú de redacción de asunto y contenido email en la aplicación.**	**3**
TAREA #34	**Enviar el email**	**2**
TAREA #35	**Testear y confirmar que la aplicación está finalizada.**	**40**
TAREA #36	**Finalizar y revisar la memoria del trabajo.**	**40**
TAREA #37	**Preparar la presentación.**	**40**
TOTAL		503

Tabla 2.Listado de tareas

2.3 Valoración de alternativas y plan de actuación

Es muy probable que este proyecto sufra desviaciones de la planificación, como la mayoría de proyectos, por lo tanto, cada vez que haya una desviación se comparara con el diagrama para comprobar cuál ha sido la gravedad.

Para poder solucionar las desviaciones y poder confirmar que el proyecto se finaliza en el tiempo establecido, se seguirán un conjunto de medidas.

Una de las primeras medidas para la asignación de los tiempos de las tareas, se ha realizado una sobreestimación del tiempo normal, intentando no apurar el tiempo normal del desarrollo, ya que es muy probable que sobre todo, el diseño de la aplicación vaya variando en cada iteración, y que cada vez que se amplíen funcionalidades que afecten a otras funcionalidades ya realizadas, todas estas se tendrán que volver a validar.

La presentación se ha planificado acabarla antes de la entrega de la memoria, en caso de que se produzca una desviación muy grave, se podría posponer la realización, y la planificación finaliza dos semanas antes de la entrega de la memoria. En cualquier caso, si hubiese una desviación de más de una semana, se realizaría una reunión con la tutora para tomar medidas y solucionarlo.

2.4 Identificación de los costes

En este apartado veremos todos los recursos que utilizaremos y sus costes. Los recursos los identificaremos según sean humanos, materiales o de *software*.

2.4.1 Costes Directos

2.4.1.1 Recursos Humanos

Como recurso humano se dispondrá de una persona para realizar el proyecto, y esta asumirá diferentes roles para poder realizar las diferentes tareas.

Para asignar un salario a cada rol, se ha comprobado los salarios del estudio de remuneración Michael Page [10], y se ha seleccionado la remuneración media entre el máximo y el mínimo y se ha supuesto que la experiencia en desempeño de los roles era de 1 año y la edad de la persona es inferior de 30 años. Para obtener el coste en horas, se ha estimado que un trabajador realiza unas 1785 horas anuales.

Los roles que se necesitan para realizar el proyecto son:

- Un jefe de proyecto: es el responsable, designa el material a utilizar, y negocia el proyecto con los clientes.
- Un analista funcional: se encarga de redactar las especificaciones técnicas y del desarrollo de las aplicaciones.
- Un arquitecto de base de datos: se encarga de la creación, el desarrollo y la integración de la base de datos con el proyecto.
- Un responsable de calidad: es el encargado de realizar los procedimientos de test y establecer la calidad a los proyectos.

Cargo	Salario Anual (€)	Remuneración (€/hora)
Jefe de proyecto	39.500	22
Analista funcional	27.500	15
Arquitecto de base de datos	33.000	18
Responsable de calidad	45.000	25

Tabla 3.Remuneración de cargos

En la tabla siguiente podremos ver el coste asociado a cada una de las tareas descritas en el diagrama de Gantt.

TAREA	RESPONSABLE	DURACION HORAS	COSTE EUROS
TAREA #00	**Jefe de proyecto**	**20**	**440**
TAREA #01	**Analista funcional**	**40**	**600**
TAREA #02	**Analista funcional**	**50**	**750**
TAREA #03	**Analista funcional**	**8**	**120**
TAREA #04	**Arquitecto de base de datos**	**4**	**72**
TAREA #05	**Arquitecto de base de datos**	**16**	**288**
TAREA #06	**Analista funcional**	**8**	**120**
TAREA #07	**Analista funcional**	**32**	**480**
TAREA #08	**Analista funcional**	**6**	**90**
TAREA #09	**Analista funcional**	**2**	**30**
TAREA #10	**Analista funcional**	**8**	**120**
TAREA #11	**Analista funcional**	**16**	**240**
TAREA #12	**Analista funcional**	**10**	**150**
TAREA #13	**Analista funcional**	**16**	**240**
TAREA #14	**Analista funcional**	**16**	**240**
TAREA #15	**Analista funcional**	**2**	**30**
TAREA #16	**Analista funcional**	**2**	**30**
TAREA #17	**Analista funcional**	**2**	**30**
TAREA #18	**Analista funcional**	**8**	**120**
TAREA #19	**Analista funcional**	**6**	**90**
TAREA #20	**Analista funcional**	**2**	**30**
TAREA #21	**Analista funcional**	**8**	**120**
TAREA #22	**Analista funcional**	**4**	**60**
TAREA #23	**Analista funcional**	**2**	**30**
TAREA #24	**Analista funcional**	**10**	**150**
TAREA #25	**Analista funcional**	**8**	**120**
TAREA #26	**Analista funcional**	**16**	**240**
TAREA #27	**Analista funcional**	**8**	**120**
TAREA #28	**Analista funcional**	**6**	**90**
TAREA #29	**Analista funcional**	**2**	**30**
TAREA #30	**Analista funcional**	**16**	**240**
TAREA #31	**Analista funcional**	**8**	**120**
TAREA #32	**Analista funcional**	**16**	**240**
TAREA #33	**Analista funcional**	**3**	**45**
TAREA #34	**Analista funcional**	**2**	**30**
TAREA #35	**Responsable de calidad**	**40**	**1000**
TAREA #36	**Analista funcional**	**40**	**600**
TAREA #37	**Analista funcional**	**40**	**600**
TOTAL		503	8145

Tabla 4.Coste de tareas del Gantt.

2.4.1.2 *Hardware*

Para poder realizar el proyecto, se utilizarán una serie de recursos *hardware,* en concreto, un portátil y un *smartphone.* Para saber el coste de amortización, primero tendremos que calcular el coste de amortización por cada hora.

En la tabla del diagrama de Gantt, se puede ver que el desarrollo total del proyecto es de 503 horas. Se ha aproximado que el uso del portátil será de un 95% del total del tiempo, y el uso del *smartphone* un 20%. La amortización de los dispositivos, se alcanza cuando la vida útil llega a los 4 años, teniendo en cuenta que hay unos 220 días de trabajo y cada dio se trabaja unas 8 horas con ellos, da un resultado de un tiempo total de amortización de 7040 horas, hasta que el dispositivo deberá sustituirse.

Dispositivo	Precio (€)	Tiempo proyecto (h)	Amortización (€/h)	Coste proyecto (€)
Portátil i7	1550	478	0,2202	105,2222
Smartphone	350	101	0,04972	5,0018
TOTAL				110,22

Tabla 5.Costes hardware

2.4.1.3 *Software*

En este apartado se calcularán los costes de *software*, como queremos calcular el coste del proyecto se omitirán los programas gratuitos. El tiempo de amortización de los programas comprados es de 3 años y el uso es de 8 horas diarias durante 220 días anuales.

Programa	Precio (€)	Tiempo proyecto (h)	Amortización (€/h)	Coste proyecto (€)
Windows 10 Pro	279	503	0,05284	26,5785
TOTAL				26,58

Tabla 6.Costes software

2.4.1.4 Azure

Para desarrollar el proyecto se adquirirán un conjunto de servicios de Azure, para obtener los servidores de almacenamiento y el servidor de mensajería, este gasto será mensual mientras esté en funcionamiento la aplicación.

Para ello necesitaremos contratar los servicios de AppService, para obtener un servidor y un servicio *backend*, y una base de datos SQL Database.

Las siguientes imágenes muestran la información y el precio de los servicios, el precio de los servicios puede variar en cualquier momento según las condiciones de Microsoft, para más información ver la calculadora de Azure [11].

Ilustración 2.Azure AppServices

Ilustración 3.Azure SQL Database

2.4.2 Costes Indirectos

En este apartado se describirán los gastos indirectos asociados al proyecto.

Tendremos un gasto energético. Si disponemos de un portátil que consume 330W, y que se utilizará durante 503 horas lo cual nos da 166kW. También tenemos un smartphone que consume 5W, al utilizarlo durante 95 horas, tendremos un resultado de 475W.

Además, como vemos en la tabla, tendremos otros gastos.

Servicio	Precio	Cantidad	Total €
Electricidad	0,128€/kWh	166KW	21,2467
Internet	60€/mes	3 meses	180
Papel	5€/pack	1	5
TOTAL			206,25

Tabla 7.Costes Indirectos

2.4.3 Imprevistos

Como en cualquier proyecto, no es de extrañar que a medida que se avance en la realización aparezcan imprevistos. Para que los imprevistos no supongan un aumento significante en el coste total, ahora se calculará una suma que se le aplicará a costes como costes de imprevistos.

Hemos visto que las horas de realización del proyecto serán de unas 503 horas, pero es bastante probable que aparezcan situaciones que hagan ralentizar el proyecto. Tenemos un margen de 14 días para finalizar el proyecto como se puede comprobar en el Gantt, y la probabilidad de que se utilicen estos días es del 45%. Si el coste total de mano de obra es de 8145 euros en 503 horas, calculando, tenemos que la media es de 17.74€/día, en caso de utilizar 14 días adicionases a 17.74€/día con una probabilidad del 45%, obtenemos un resultado de un aumento en los recursos humanos de 111.76€ en concepto de imprevistos.

Durante la elaboración del trabajo, también se utilizarán recursos *hardware*, los cuales, a veces se estropean. La probabilidad de una avería de los componentes es de un 3%, si el coste total del *hardware* es de 1900 euros, con esta probabilidad nos da un resultado de 57 euros de aumento en los imprevistos por costes de *hardware.*

Imprevisto	Coste (€)	Probabilidad	Total €
14 días de trabajo	248,36	45%	111,76
Hardware	1900	3%	57
TOTAL			168,76

Tabla 8.Imprevistos

2.4.4 Contingencia

La mejor manera de evitar que se exceda el presupuesto acordado, es prevenir que esto ocurra, con una reserva de dinero para contingencias. Se le aplicara un porcentaje del 10% a la suma de los costes.

2.4.5 Presupuesto

En la tabla siguiente se resumen todos los gastos descritos anteriormente, para obtener un presupuesto de la creación del proyecto.

Concepto	Coste (€)
Recursos Humanos	8145
Hardware	100,22
Software	26,58
Indirectos	206,25
Imprevistos	168,76
Coste Parcial	8.646,81
Contingencia	10%
TOTAL	9511*

Tabla 9.Presupuesto

*Además, se tendrán que aportar 12.36€/mes por los servicios de Azure.

2.5 Desviación respecto a la previsión inicial

La previsión para finalizar el proyecto era el 9 de junio de 2017.

Durante el desarrollo del proyecto se tuvieron que posponer las tareas porque 4 días no pude realizar trabajo. Por este motivo, se fijaba la nueva fecha final el 15 de junio. Este hecho no afectaba al coste económico, puesto que no se realizó el proyecto durante esos días.

Finalizando el trabajo, se percibió un desfase entre la implementación realizada y la documentación finalizada, por este motivo se alargó el plazo final 5 días más, teniendo 40 horas adicionales no contempladas de trabajo y teniendo que posponer la preparación de la lectura después de la fecha límite de la memoria.

Eso ha hecho que el tiempo total de realización del proyecto haya sido 543 horas, un 8% superior a lo estimado. El presupuesto previsto es correcto, ya que al precio total se le sumo un 10% de contingencia, por lo cual no se ha excedido el presupuesto.

3 Metodología

En este apartado veremos cómo se especificará el trabajo a realizar, y que estilo de planificación se utilizará para el desarrollo del proyecto.

Para la realización de AgentM, se pretende utilizar una metodología ágil propia, utilizando algunas características de Scrum [12], por lo que el desarrollo se dividirá en iteraciones, aunque en pequeñas ocasiones aplicaremos técnicas de la metodología clásica en cascada [13]. En los siguientes párrafos, se detallarán las iteraciones y las metodologías se han aplicado. Las iteraciones del proyecto tendrán una duración de dos semanas.

La primera iteración se dedicará a la toma de requisitos funcionales y no funcionales. En la obtención de los requisitos funcionales, se utilizarán historias de usuario[1], creando una tarjeta [14] para cada requisito. Una vez finalizado el proceso, se obtendrá el backlog, que es el conjunto de historias de usuario pendientes de realizar.

Las tarjetas contarán con los siguientes elementos y con la siguiente disposición.

- ID: número identificador de la historia de usuario.
- Título: título de la historia de usuario.
- Descripción: breve descripción de la historia detallando cualquier característica importante.
- Estimación: putos de dificultad para desarrollar la historia de usuario, cuanto más alto más dificultad.
- Prioridad: Necesidad de la junta para que esta historia este desarrollada, cuanto más alto más prioritario.
- Pruebas de aceptación o criterio de aceptación: premisas que se han de cumplir para que la historia de usuario se pueda dar como validada.

[1] Como indica el experto en Scrum, Charles Bradley [15] en su página web [16], las historias de usuario no forman parte de Scrum y por lo tanto Scrum no las define.

ID	TÍTULO
DESCRIPCIÓN	
ESTIMACIÓN	PRIORIDAD
PRUEBAS DE ACEPTACIÓN O CRITERIO DE ACEPTACIÓN	

Tabla 10.Tarjeta de historia de usuario.

Para realizar la estimación de las historias de usuario, se enumerará con la secuencia de Fibonacci, secuencia que se utiliza en la técnica de estimación *Planning Poker* [17], aunque no se realizará esta técnica, ya que solo estaré yo como implicado en el desarrollo de la aplicación.

La prioridad será definida por los encargados del seguimiento del proyecto de la junta de la asociación y la valorarán entre 0 y 100, siendo 100 lo más prioritario.

En la siguiente iteración, se creará la base de datos, se configurará el servidor y se creará la conexión con la aplicación. En esta iteración, aplicaremos metodología clásica en cascada, ya que, a partir de los requisitos, diseñaremos e implementaremos el backend. Esto resulta viable porque el sistema no es complejo, y si en una iteración futura hay que modificarlo, resultará una tarea sencilla.

Las siguientes iteraciones se dedicarán al desarrollo de la aplicación móvil. Al principio de cada iteración se realizará una reunión con los encargados de la Junta, esta reunión se dividirá en dos fases:

- **Revisión de la iteración**: En este proceso, se comprobará que todas las historias de usuario realizadas estén validadas y los responsables de la Junta darán su aprobación. En caso de que haya algún error en alguna historia o tenga que modificarse, esta no se dará por finalizada y se añadirá al *backlog*.
- **Planificación de la iteración**: En esta parte de la reunión, se decidirá que historias del *backlog* se desarrollarán para la próxima reunión.

En las reuniones, puede que aparezcan nuevas historias de usuario, si es así, se especificarán y se añadirán al *backlog.* Trabajando de esta manera, conseguimos tener un pequeño prototipo de la aplicación en cada iteración para que la Junta pueda ver su funcionamiento, así poder identificar errores lo antes posible.

En la última iteración, se finalizará la documentación que falte por realizar, se revisará, y se preparará la presentación.

Una vez finalizada esta última iteración, se dará por acabado el proyecto, entregando a la Junta el código de la aplicación, las claves del servidor, y la documentación para que ellos puedan realizar el mantenimiento.

4 Definición de requisitos

En este apartado se muestran todos los requisitos tomados durante todo el proyecto, tomados junto a los encargados de la Junta. Estos requisitos los podemos diferenciar entre funcionales y no funcionales.

4.1 Requisitos funcionales

Para facilitar la comprensión y el desarrollo de AgentM, las historias de usuario se han agrupado en bloques según a la funcionalidad que pertenecen, ya que en la aplicación se podrá diferenciar si se está gestionando los socios, cuotas, balance, eventos, o envío de email.

4.1.1 Gestión de socios

Este conjunto de historias explica las funcionalidades que tendrá AgentM para gestionar la información de los socios.

1	Ver el listado de socios
Como gestor quiero poder ver el listado de socios, mostrando el apellido, el nombre y el NIF.	
5	100
▪ Al ver el listado de socios tiene que aparecer el apellido, el nombre y el NIF de cada uno de los socios que pertenecen a la asociación.	

2	Crear socio
Como gestor quiero poder crear un socio, indicando su nombre, apellidos, NIF, teléfono, email y si pertenece a la UPC.	
13	100
▪ No introducir un nombre y comprobar que sale un error. ▪ No introducir un apellido y comprobar que sale un error. ▪ No introducir un NIF y comprobar que sale un error. ▪ Introducir un NIF con formato incorrecto y comprobar que sale un error. ▪ No introducir un teléfono y comprobar que sale un error. ▪ Introducir un teléfono con formato incorrecto, nueve cifras, y comprobar que sale un error. ▪ No introducir un email y que salga un error. ▪ Introducir un email con formato incorrecto y que aparezca un error.	

3	Ver un socio
Como gestor quiero poder ver los datos de un socio.	
3	100
▪ Tiene que aparecer el nombre, apellidos, NIF, teléfono, email y si pertenece a la UPC, del socio seleccionado, además de ver si le faltan cuotas anuales por pagar y los cursos que ha pagado.	

4	Eliminar un socio
Como gestor quiero poder eliminar un socio.	
2	60
▪ Al eliminar un socio, se elimina también de la cuota de los socios.	

5	Modificar un socio
Como gestor quiero poder crear modificar la información de un socio; nombre, apellidos, NIF, teléfono, email y si pertenece a la UPC.	
3	100
▪ No introducir un nombre y comprobar que sale un error. ▪ No introducir un apellido y comprobar que sale un error. ▪ No introducir un NIF y comprobar que sale un error. ▪ Introducir un NIF con formato incorrecto y comprobar que sale un error. ▪ No introducir un teléfono y comprobar que sale un error. ▪ Introducir un teléfono con formato incorrecto y comprobar que sale un error. ▪ No introducir un email y que salga un error. ▪ Introducir un email con formato incorrecto y que aparezca un error.	

6	Buscar socio por nombre, apellido o NIF.
Como gestor quiero poder buscar un socio por nombre, apellido o NIF.	
8	80
▪ El nombre o el apellido o el NIF de los socios mostrados, tiene que contener la palabra de búsqueda.	

7	Ordenar listado de los socios por Nombre, Apellido o NIF.
Como gestor quiero poder ordenar la lista de socios por nombre, apellido o NIF.	
3	50
▪ El listado estará ordenado por defecto por el apellido.	

8	Exportar socios a Excel
Como gestor quiero poder exportar la información de los socios a formato Excel.	
8	60
▪ El Excel tiene que contener el nombre, apellidos, NIF, teléfono, email y si pertenece a la UPC de los socios, de los socios que se muestran en el listado.	

4.1.2 Gestión de las cuotas

En este apartado veremos las historias de usuario que agrupan las características sobre la gestión de las cuotas anuales de los socios.

9	Ver el listado de las cuotas.
Como gestor quiero poder ver el listado del pago de las cuotas de los socios.	
13	90
▪ El listado mostrará de cada socio, el apellido, el nombre, el NIF y si la cuota del año se ha pagado.	

10	Confirmar pago anualidad.
Como gestor quiero poder confirmar el pago de la anualidad de un socio.	
5	90
▪ Para completar la acción se requerirá una aprobación adicional. ▪ Al completar la acción la cuota del socio se verá pagada. ▪ En caso de cancelar la acción, no se marcará como pagado.	

11	Deshacer pago anualidad.
Como gestor quiero poder deshacer una confirmación de pago de la anualidad de un socio.	
2	90
▪ Para completar la acción se requerirá una aprobación adicional. ▪ Al completar la acción la cuota del socio se verá no pagada. ▪ En caso de cancelar la acción, se dejará marcado como pagado.	

12	Filtrar cuotas por año.
Como gestor quiero poder modificar el año de visualización de la cuota de los socios.	
2	90
▪ Se mostrarán las cuotas anuales de los socios del año seleccionado. ▪ Se mostrarán solo los socios pertenecientes al año seleccionado.	

13	Filtrar cuotas por pagados e impagados.
Como gestor quiero poder filtrar las cuotas de los socios por pagados e impagados.	
2	60
▪ Se mostrarán las cuotas anuales de los socios que tienen pagada o no pagada la anualidad de ese año.	

14	Buscar cuota por nombre, apellido o NIF.
Como gestor quiero poder buscar un socio por nombre, apellido o NIF.	
8	60
▪ El nombre o el apellido o el NIF de los socios mostrados, tiene que contener la palabra de búsqueda.	

15	Ordenar listado de cuotas por Nombre, Apellido o NIF.
Como gestor quiero poder ordenar la lista de cuotas por nombre, apellido o NIF de los socios.	
3	50
▪ El listado estará ordenado por defecto por el apellido.	

16	Exportar cuotas a Excel
Como gestor quiero poder exportar la información de las cuotas a formato Excel.	
8	60
▪ El Excel tiene que contener el nombre, apellidos, NIF y si ha pagado la cuota de las cuotas que se muestren en el listado	

4.1.3 Gestión del balance

A continuación, se detallan las historias de usuario asociadas a la gestión del balance y registro de la asociación.

17	Crear un registro.
Como gestor quiero poder crear un registro indicando el título, descripción, precio, fecha y fotografía	
8	90
▪ No introducir un título y comprobar que sale un error. ▪ No introducir un precio y comprobar que sale un error. ▪ Introducir un precio con formato incorrecto, numero positivo o negativo, y comprobar que sale un error. ▪ No introducir una fecha y comprobar que sale un error. ▪ Introducir una fecha con formato incorrecto, dd/mm/aaaa, y comprobar que sale un error.	

18	Ver el balance.
Como gestor quiero poder ver el listado con los registros.	
8	90
▪ Cada registro debe mostrar el título, la fecha y el precio.	

19	Ver el total del balance.
Como gestor quiero poder ver el importe total de los registros.	
2	60
▪ El valor total del balance tiene que ser la suma del precio de los registros que se muestran.	

20	Ver un registro.
Como gestor quiero poder ver un registro.	
5	90
▪ Se tiene que poder ver el título, la descripción, el precio, la fecha y la fotografía si tiene.	

21	Modificar un registro.
Como gestor quiero poder modificar la información de un registro; el título, descripción, precio, fecha y fotografía	
8	90
▪ No introducir un título y comprobar que sale un error. ▪ No introducir un precio y comprobar que sale un error. ▪ Introducir un precio con formato incorrecto, numero positivo o negativo, y comprobar que sale un error. ▪ No introducir una fecha y comprobar que sale un error. Introducir una fecha con formato incorrecto y comprobar que sale un error.	

22	Eliminar un registro.
Como gestor quiero poder eliminar un registro.	
2	90
▪ Para completar la acción se requerirá una aprobación adicional.	

23	Filtrar el listado del balance.
Como gestor quiero poder filtrar el listado del balance entre dos fechas.	
5	60
▪ Se muestran todos los registros entre las fechas seleccionadas incluyendo los días seleccionados.	

24	Exportar registros a Excel.
Como gestor quiero poder exportar la información de los registros a formato Excel.	
5	50
▪ El Excel tiene que contener el título, descripción, precio y fecha de los registros que se muestran en el listado.	

4.1.4 Gestión de eventos

En este otro apartado se especifican las historias de usuario para el desarrollo de la gestión de eventos.

25	Ver el listado de los eventos.
Como gestor quiero poder ver el listado de los eventos futuros.	
13	90
▪ En el listado se mostrará el día, hora, mes, año y título de cada evento.	

26	Ver el calendario de los eventos.
Como gestor quiero poder ver un calendario donde se marquen los eventos y ver los eventos de un día.	
18	90
▪ En el listado se mostrará el día, hora, mes, año y título de cada evento.	

27	Crear un evento.
Como gestor quiero poder crear un evento indicando el título, descripción, lugar, ubicación, precio, fecha y hora.	
5	90
▪ No introducir un título y comprobar que sale un error. ▪ No introducir una ubicación y comprobar que sale un error. ▪ No introducir un precio y comprobar que sale un error. ▪ Introducir un precio con formato incorrecto y comprobar que sale un error. ▪ No introducir una fecha y comprobar que sale un error. ▪ Introducir una fecha con formato incorrecto y comprobar que sale un error. ▪ No introducir una hora y comprobar que sale un error. ▪ Introducir una hora con formato incorrecto y comprobar que sale un error. Introducir una fecha y hora anterior a la actual y comprobar que sale un error.	

28	Ver un evento.
Como gestor quiero poder ver la información de un evento.	
13	90
▪ Al ver un evento se mostrará el título, descripción, lugar, ubicación, precio, fecha, hora y un lisado con el nombre, apellidos, NIF y confirmación de pago de los socios que participan.	

29	Modificar un evento.
Como gestor quiero poder modificar la información de un evento; el título, descripción, lugar, ubicación, precio, fecha y hora.	
5	90
▪ No introducir un título y comprobar que sale un error. ▪ No introducir una ubicación y comprobar que sale un error. ▪ No introducir un precio y comprobar que sale un error. ▪ Introducir un precio con formato incorrecto y comprobar que sale un error. ▪ No introducir una fecha y comprobar que sale un error. ▪ Introducir una fecha con formato incorrecto y comprobar que sale un error. ▪ No introducir una hora y comprobar que sale un error. ▪ Introducir una hora con formato incorrecto y comprobar que sale un error. ▪ Introducir una fecha y hora anterior a la actual y comprobar que sale un error.	

30	Eliminar un evento.
Como gestor quiero poder eliminar un evento.	
2	90
▪ Para completar la acción se requerirá una aprobación adicional.	

31	Confirmar pago de un evento.
Como gestor quiero poder confirmar el pago de un evento de un socio.	
5	70
▪ Para completar la acción se requerirá una aprobación adicional.	

32	Eliminar socios de un evento.
Como gestor quiero poder eliminar un socio de un evento.	
3	70
▪ Solo se pueden eliminar socios que no hayan pagado el evento.	

33	Añadir socios a un evento.
Como gestor quiero poder añadir socios a un evento.	
3	70
▪ No se podrán tener socios repetidos apuntados a un evento.	

4.1.5 Envío de emails

Por último, se describe la historia de usuario referente al envío de emails.

34	Crear un email.
Como gestor quiero poder crear un email indicando el asunto y el contenido, y enviarlo a todos los socios.	
5	60
▪ Se tienen que añadir todos los socios como destinatarios del email.	

Una vez creados las historias de usuario, hemos obtenido un backlog con un total de 34 historias de usuario que acumulan un total de 207 puntos.

4.2 Requisitos no funcionales

En este apartado, se explicarán los diferentes tipos de requisitos no funcionales que dispondrá nuestra aplicación.

4.2.1 Requisito de diseño

El diseño de la aplicación será atractivo y sencillo. La Junta de la asociación ha pedido tener una aplicación que tenga un buen diseño. Los encargados de la Junta validarán los aspectos de la interfaz de la aplicación para que sea acorde a su gusto, cada uno valorará el diseño con una puntuación de 0 a 10 y se dará como cumplido si la media de la puntuación es superior a 8.

4.2.2 Requisito de usabilidad

El sistema tiene que ser fácil e intuitivo para los socios que pertenecen a la junta de la asociación y vayan a utilizar la aplicación. Un nuevo usuario no necesitará más de 5 minutos para adaptarse al uso de la aplicación.

Para cumplir el requisito de usabilidad, se utilizarán los logos oficiales de Android para que la aplicación tenga una apariencia similar a otras apps y ayude a los usuarios a sentirse cómodos. La aplicación la testearán 5 personas y tendrán que realizar diferentes acciones; modificar un socio, crear un registro, exportar las cuotas, y añadir un socio a un evento, en menos de 30 segundos cada una.

4.2.3 Requisito de aprendizaje

Si un nuevo usuario tiene que empezar a utilizar la aplicación, no necesitará tener conocimientos adicionales al del uso de un smartphone o tablet. AgentM tiene que ser sencilla de utilizar, por lo tanto, los responsables de la junta, probarán las funcionalidades de la aplicación sin una previa explicación de cómo utilizarla, y así, validar este requisito.

4.2.4 Requisito de disponibilidad

El sistema ha de estar disponible las 24 horas del día durante los 365 días del año, ya que la asociación utilizará la aplicación en cualquier momento, por lo tanto, la aplicación estará disponible como mínimo el 99% del tiempo.

4.2.5 Requisito de latencia y velocidad

La respuesta del sistema ante una petición ha de ser menos de 3 segundos como mínimo en un 95% de los casos. Una respuesta rápida del sistema evitara la pérdida de tiempo al usuario y que este no piense que la aplicación no funciona correctamente.

4.2.6 Requisito de portabilidad

El sistema tiene que ser apto para dispositivos Android con SDK mínima de API 19. No todos los miembros de la junta tendrán la misma versión del API de Android, por lo que la aplicación tiene que funcionar en diferentes dispositivos. Para validar este requisito se probará la aplicación en un smartphone con API 25, y se emulará en smartphones con API 19, 21 y 23.

4.2.7 Requisito de seguridad

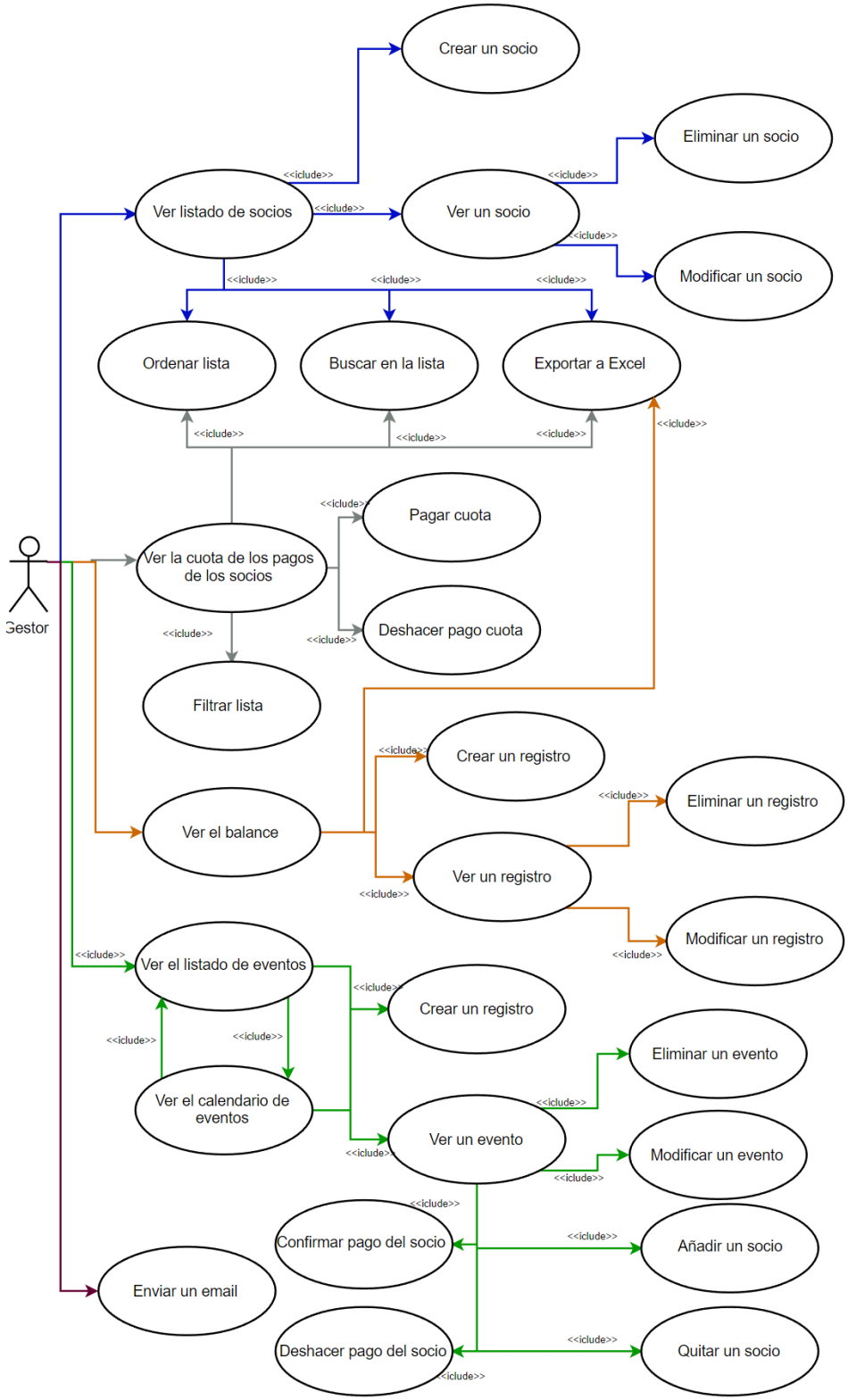
El acceso a AgentM ha de estar restringido a unos pocos usuarios. La información del sistema solo debe estar visible para la Junta de la asociación. Cada miembro de la Junta dispondrá de sus credenciales que AgentM le pedirá para acceder a las funcionalidades.

Se intentará conectarse a la aplicación sin introducir las credenciales y utilizando emails no registrados y validará que no es posible.

4.2 Casos de uso

4.2.1 Diagrama de casos de uso

El diagrama de casos de uso nos ayuda a hacernos una idea de la relación entre el actor, en este caso el gestor, y las diferentes acciones que pude hacer dentro de la aplicación.



4.2.2 Descripción de los casos de uso

Las diferentes gestiones de los socios, eventos, registros, etc. Son muy parecidas entre ellas, y como tenemos definidos los requisitos, en este apartado solo se explicarán los casos de uso de los socios, ya que los demás casos de uso serán muy similares.

CU #01	Ver el listado de socios
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere ver los socios.
Escenario principal	1. El gestor abre el menú lateral y selecciona socios. 2. El sistema muestra todos los socios de la asociación.

CU #02	Crear un socio
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere crear un nuevo socio
Escenario principal	1. El gestor selecciona el botón nuevo socio. 2. El sistema muestra el formulario para crear el nuevo socio. 3. El gestor rellena la información del socio. 4. El gestor clica en el botón guardar. 5. El sistema valida la información. 6. El sistema guarda la información y crea el nuevo socio. 7. El sistema cierra la pantalla del formulario.

Escenario alternativo.	5a. El sistema indica al gestor que no relleno el campo nombre. 5a1. Se vuelve al paso 3. 5b. El sistema indica al gestor que no relleno el campo apellidos. 5b1. Se vuelve al paso 3. 5c. El sistema indica al gestor que no relleno el campo NIF. 5c1. Se vuelve al paso 3. 5d. El sistema indica al gestor que no relleno el campo teléfono. 5d1. Se vuelve al paso 3. 5f. El sistema indica al gestor que no relleno el campo email. 5f1. Se vuelve al paso 3. 5g. El sistema indica al gestor que el NIF tiene formato incorrecto. 5g1. Se vuelve al paso 3. 5h. El sistema indica al gestor que el teléfono tiene formato incorrecto. 5h1. Se vuelve al paso 3. 5i. El sistema indica al gestor que el email tiene formato incorrecto. 5i1. Se vuelve al paso 3.

CU #03	Ver un socio
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere ver los datos de un socio.
Escenario principal	1. El gestor selecciona un socio de la lista. 2. El sistema muestra toda la información de los datos del socio, los cursos que ha pagado y las cuotas anuales no pagadas.

CU #04	Eliminar un socio
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere eliminar un socio de la asociación.
Escenario principal	1. El gestor clica en el botón eliminar. 2. El sistema muestra una ventana de confirmación. 3. El gestor clica en aceptar. 4. El sistema elimina al socio.
Escenario alternativo.	3a. El gestor clica en cancelar.

CU #05	Modificar un socio
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere modificar los datos de un socio.
Escenario principal	1. El gestor selecciona el botón editar socio. 2. El sistema muestra el formulario para editar el socio, con la información actual del socio. 3. El gestor modifica la información del socio. 4. El gestor clica en el botón guardar. 5. El sistema valida la información.

	6. El sistema guarda la información y modifica el socio. 7. El sistema cierra la pantalla del formulario.
Escenario alternativo.	5a. El sistema indica al gestor que no relleno el campo nombre. 5a1. Se vuelve al paso 3. 5b. El sistema indica al gestor que no relleno el campo apellidos. 5b1. Se vuelve al paso 3. 5c. El sistema indica al gestor que no relleno el campo NIF. 5c1. Se vuelve al paso 3. 5d. El sistema indica al gestor que no relleno el campo teléfono. 5d1. Se vuelve al paso 3. 5f. El sistema indica al gestor que no relleno el campo email. 5f1. Se vuelve al paso 3. 5g. El sistema indica al gestor que el NIF tiene formato incorrecto. 5g1. Se vuelve al paso 3. 5h. El sistema indica al gestor que el teléfono tiene formato incorrecto. 5h1. Se vuelve al paso 3. 5i. El sistema indica al gestor que el email tiene formato incorrecto. 5i1. Se vuelve al paso 3.

CU #06	Ordenar socios
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere ordenar el listado de los socios.
Escenario principal	1. El gestor abre el menú de opciones. 2. El gestor selecciona si quiere ordenar la lista por nombre, apellidos o NIF. 3. El sistema muestra el listado ordenado por la preferencia elegida en el paso 2.

CU #07	Buscar socio
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere buscar un socio.
Escenario principal	1. El gestor clica en el botón buscar. 2. El gestor escribe el nombre, apellidos o NIF del socio que busca. 3. El sistema muestra el listado de los resultados que coincidan con el texto introducido por el gestor en el paso 2.

CU #08	Exportar socios
Actor primario	Gestor
Precondición	El gestor ha iniciado sesión.
Disparador	El gestor quiere exportar los socios a EXCEL
Escenario principal	1. El gestor clica en el botón exportar a EXCEL. 2. El gestor decide donde quiere guardar el EXCEL. 3. El sistema guarda el EXCEL con la información de los socios.

5 Especificación

5.1 Modelo Conceptual

Este es el modelo conceptual de nuestra aplicación donde se definen las clases que tendrá y que relación habrá entre ellas.

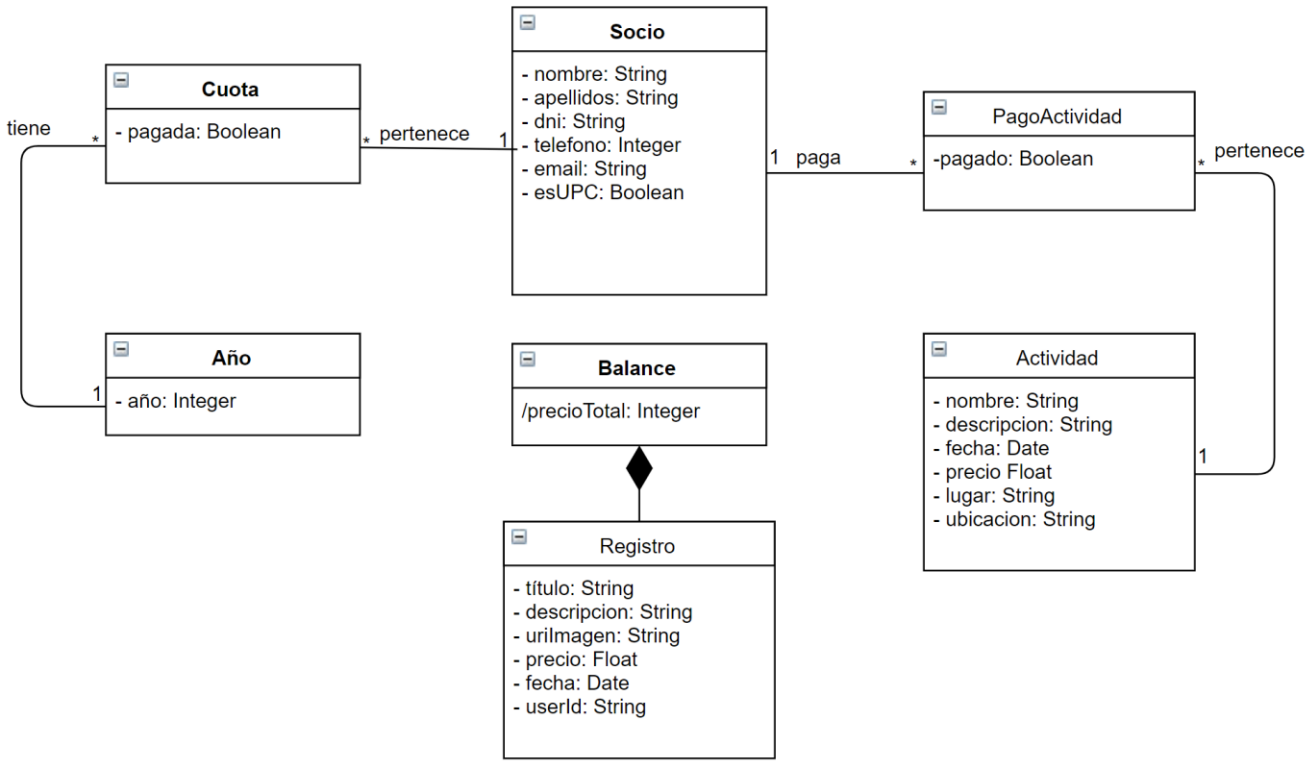


Ilustración 4.Modelo Conceptual

RT. 1 Un socio no puede tener dos cuotas el mismo año.

RT. 2 Un socio no puede participar más de una vez en la misma actividad.

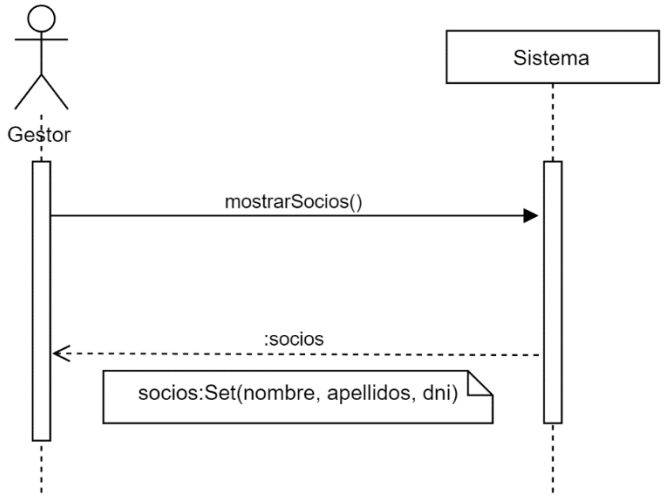
RT. 3 Si el precio de la actividad es 0, pagado de PagadoActividad, siempre es cierto.

- SOCIO: Representa un socio de la asociación, contiene su nombre, apellido, teléfono, email, y si es estudiante de la UPC.
- CUOTA: Es el pago de la anualidad que deberá realizar un socio.
- ACTIVIDAD: Simboliza un evento de la asociación, cuenta con un nombre, la descripción, el precio, un lugar (como un aula concreta o una sala) y una ubicación (dirección física).
- PAGOACTIVIDAD: Pago que realiza el socio por asistir a una actividad.
- BALANCE: Representa un conjunto de registros, tiene como atributo la suma de los precios de los registros.
- REGISTRO: es un registro contable, contiene un título, una descripción, la imagen, un precio, una fecha y la identificación del ultimo usuario que modifico el registro.

5.2 Esquema del comportamiento

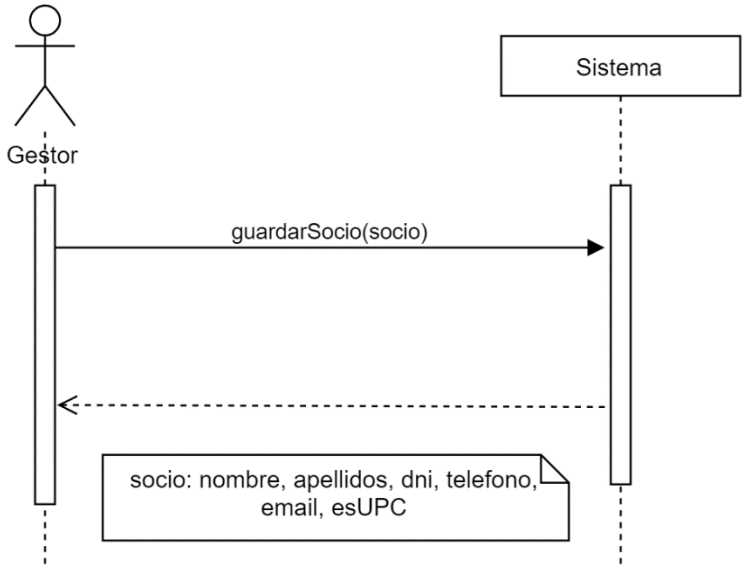
En este apartado se muestra la secuencia de eventos entre el actor y el sistema de los casos de uso definidos en el apartado anterior, de esta manera podremos ver que operaciones tendrá nuestra aplicación.

5.2.1 CU #01 Ver el listado de socios



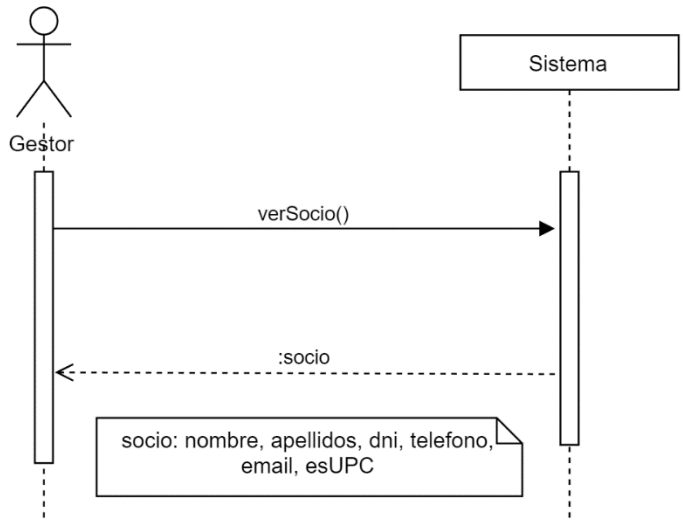
- Context Sistema::mostrarSocios()
- Pre -
- Post return = conjunto de socios de la asociación

5.2.2 CU #02 Nuevo socio



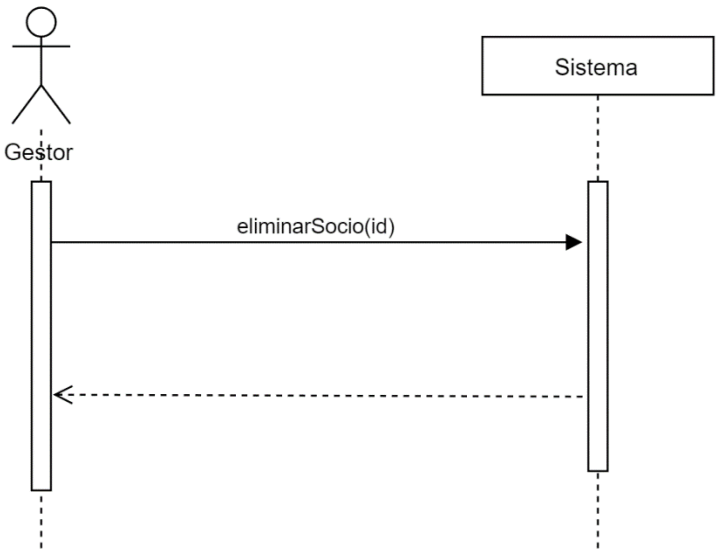
- Context Sistema::guardarSocio(socio)
- Pre Los datos de los socios no son nulos y tienen el formato correcto
- Post El sistema crea un nuevo socio y lo guarda en el sistema

5.2.3 CU #03 Ver socio



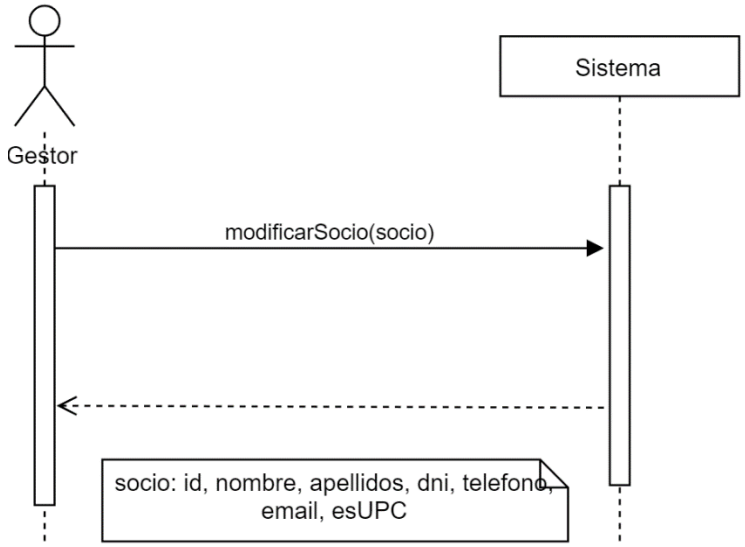
- Context Sistema::verSocios()
- Pre -
- Post return = socio de la asociación

5.2.4 CU #04 Eliminar socio



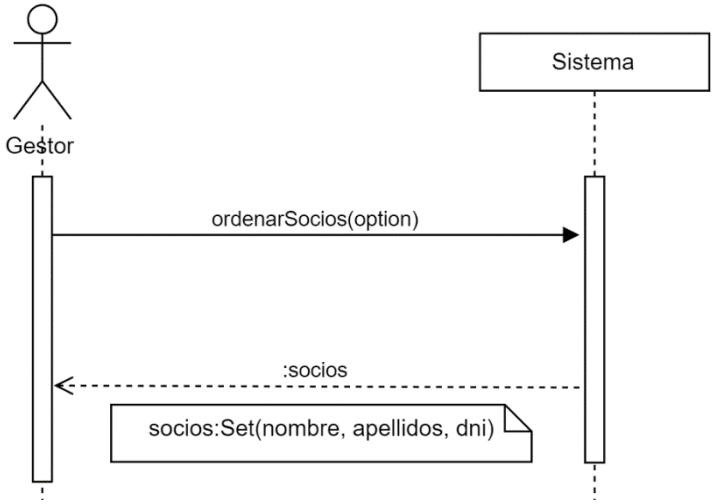
- Context Sistema::eliminarSocios(id)
- Pre El id del socio existe.
- Post El socio con id = socio.id se elimina del sistema.

5.2.5 CU #05 Modificar socio



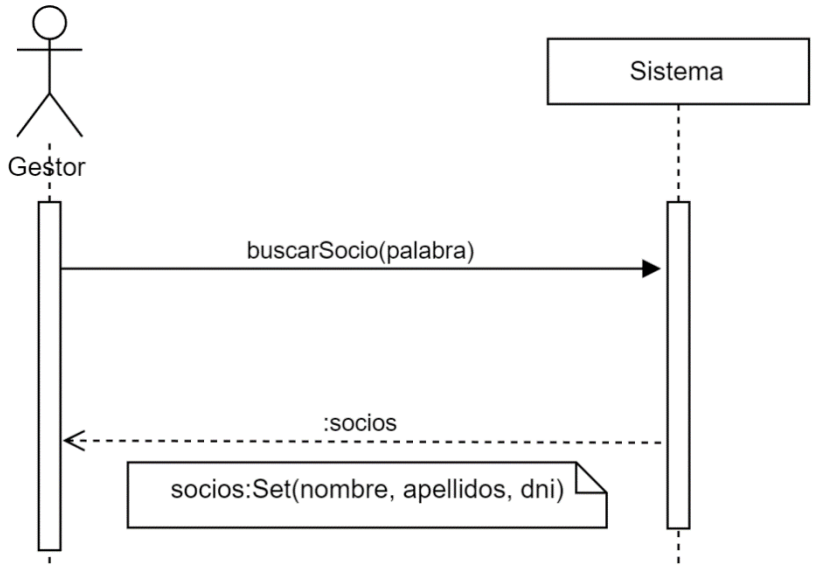
- Context Sistema::modificarSocio(socio)
- Pre Los datos de los socios no son nulos, tienen el formato correcto y id del socio existe en el sistema.
- Post El sistema modificar los datos del socio donde socio.id = id

5.2.6 CU #06 Ordenar socios



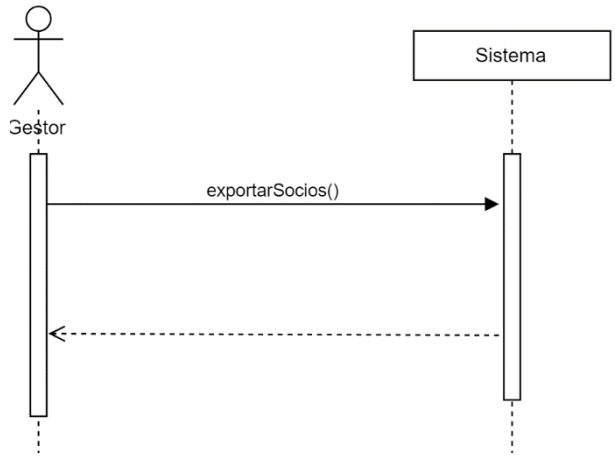
- Context Sistema::ordenarSocios(option)
- Pre -
- Post El sistema devuelve los socios ordenados por la opción seleccionada.

5.2.7 CU #07 Buscar socio



- Context Sistema::buscarSocio(palabra)
- Pre -
- Post El sistema muestra los socios que contengan palabra en nombre, apellido o dni

5.2.8 CU #08 Exportar socios



- Context Sistema::exportarSocio(palabra)
- Pre -
- Post El sistema crea un EXCEL para almacenarlo en una plataforma en la nube, como Drive o OneDrive.

6 Diseño

En este apartado se mostrará el diseño final que ha tenido la aplicación.

6.1 Arquitectura

AgentM se basará en una aplicación móvil que estará conectada mediante internet a un servidor para gestionar las peticiones, el cual, estará comunicado con una base de datos donde se guardará toda la información.

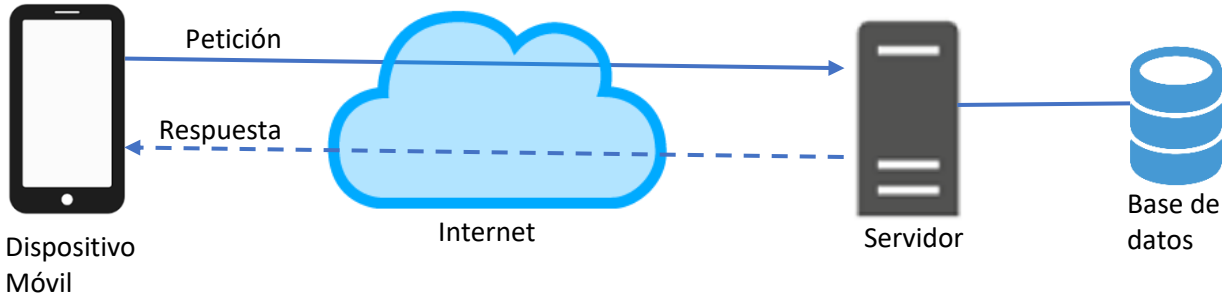


Ilustración 5. Diseño Arquitectura

La aplicación utilizara el patrón Modelo Vista Controlador (MVC) para separar las diferentes capas del sistema.

- La capa de datos, desarrollada en la base de datos, se encargará de la persistencia de los datos.
- La capa de dominio, desarrollada en el dispositivo móvil y en el servidor, se encargará de gestionar los datos entre la interfaz y la capa de datos.
- La capa de presentación, estará desarrollada en el dispositivo móvil, y tendrá la finalidad de mostrar la información y gestionar los eventos.

6.2 Diseño Azure

Antes de comenzar a ver en detalle como es el diseño de cada capa, vamos a explicar cómo será el diseño que tendrá nuestro backend desarrollado en Azure y que herramientas se utilizarán.

Para desarrollar el backend, los encargados de la junta sugirieron utilizar los servicios de Azure de Microsoft.

Azure ofrece muchos servicios y muchas herramientas para desarrollar diferentes tipos de programas. Para nuestra aplicación hemos necesitado los siguientes recursos:

- **Mobile App [18]:** Un backend escalable y seguro para desarrollar aplicaciones móviles.
- **Base de datos SQL [19]**: Base de datos relacional escalable para aplicaciones.
- **Azure Active Directory [20]**: Servicio de administración de identidades para aplicaciones.
- **Storage Account – blob [21]**: Solución escalable para el almacenamiento en la nube.

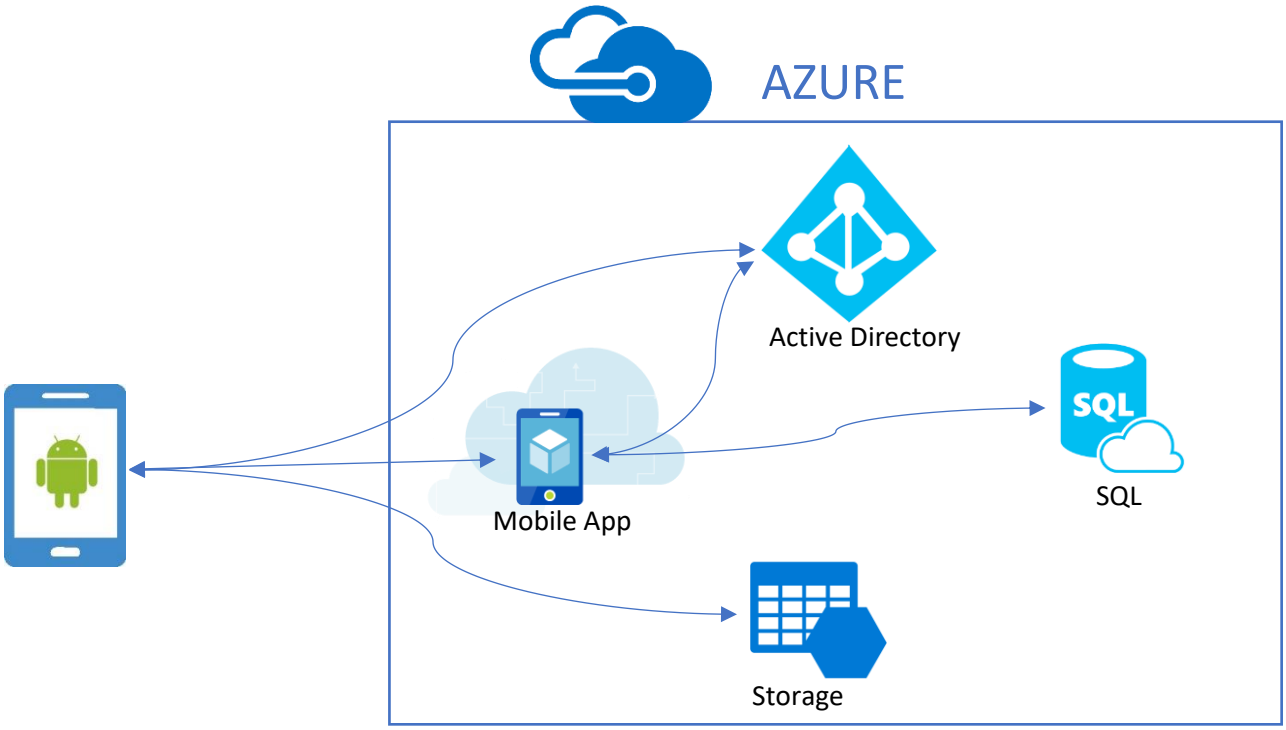


Ilustración 6.Backend Azure.

6.3 Diseño de la capa de datos

6.3.1 Base de datos

La capa de datos tendrá la base de datos de Azure, a continuación, se explicará el diseño de las tablas y sus atributos.

Todas las tablas de las bases de datos de Azure tienen que contener los siguientes campos:

- id -> identificador único del registro.
- *createAt* -> fecha en la que se creó el registro.
- *updateAt* -> fecha de la última modificación del registro.
- *version* -> campo utilizado para la sincronización de registros.
- *deleted* -> campo que indica si el registro esta eliminado

Estos campos son controlados por la base de datos de Azure. Por este motivo se ha explicado ahora en vez de explicarlos en cada tabla.

- Actividad (id, createdAt, updateAt, version, deleted, nombre, descripcion, fecha, precio, lugar, ubicacion).
    - nombre -> es el nombre de la actividad.
    - descripcion -> es un campo en el cual se describe la actividad que se realizará.
    - fecha -> indica el momento en el que se realizara la actividad.
    - precio -> indica el valor que se tendrá que pagar para realizar la actividad.
    - lugar -> expresa el sitio concreto donde se realizará la actividad.
    - ubicacion -> es la dirección física donde se realizará una actividad.

- Registro (id, createdAt, updateAt, version, deleted, titulo, descripcion, imagen, fecha, precio, modifiedBy)
    - titulo -> representa el nombre del registro.
    - descripcion -> es un resumen del contenido del registro.
    - imagen -> contiene el *path* de la imagen del registro.
    - fecha -> es la fecha en la que se realizó la transacción.
    - precio -> es el importe del registro.
    - modifiedBy -> es el id del usuario que modifico por última vez el registro.

- Socio (id, createdAt, updateAt, version, deleted, nombre, apellidos, dni, telefono, email, esUPC)
    - nombre -> es el nombre del socio.
    - apellidos -> son los apellidos del socio.
    - dni -> representa el NIF del socio.
    - telefono -> es el número de teléfono del socio.
    - email -> es la dirección de correo electrónica del socio.
    - esUPC -> indica si el socio es estudiante de la UPC.

- SocioActividad (id, createdAt, updateAt, version, deleted, pagado, socio_id, actividad_id)
    - pagado -> indica si el socio ha pagado el evento.
    - socio_id -> representa el socio que quiere ir a la actividad.
    - actividad_id -> representa la actividad.

- CuotasSocio (id, createdAt, updateAt, version, deleted, pagado, year, id_socio)
    - pagado -> indica si la cuota de ese socio esta pagada.
    - year -> es el año de la cuota.
    - socio_id -> es el socio.

6.3.2 Azure Active Directory

Para almacenar las identidades de los gestores utilizaremos Azure Active Directory.

Azure Active Directory, es una funcionalidad disponible en Azure, que nos permite integrar una solución de administración de identidades a nuestro servicio utilizando el protocolo de autenticación OAuth 2.0[22].

Los usuarios que necesiten utilizar la aplicación necesitarán una cuenta de Microsoft, esta cuenta se agregará al directorio de identidades de Active Directory, y así tener acceso a nuestro servicio Mobile App.

Un usuario al iniciar su aplicación desde su dispositivo físico, iniciará el protocolo de autenticación. El servicio Mobile App redirigirá al usuario a una URL de inicio de sesión en Active Directory. Una vez introducidas las credenciales correctamente, Active Directory le proporcionará a Mobile App el *id* y el *token* del usuario, que, a través de una URI de redirección, se le enviarán a la aplicación del dispositivo.

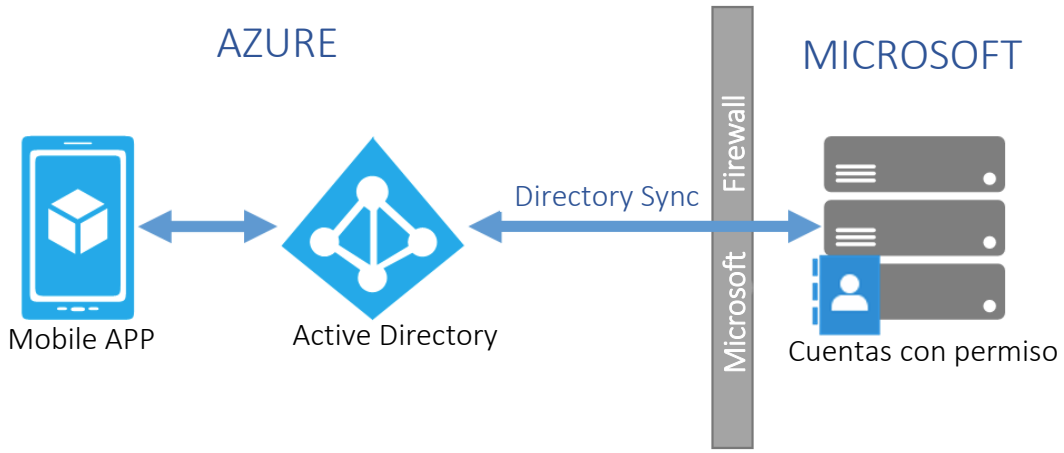


Ilustración 7.Azure Active Directory.

6.3.4 Storage Account

Para poder almacenar las fotografías utilizaremos Storage Account, que es el servicio de almacenamiento que nos ofrece Azure.

Este servicio nos permite crear contenedores, donde se crean y almacenan archivos denominados *blobs.* Este fichero *blob* puede ser de tipo de datos binario, texto, documento, archivo multimedia o instalador de aplicaciones. La forma de identificar a los *blobs* es por el contenedor y su nombre, por lo tanto, no podremos tener dos archivos con el mismo nombre en el mismo contenedor. Para proteger la información, este servicio cuenta con un método de seguridad de claves.

Storage Account, se necesita para poder almacenar las fotografías que pertenecen a los registros de la aplicación Android. Al guardar un registro, si tiene fotografía, a esta se le asigna un nombre identificativo, se almacena en el contenedor *imagenes*, y además se almacena su nombre en la base de datos. Al acceder a la información de un registro desde el dispositivo móvil, se recupera el nombre de la fotografía de la base de datos y se descarga la imagen del servicio de almacenamiento.

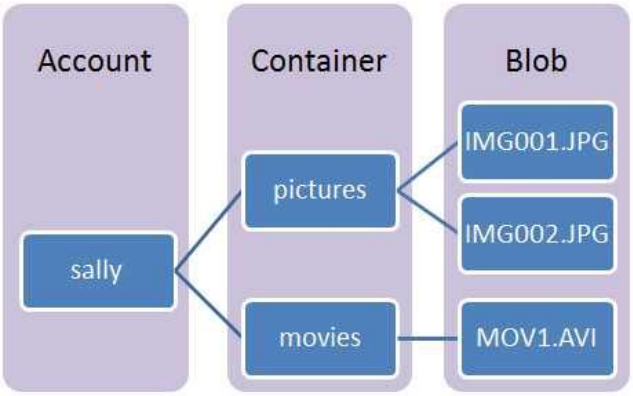


Ilustración 8.Azure Storage Account.

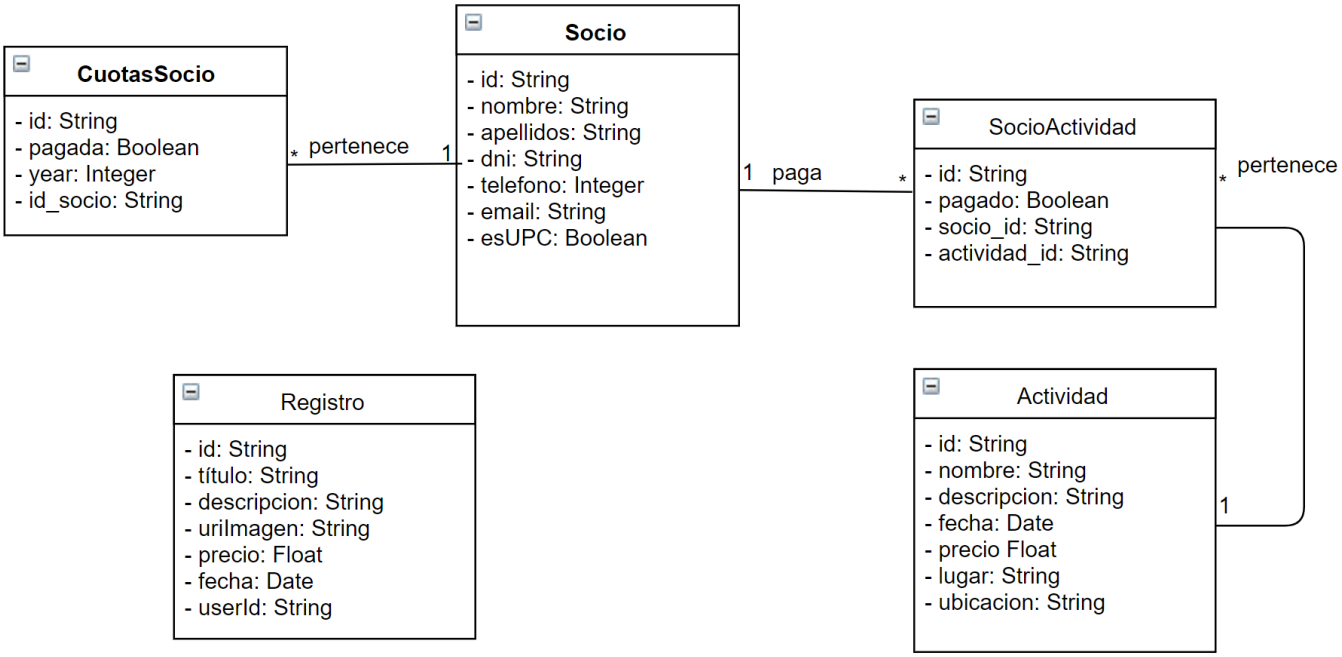
6.4 Diseño de la capa de dominio

En este apartado, veremos el diseño que tiene la capa de dominio.

La capa de dominio estará distribuida entre el servidor Mobile App y la aplicación Android.

6.4.1 Diagrama de clases

Este es el diagrama de clases que tendrá la capa de dominio, y lo tendrá tanto el servidor como Android.



6.4.2 Mobile App

El servicio Mobile Apps, es el servidor para aplicaciones móviles que ofrece Azure. En nuestro caso escrito en lenguaje Node.js y dispondrá de una API para que desde el dispositivo móvil se puedan realizar solicitudes.

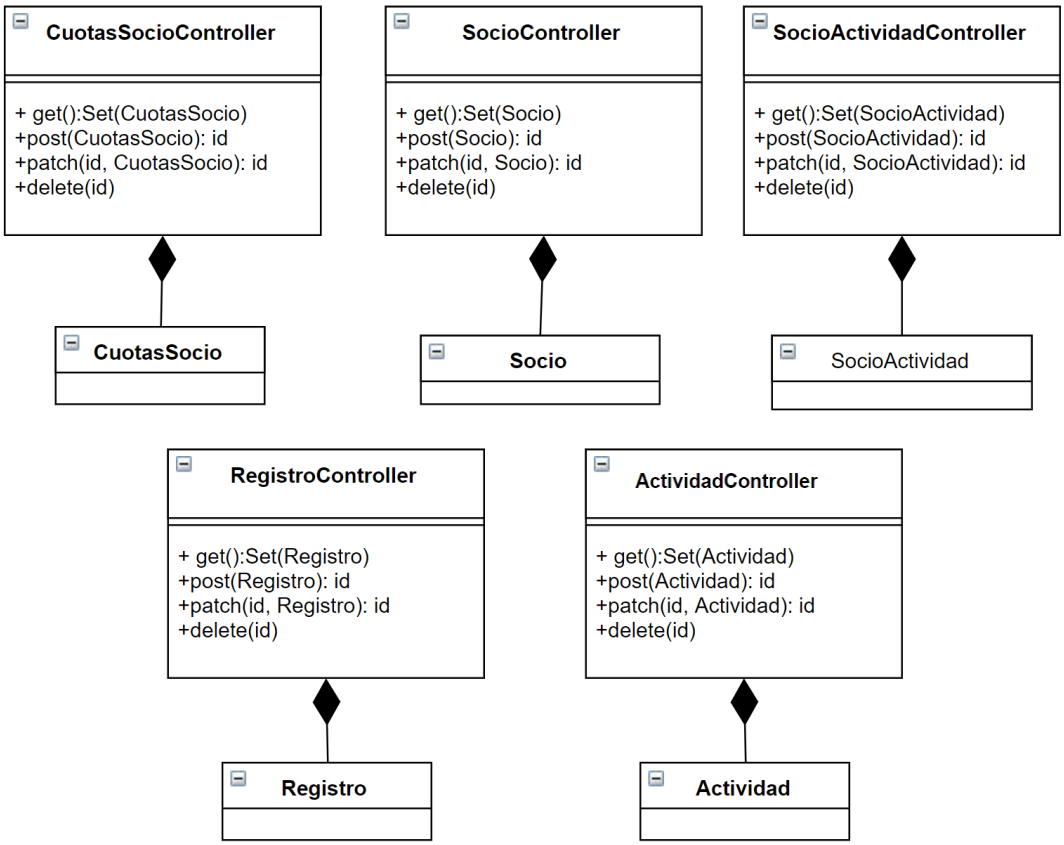
Mobile Apps estará conectado con la base de datos y realizará las operaciones de las solicitudes que le lleguen a través de la API. Este servicio también estará conectado con Active Directory, para comprobar que previamente los clientes se hayan autentificado.

La API que nos ofrece Mobile Apps cuenta con la ZUMO-API-VERSION 2.0.0 y las operaciones que se pueden realizar son las siguientes:

- GET: Solicitar información al servidor.
- POST: Insertar nueva información al servidor.
- PATCH: Actualizar información del servidor.
- DELETE: Para eliminar información del servidor. En nuestro *server* se realiza una *soft delet*, nuestra base de datos contiene un bit que indica si el registro esta eliminado.

Todas las operaciones requieren un cuerpo HTTP y se codifican en formato JSON.

Este es el diagrama de las clases de Mobile App. Vemos como cada clase del modelo, tiene su propio controlador, encargado de gestionar las peticiones.



6.4.3 Android

La aplicación se desarrollará para dispositivos Android y estará conectada a los servidores de Azure descritos anteriormente.

Para desarrollar la aplicación AgentM para los dispositivos, además de utilizar el SDK de Android, se utilizará el SDK Azure Mobile App, que nos permite autentificarnos y conectarnos al Mobile App, y el SDK Azure Storage, que nos permitirá gestionar los *blobs*.

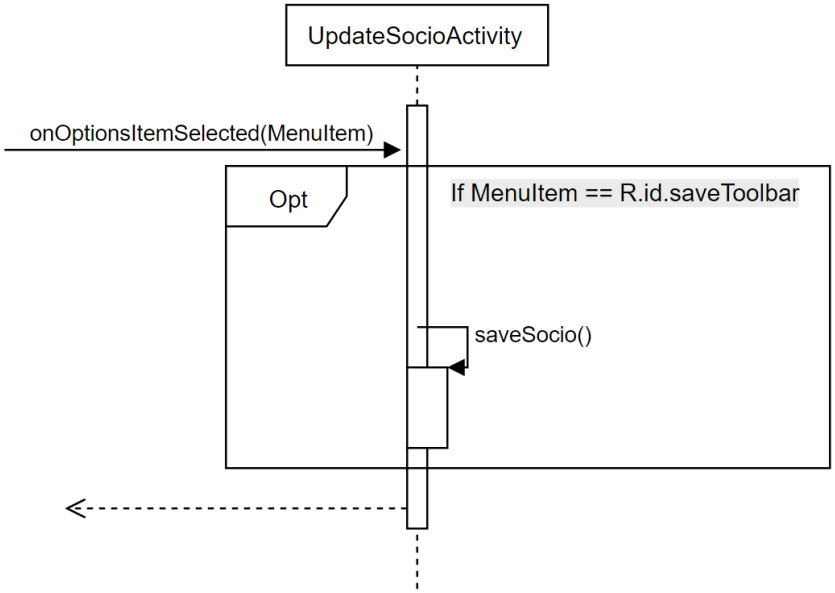
6.4.3.1 SDK Mobile App

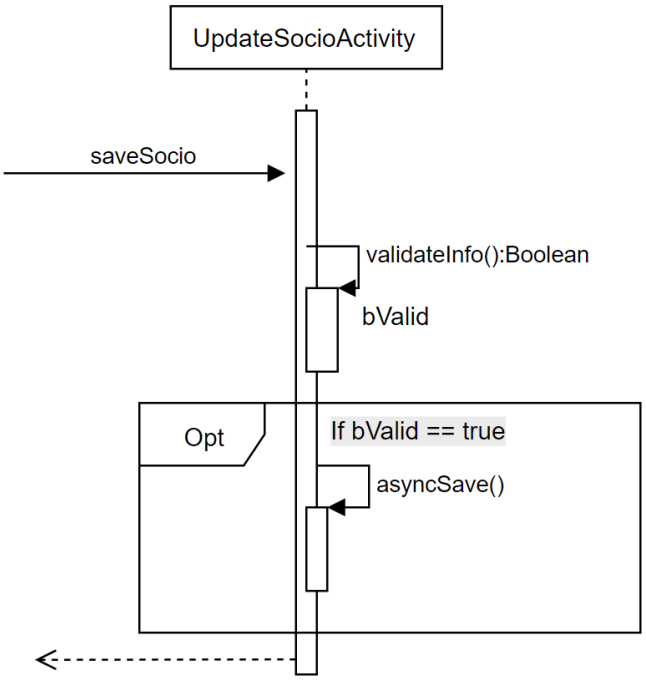
El SDK Mobile App, como hemos comentado, nos permitirá conectarnos al servidor, autentificarnos y hacer peticiones a la base de datos. Para ello, este SDK nos facilita las clases MobileServiceClient y MobileServiceTable.

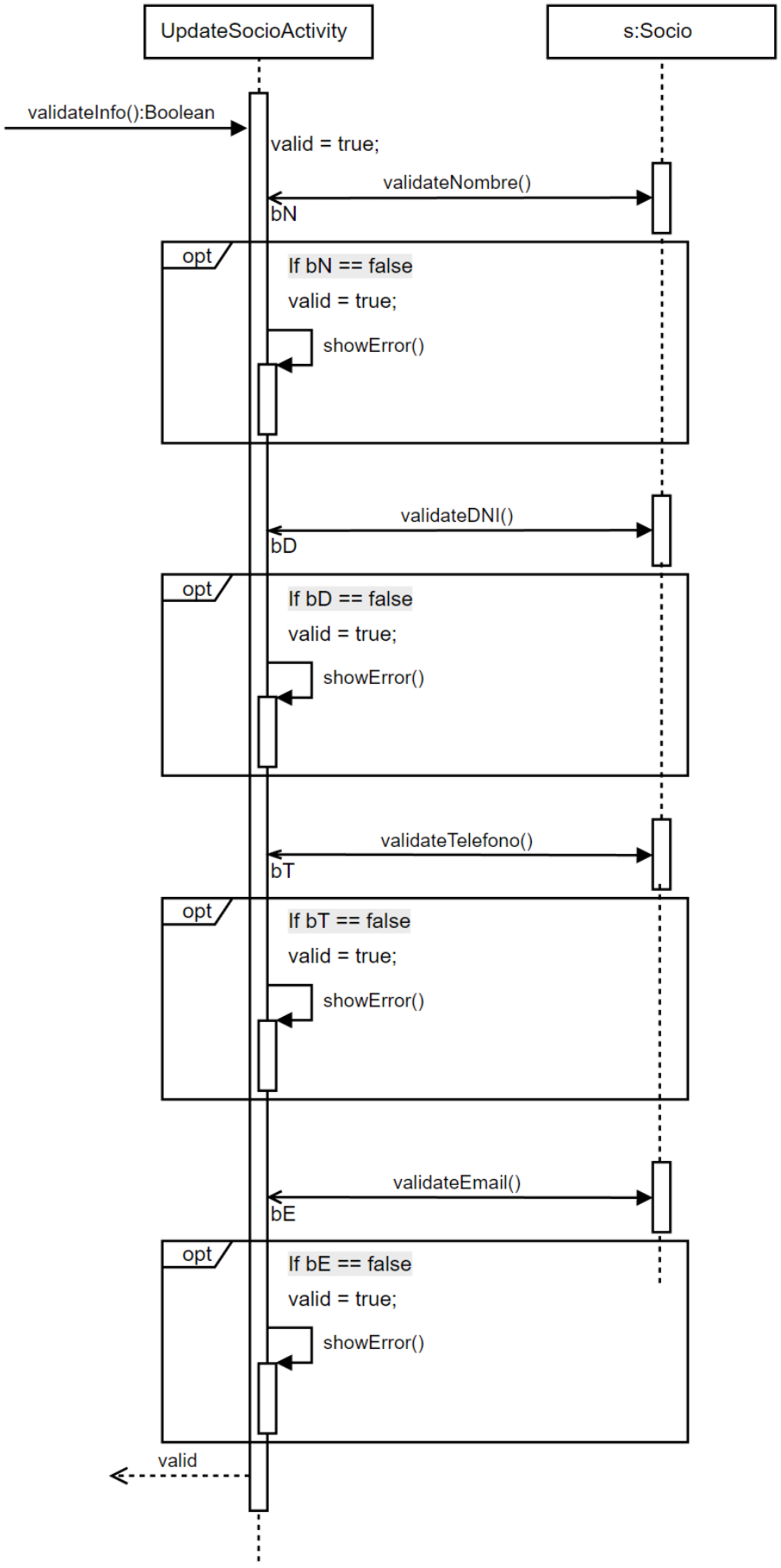
MobileServiceClient nos permitirá realizar la conexión con Mobile App, además, contiene el usuario cliente, lo que nos permitirá autentificarnos almacenando el id y el token del usuario. Esta clase también nos permitirá solicitar los MovileServiceTable.

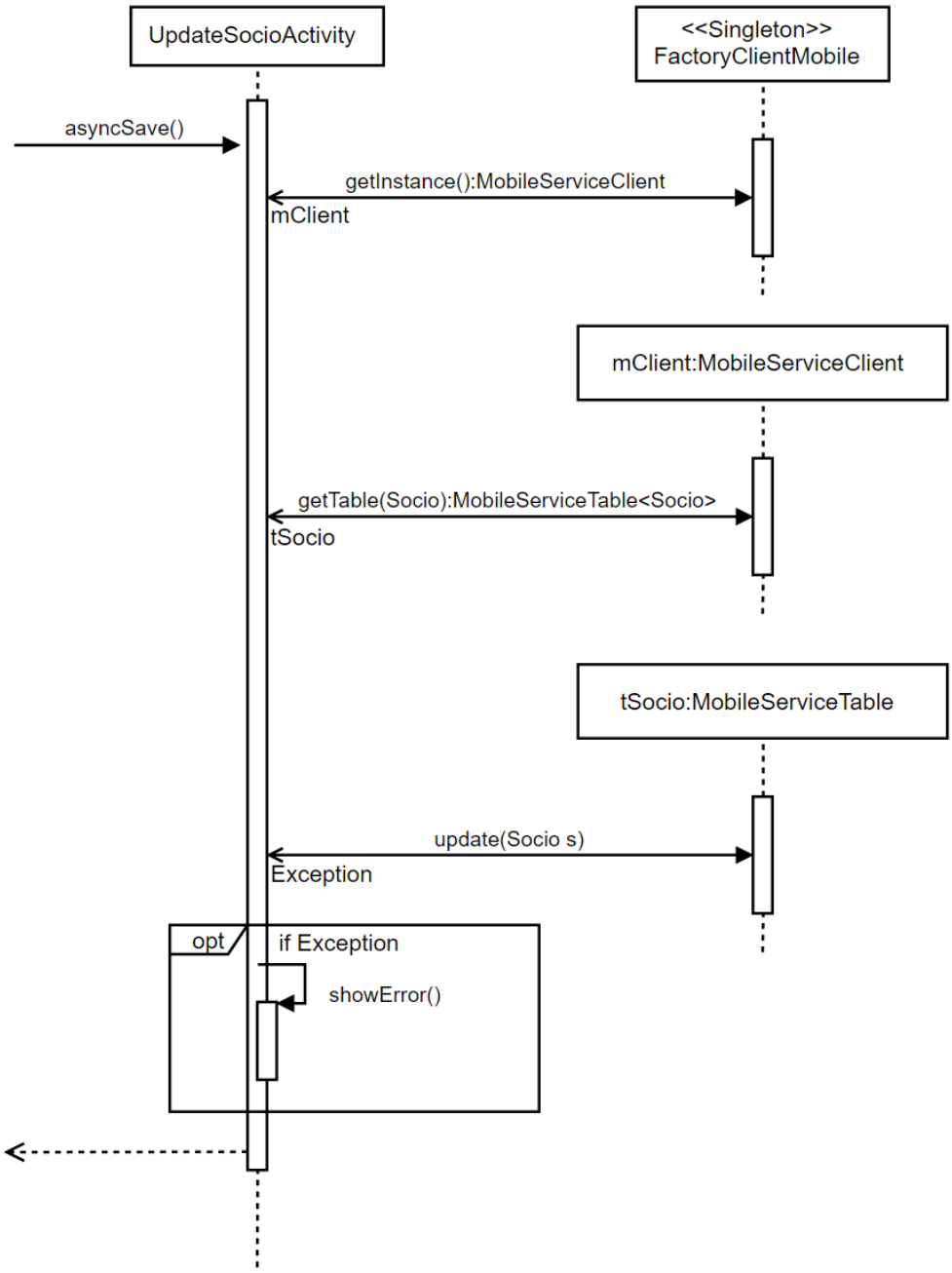
Para solo tener que configurar una vez este servicio se ha creado la clase FactoryClientMobile, que será una instancia que contendrá MobileServiceClient.

A continuación, se muestra el diagrama de secuencia de la función actualizar un socio, desde el momento en que llega al controlador el evento de guardar la información. Con este claro ejemplo veremos el patrón factoría y como se utiliza el SDK de Mobile App.


UpdateSocioActivity
onOptionsItemSelected(MenuItem)
Opt
If MenuItem == R.id.saveToolbar
saveSocio()


UpdateSocioActivity
saveSocio
validateInfo():Boolean
bValid
Opt
If bValid == true
asyncSave()


UpdateSocioActivity
s:Socio
validateInfo():Boolean
valid = true;
validateNombre()
bN
opt
If bN == false
valid = true;
showError()
validateDNI()
bD
opt
If bD == false
valid = true;
showError()
validateTelefono()
bT
opt
If bT == false
valid = true;
showError()
validateEmail()
bE
opt
If bE == false
valid = true;
showError()
valid

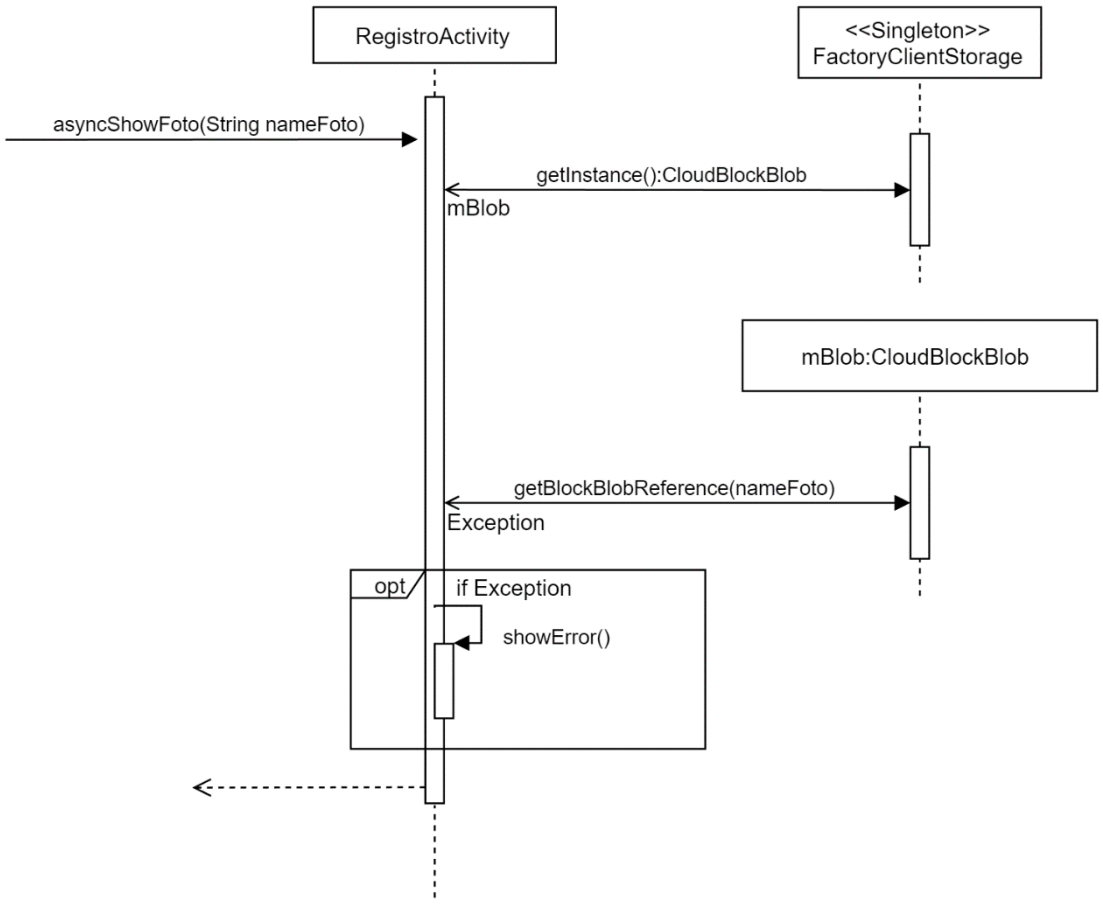

UpdateSocioActivity
<<Singleton>>
FactoryClientMobile
asyncSave()
getInstance():MobileServiceClient
mClient
mClient:MobileServiceClient
getTable(Socio):MobileServiceTable<Socio>
tSocio
tSocio:MobileServiceTable
update(Socio s)
Exception
opt
if Exception
showError()

6.4.3.2 SDK Azure Storage

La aplicación móvil dispondrá de la clase FactoryClientStorage, esta clase *singleton* tendrá un comportamiento similar que FactoryClientMobile. Nuestra aplicación solo necesitará conectarse al contendor *imagenes* del Storage Account, por lo que en la función de inicialización configurará la conexión a la cuenta y al contenedor.

Posteriormente se crearán los blobs cuando sea necesario para pedir o enviar información.

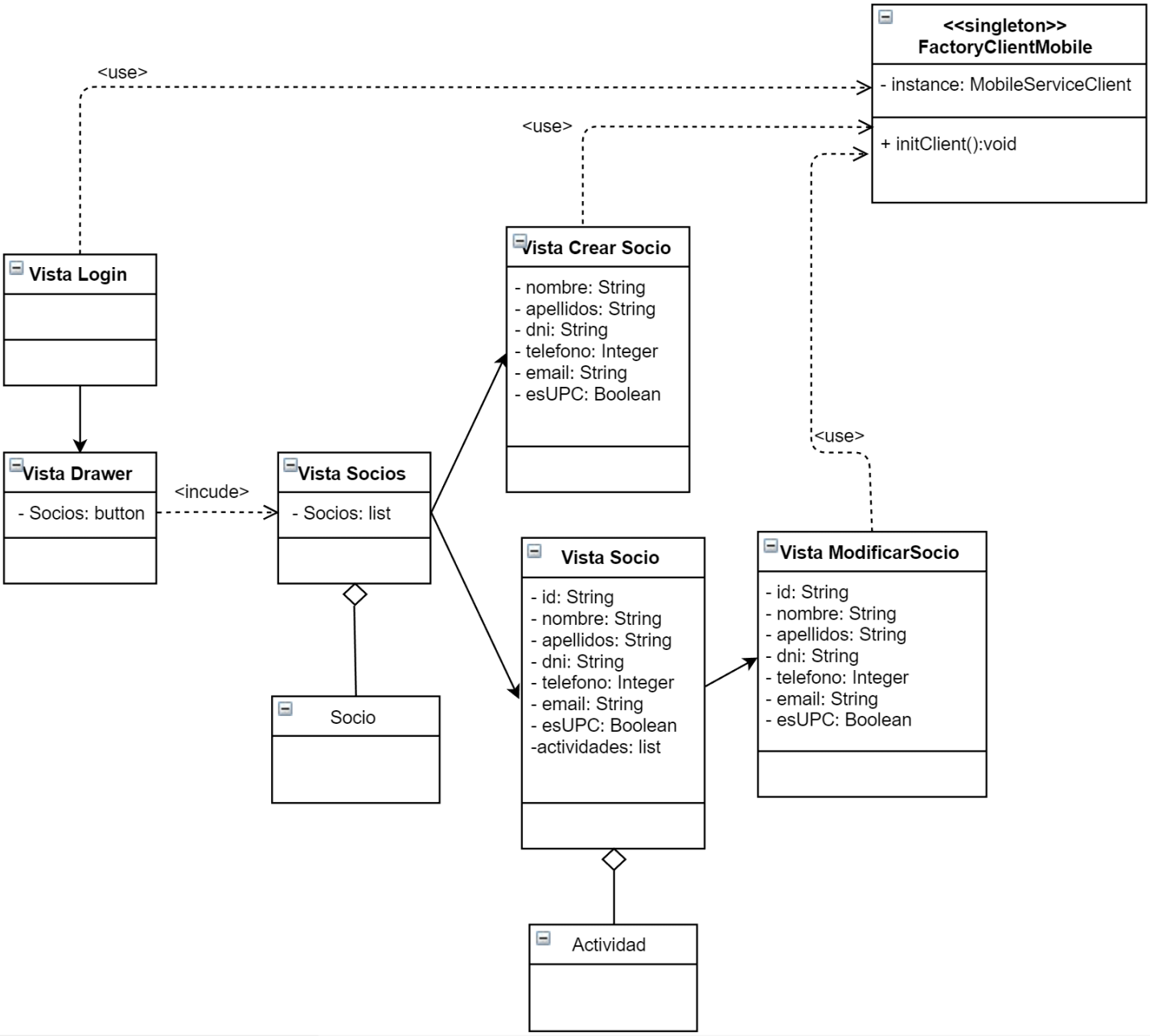
En este diagrama se muestra cómo se obtiene una imagen del servidor.



6.5 Diseño de la capa de presentación

6.5.1 Diseño interno

Al utilizar el patrón MVC, cada vista de AgentM tendrá su propio controlador. En el diagrama de vistas siguiente podremos ver un ejemplo de las vistas pertenecientes a las funciones de gestión de los socios.



6.5.2 Diseño externo

En este apartado se muestran las diferentes pantallas de la aplicación y se explicarán las funcionalidades.

6.4.4.1 Diseño y Usabilidad

Para que la aplicación tenga un buen diseño, se ha personalizado la aplicación para que tenga unos tonos azules y grises, se han utilizados los iconos de Android [23], y se ha personalizado la fuente en algunas funcionalidades.

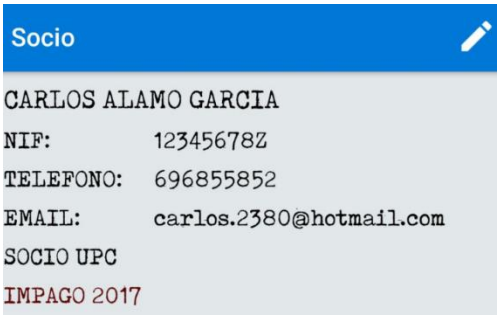


Ilustración 10.Fuente e iconos.

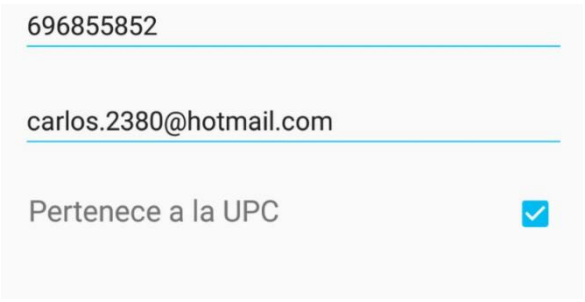


Ilustración 9.Tonos azulados y grises.

Para facilitar la visualización de las listas a los socios, se han alternado el color de las filas, además de poder ocultar algunas funcionalidades para poder mostrar más elementos de las listas.

Ilustración 12.Entrelazar colores.

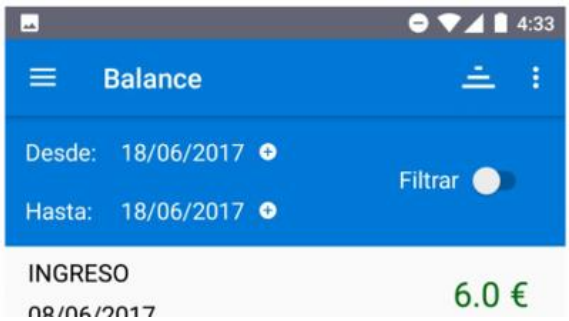


Ilustración 11.Funcionalidad visible.

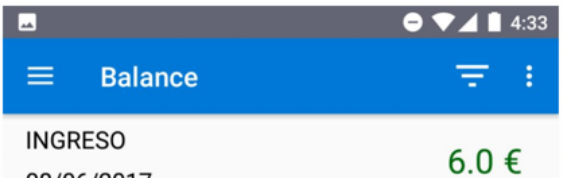


Ilustración 13.Funcionalidad oculta.

Para evitar errores del usuario al introducir texto, las horas y las fechas se introducen a través de un calendario y un reloj. Las casillas para introducir texto que sean numéricas, solo permiten introducir valores numéricos de su tipo, y asi evitarle al usuario cometer más errores.

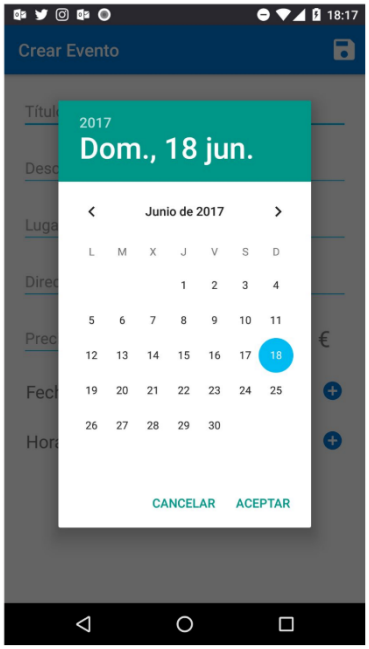


Ilustración 16.Calendario.

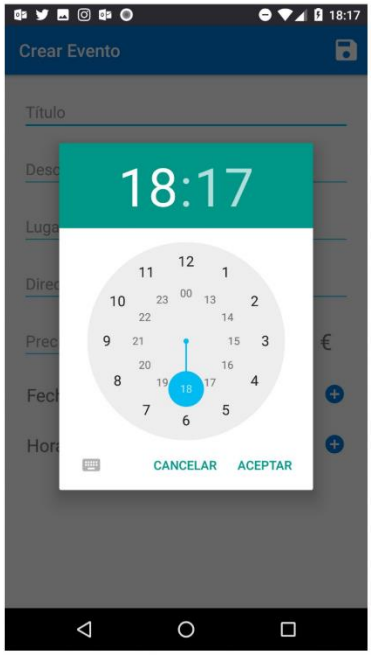


Ilustración 15.Reloj.

Ilustración 14.Teclado personalizado.

Finalmente, al guardar información se comprobará que la información introducida sea correcta y se le informara al usuario para que pueda ver donde se ha equivocado.

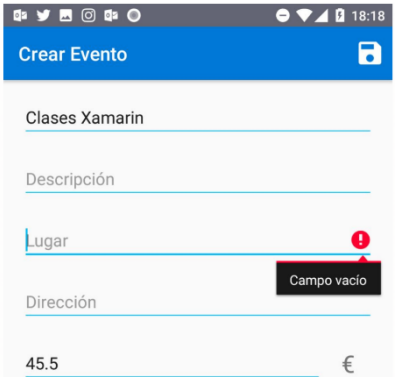


Ilustración 17.Información de error.

6.4.4.2 Inicio de sesión

Al iniciar la aplicación se mostrará un webview con la pantalla de inicio de sesión de Microsoft.

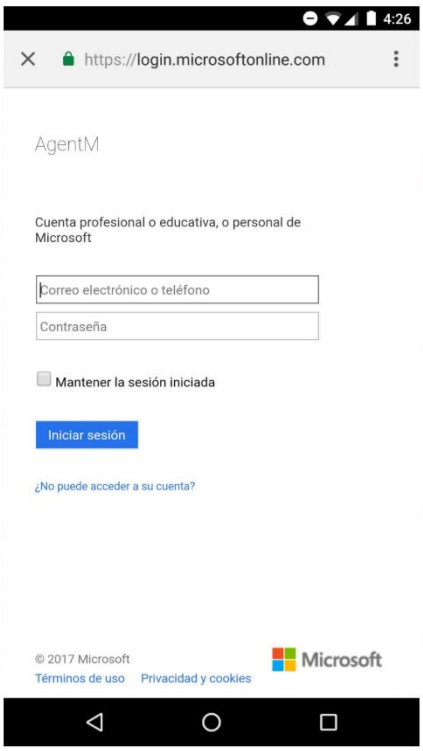


Ilustración 18. Inicio sesión.

Al acceder correctamente las credenciales se mostrarán las demás funcionalidades de la aplicación.

6.4.4.3 Menú lateral

Se diferencian claramente seis funcionalidades distintas, por lo que se ha añadido un menú lateral para poder acceder a cada una de ellas.

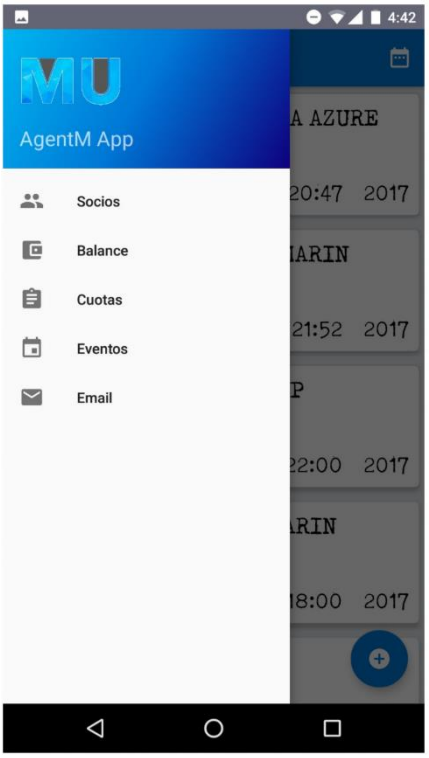


Ilustración 19.Menú lateral.

- Socios: accedemos a las funcionalidades de gestión de los socios.
- Balance: accedemos a las funcionalidades de los registros.
- Cuotas: para gestionar el pago de la anualidad de los socios.
- Eventos: gestionar los eventos y los socios que participan.
- Email: enviar un email global a todos los socios de la asociación.

6.4.4.4 Socios

Al acceder a socios veremos un listado de los socios que tiene la asociación. Podremos buscar a un socio por el nombre, apellido o NIF, además de poder ordenar el listado por nombre, apellido o NIF y exportar el resultado en EXCEL.

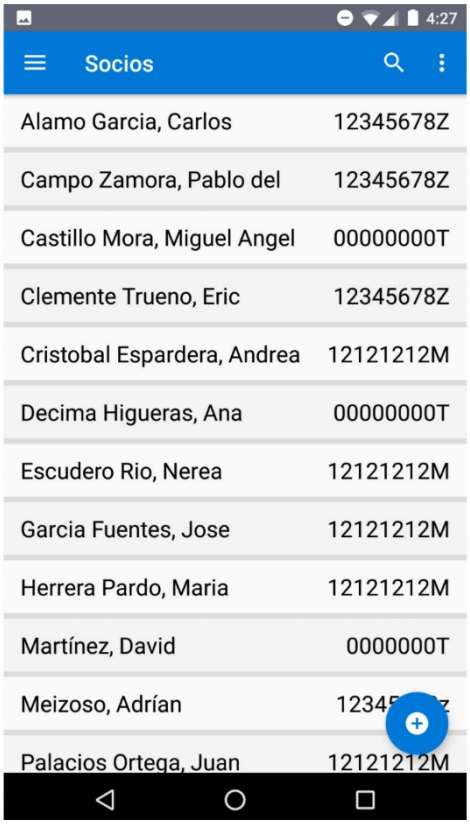


Ilustración 20.Listado Socios.

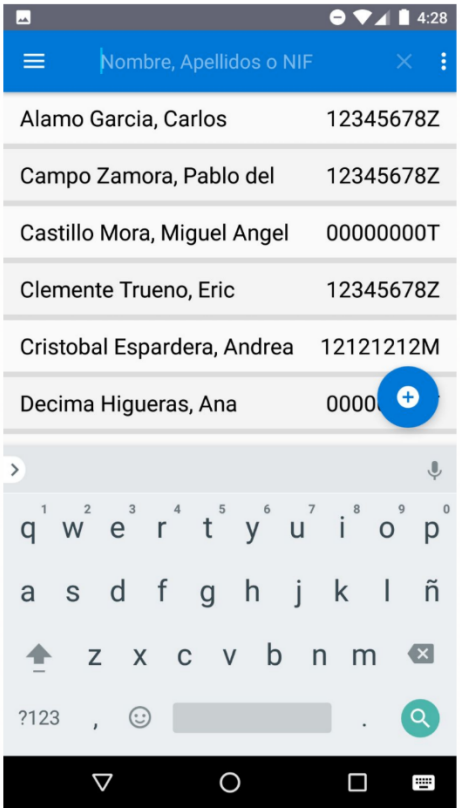


Ilustración 21.Buscar socios.

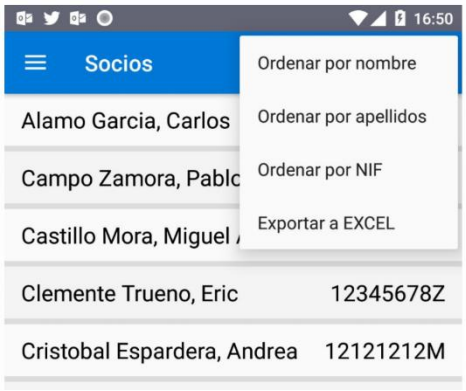


Ilustración 22. Ordenar socios.

Al clicar en el botón de añadir socios, veremos el formulario para añadir nuevos socios.

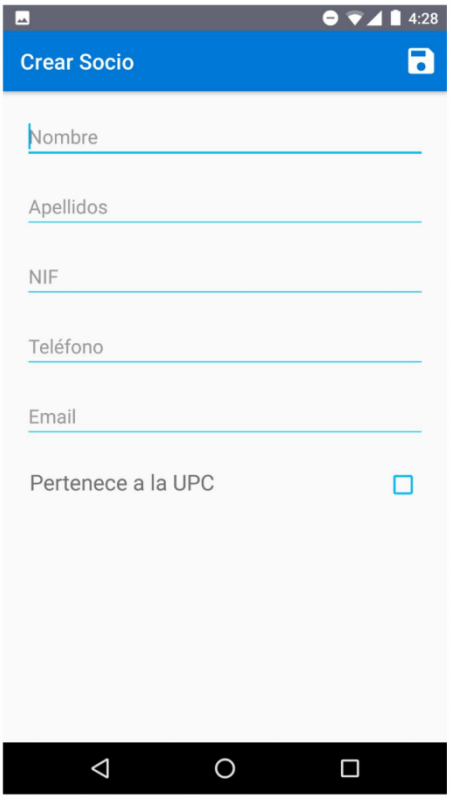


Ilustración 23.Crear socio.

Si clicamos en un socio del listado, accederemos a su información personal, además de ver si no ha pagado alguna cuota y las actividades que ha realizado.

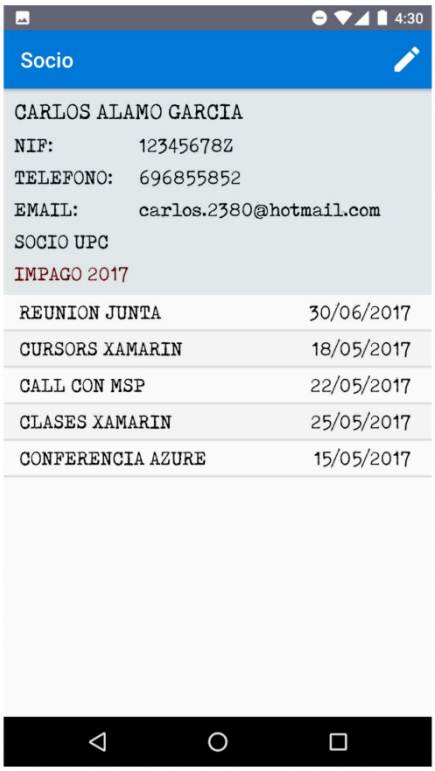


Ilustración 24.Informacíon socio.

Al clicar en modificar, podremos modificar la información personal del socio o eliminar el socio. Al eliminar un socio, tendremos que confirmar la acción.

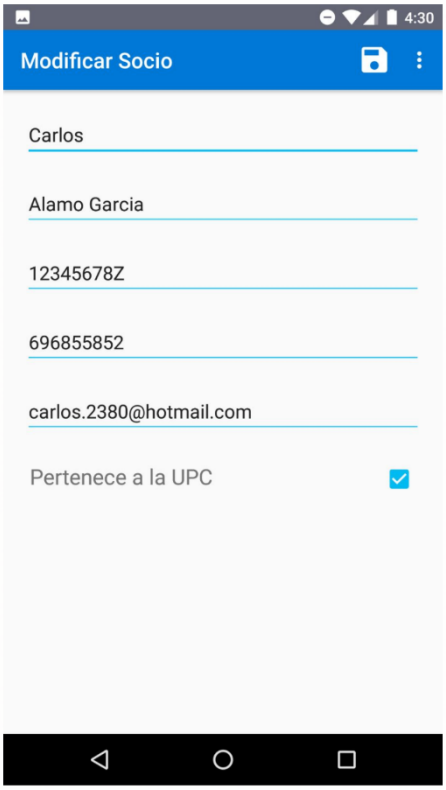


Ilustración 25.Modificar socio.

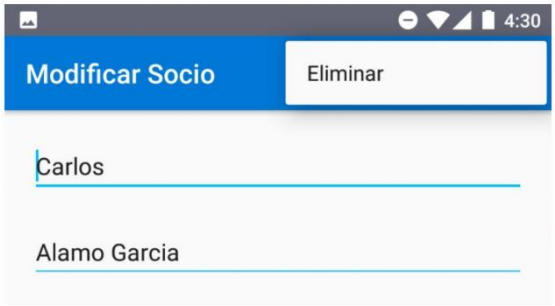


Ilustración 26.Eliminar socio.

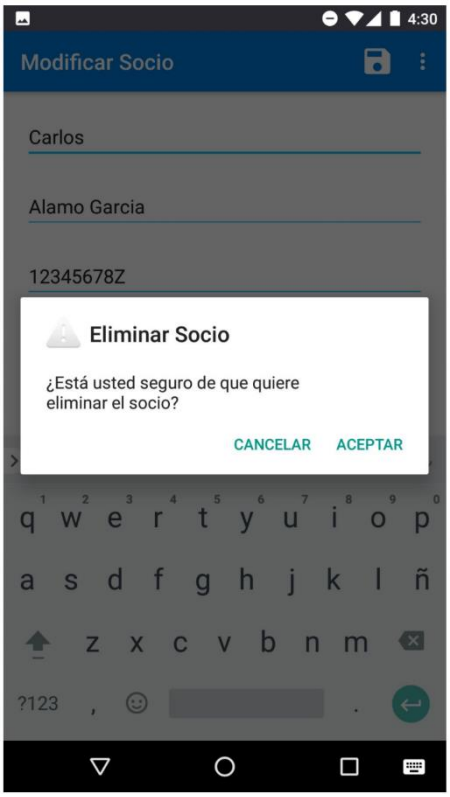


Ilustración 27.Confirmar eliminar socio.

6.4.4.5 Balance

Al acceder al balance veremos un listado con los registros, el total de la suma del valor de los registros mostrados y un botón para añadir más registros.

Ilustración 28.Balance.

En la barra de herramientas, hay el botón para mostrar u ocultar los filtros.

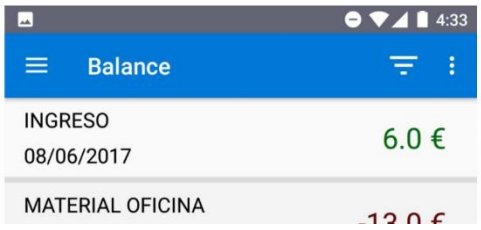


Ilustración 29.Filtros balance ocultos.

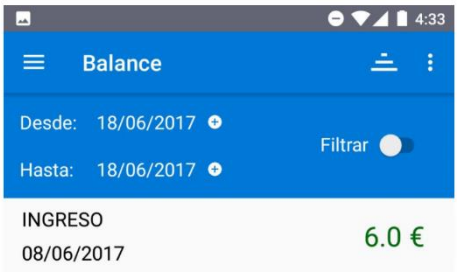


Ilustración 30.Filtro balance visibles.

En la sección de filtros podremos elegir la fecha desde donde queremos que se muestren los registros hasta la fecha final, la interfaz no permite que la fecha inicial sea posterior a la final. Al activar filtrar, se filtrarán los registros, los filtros seguirán activos, aunque la barra este oculta, hasta que se desactive filtrar.

También podremos exportar los balances mostrados a Excel.

Ilustración 31.Balance a EXCEL.

Al acceder a la funcionalidad de nuevo registro, se mostrará la plantilla para crear un nuevo registro.

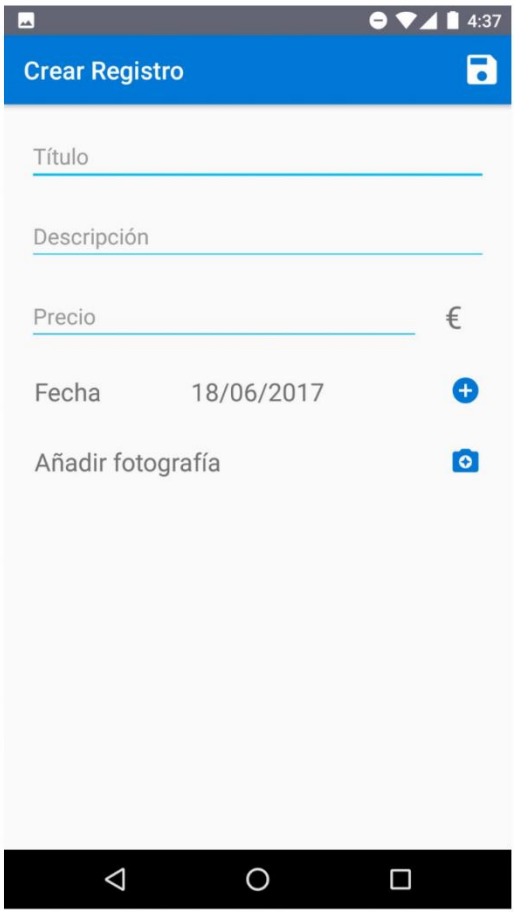


Ilustración 32.Crear Registro.

Al acceder a un registro veremos la información del registro y la fotografía.

Ilustración 33. Registro 1/2.

Ilustración 34.Registro 2/2.

Si accedemos a la pantalla de modificación podemos modificar o eliminar el registro.

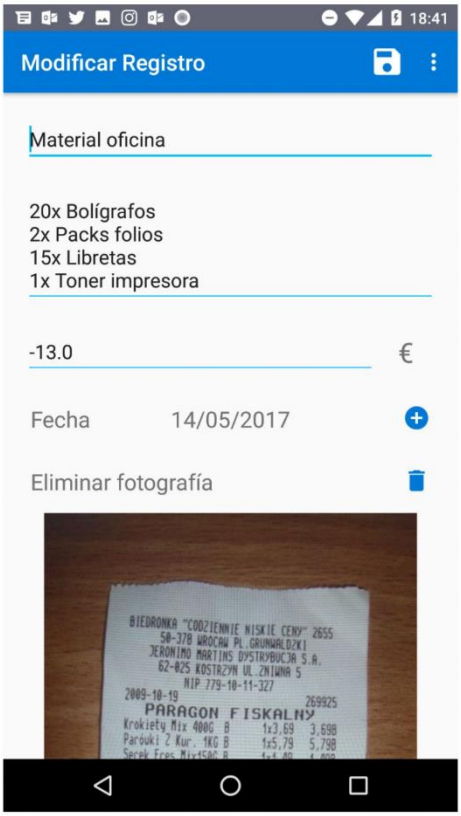


Ilustración 36.Modificar Registro.

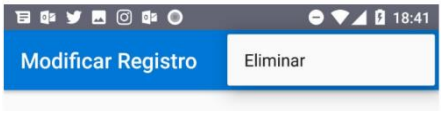


Ilustración 35.Eliminar registro.

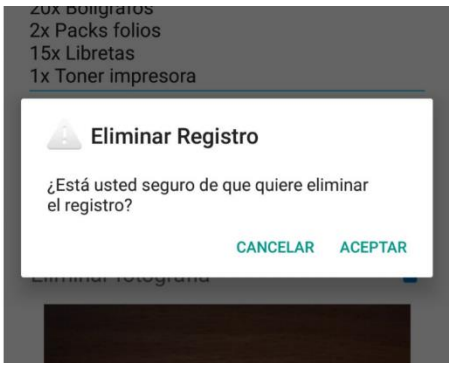


Ilustración 37.Confirmar eliminar registro.

6.4.4.6 Cuotas

En el apartado de cuotas, gestionamos el pago de las cuotas anuales de los socios.

Al acceder vemos el registro de cuotas del año actual. Los filtros están ocultos y los podemos hacer visibles como en el ejemplo del balance.

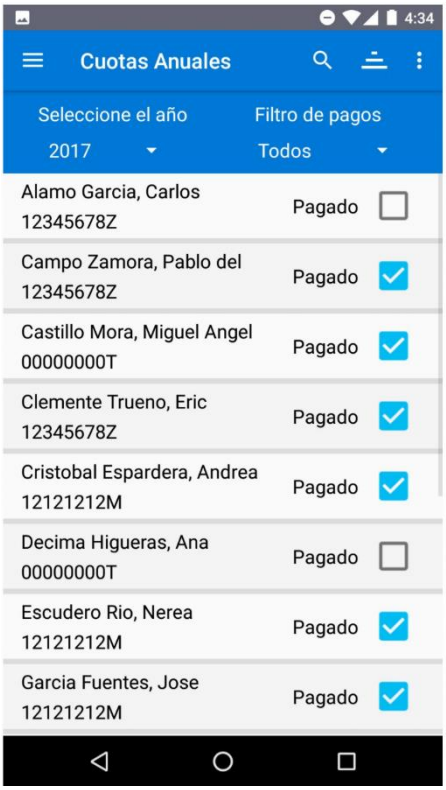


Ilustración 38.Cuotas.

En los filtros podremos elegir el año y el si queremos ver todos los socios, los pagados o los impagados.

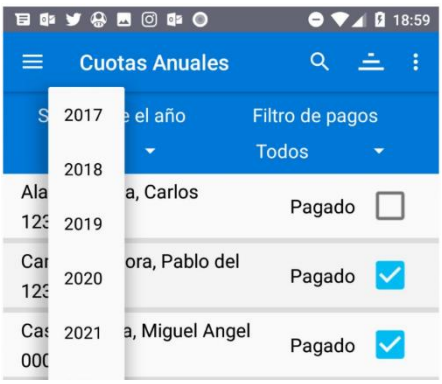


Ilustración 40.Seleccionar año.

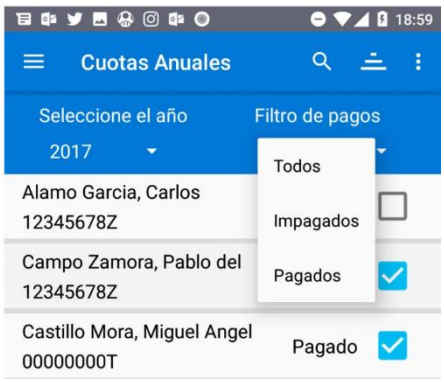


Ilustración 39.Seleccionar pagos.

En el listado de cuotas también tenemos la opción de buscar un socio por nombre, apellidos o NIF. Además, podremos ordenar el listado de socios y exportar las cuotas a Excel.

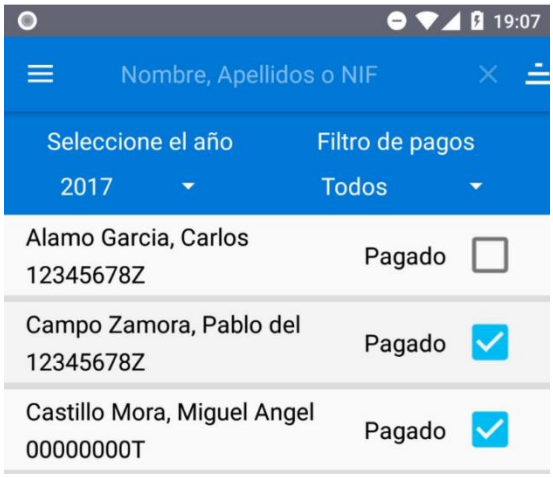


Ilustración 42.Buscar cuota.

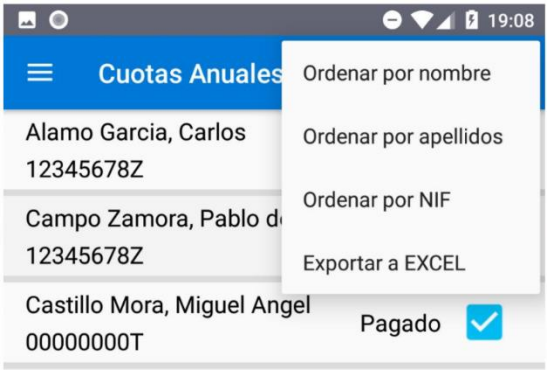


Ilustración 41.Ordenar cuotas.

Finalmente podremos confirmar o deshacer el pago de la cuota de un socio o eliminar socios que no hayan pagado el evento.

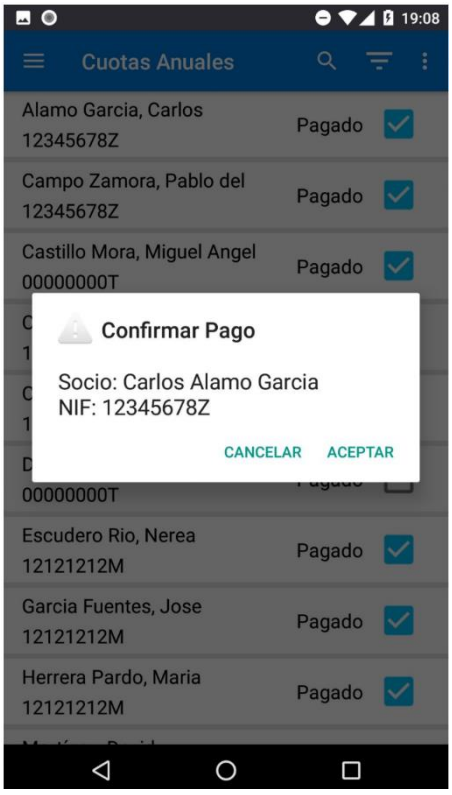


Ilustración 45.Confirmar pago.

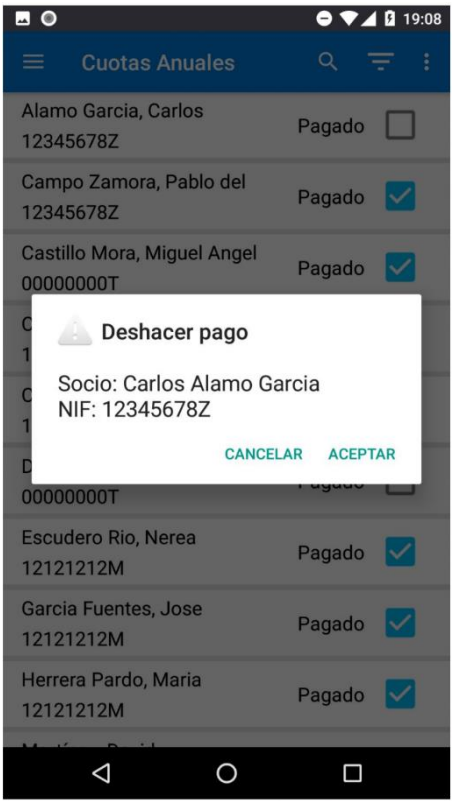


Ilustración 44.Deshacer pago.

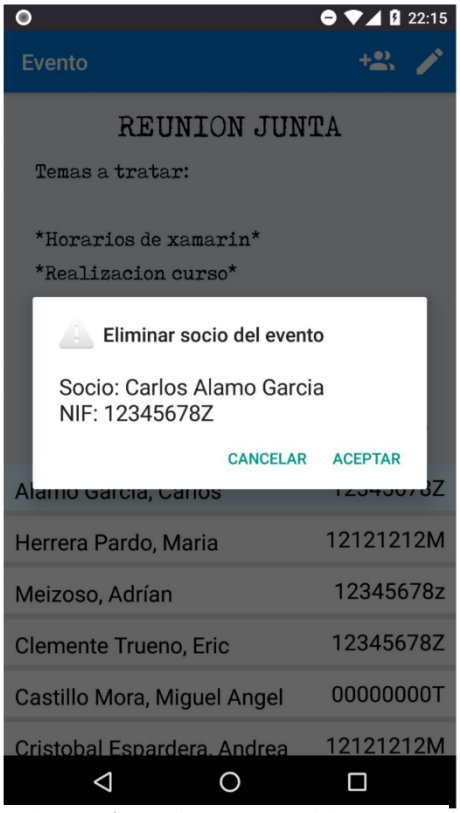


Ilustración 43.Eliminar socio del evento.

6.4.4.7 Eventos

Al acceder a los eventos, se mostrará un listado de los eventos futuros de la asociación.

Si clicamos en el icono del calendario, veremos un calendario con los días marcados que tienen eventos. Al clicar en un día, veremos los eventos de ese día. Para volver al listado simplemente tenemos que clicar en el botón del listado.

Ilustración 47.Listado de eventos.

Ilustración 46.Calendario de eventos.

Al clicar en un nuevo evento se mostrará el formulario para poder crear un nuevo evento.

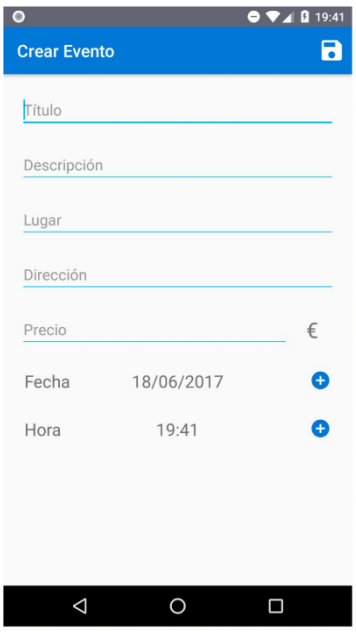


Ilustración 48.Crear evento.

Al clicar en un evento podemos acceder a toda la información del evento, ver los socios que participan, y en el caso de ser de pago, ver que socios han pagado. Según sea de pago o no, la lista que se muestra tendrá la funcionalidad de gestionar los pagos.

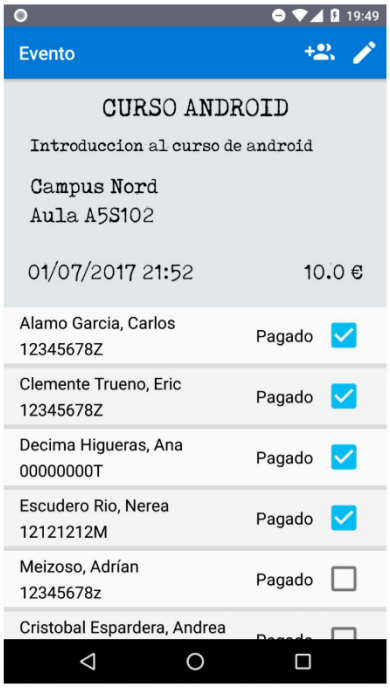


Ilustración 50.Evento de pago.

Ilustración 49.Evento gratuito.

Si deslizamos la lista de los socios podemos ocular la información del evento para que la lista de socios sea más amplia y se muestren más socios.

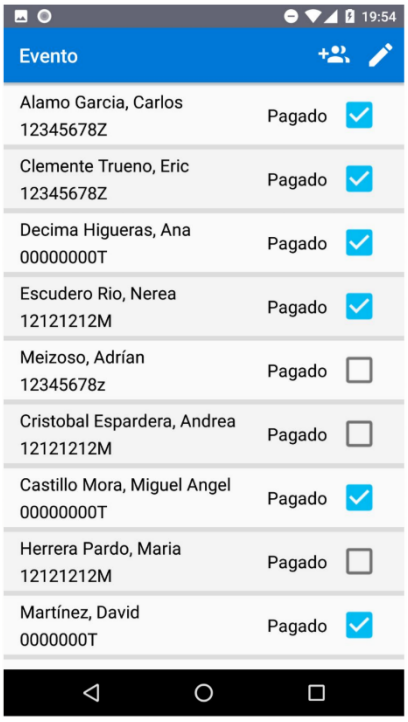


Ilustración 51.Ocultar información evento.

Si clicamos al botón añadir socios, navegamos a la pantalla de añadir socios, que contiene una lista de los socios que no participan en el evento.

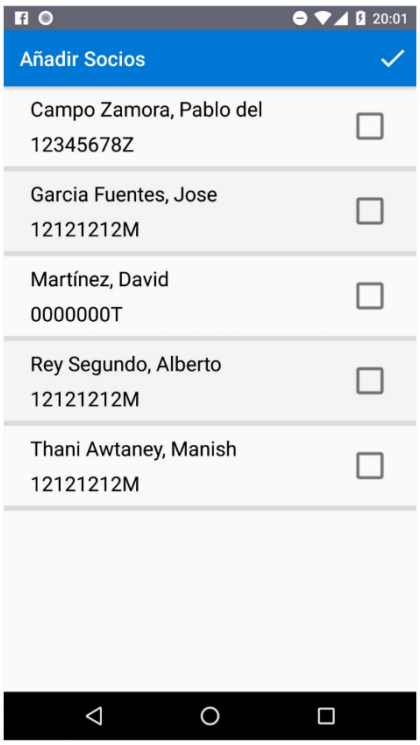


Ilustración 52.Añadir socios.

Desde la vista del evento, si clicamos en el botón editar, navegaremos a la pantalla de modificar evento, donde podremos modificar o eliminar el evento.

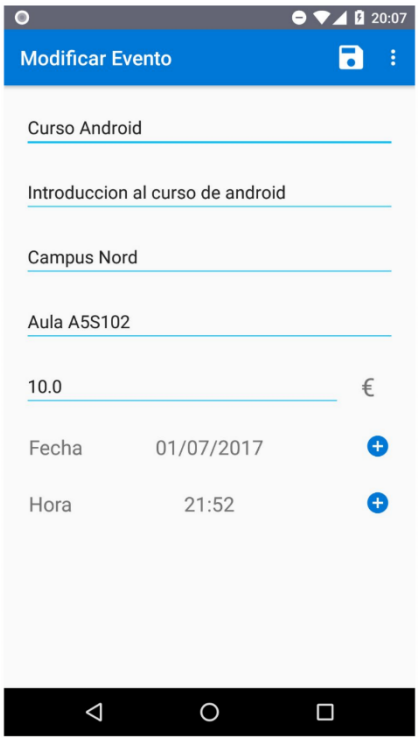


Ilustración 54.Modificar evento.

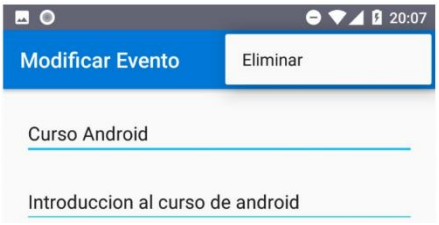


Ilustración 53.Eliminar evento.

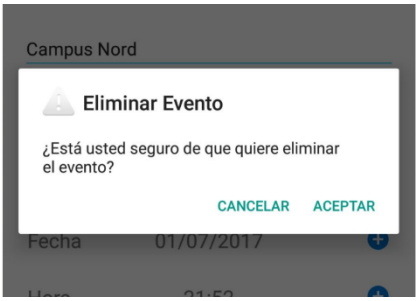


Ilustración 55.Confirmar eliminar evento.

6.4.4.8 Email

En la funcionalidad email, podemos escribir un asunto y un contenido, al enviarlo, en el programa de envió de emails, además del asunto y contenido redactado, tendrá como destinatarios a todos los socios.

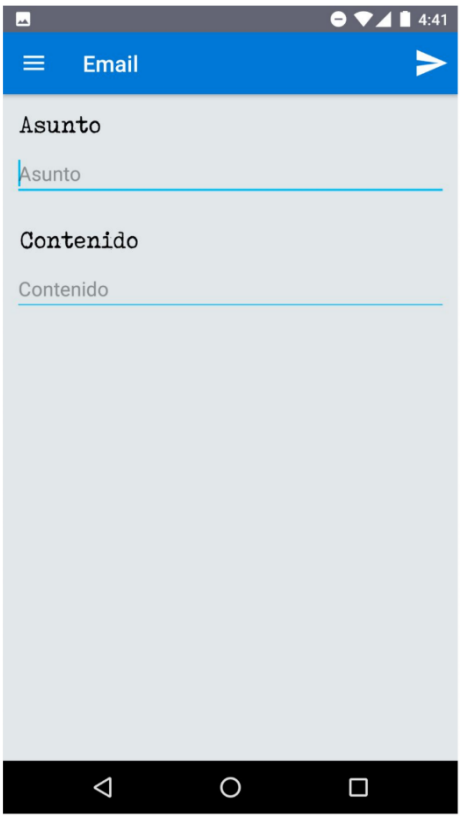


Ilustración 57.Email.

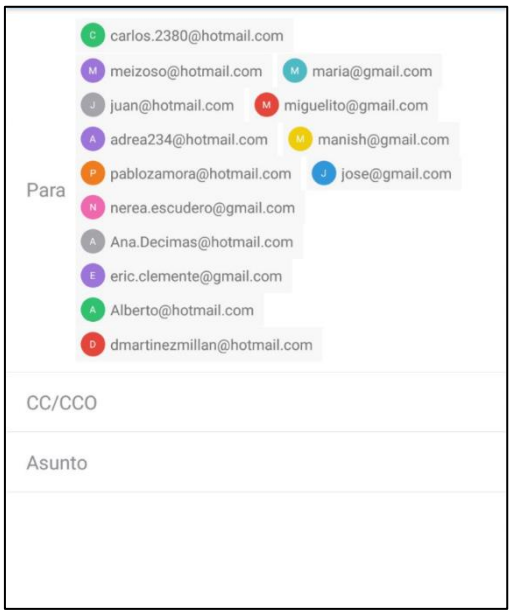
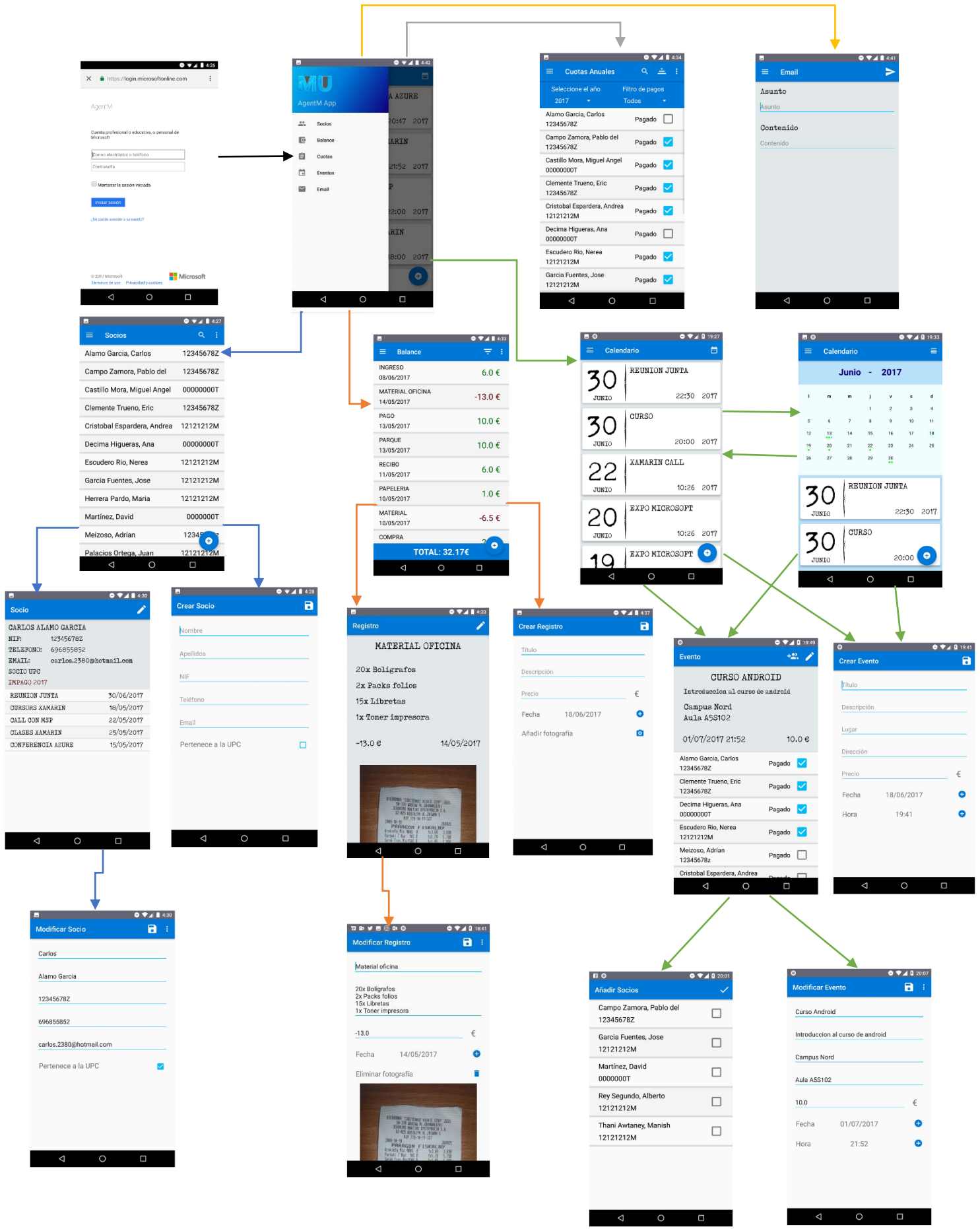


Ilustración 56.Destinatarios.

6.4.4.9 Navegabilidad

En este apartado veremos las navegabilidades entre las pantallas explicadas anteriormente.


AgentM App
Socios
Balance
Cuotas
Eventos
Email
Cuotas Anuales
Email
Asunto
Contenido
Socios
Alamo Garcia, Carlos 12345678Z
Campo Zamora, Pablo del 12345678Z
Castillo Mora, Miguel Angel 00000000T
Clemente Trueno, Eric 12345678Z
Cristobal Espardera, Andrea 12121212M
Decima Higueras, Ana 00000000T
Escudero Rio, Nerea 12121212M
Garcia Fuentes, Jose 12121212M
Herrera Pardo, Maria 12121212M
Martínez, David 0000000T
Meizoso, Adrian
Palacios Ortega, Juan
Balance
INGRESO 08/06/2017 6.0 €
MATERIAL OFICINA 14/05/2017 -13.0 €
PAGO 13/05/2017 10.0 €
PARQUE 13/05/2017 10.0 €
RECIBO 11/05/2017 6.0 €
PAPELERIA 10/05/2017 1.0 €
MATERIAL 10/05/2017 -6.5 €
COMPRA
TOTAL: 32.17€
Calendario
30 JUNIO REUNION JUNTA 22:30 2017
30 JUNIO CURSO 20:00 2017
22 JUNIO XAMARIN CALL 10:26 2017
20 JUNIO EXPO MICROSOFT 10:26 2017
19 JUNIO EXPO MICROSOFT
Calendario
Junio - 2017
30 JUNIO REUNION JUNTA 22:30 2017
30 JUNIO CURSO 20:00
Socio
CARLOS ALAMO GARCIA
NIF: 12345678Z
TELEFONO: 696855852
EMAIL: carlos.2380@hotmail.com
SOCIO UPC
IMPAGO 2017
REUNION JUNTA 30/06/2017
CURSORS XAMARIN 18/05/2017
CALL CON MSP 22/05/2017
CLASES XAMARIN 25/05/2017
CONFERENCIA AZURE 15/05/2017
Crear Socio
Nombre
Apellidos
NIF
Teléfono
Email
Pertenece a la UPC
Registro
MATERIAL OFICINA
20x Bolígrafos
2x Packs folios
15x Libretas
1x Toner impresora
-13.0 € 14/05/2017
Crear Registro
Título
Descripción
Precio €
Fecha 18/06/2017
Añadir fotografía
Evento
CURSO ANDROID
Introduccion al curso de android
Campus Nord
Aula A5S102
01/07/2017 21:52 10.0 €
Alamo Garcia, Carlos 12345678Z Pagado
Clemente Trueno, Eric 12345678Z Pagado
Decima Higueras, Ana 00000000T Pagado
Escudero Rio, Nerea 12121212M Pagado
Meizoso, Adrian 12345678z Pagado
Cristobal Espardera, Andrea
Crear Evento
Título
Descripción
Lugar
Dirección
Precio €
Fecha 18/06/2017
Hora 19:41
Modificar Socio
Carlos
Alamo Garcia
12345678Z
696855852
carlos.2380@hotmail.com
Pertenece a la UPC
Modificar Registro
Material oficina
20x Bolígrafos
2x Packs folios
15x Libretas
1x Toner impresora
-13.0 €
Fecha 14/05/2017
Eliminar fotografía
Añadir Socios
Campo Zamora, Pablo del 12345678Z
Garcia Fuentes, Jose 12121212M
Martínez, David 0000000T
Rey Segundo, Alberto 12121212M
Thani Awtaney, Manish 12121212M
Modificar Evento
Curso Android
Introduccion al curso de android
Campus Nord
Aula A5S102
10.0 €
Fecha 01/07/2017
Hora 21:52

7 Implementación

En este capítulo veremos cómo se ha implementado la aplicación AgentM y que herramientas se han utilizado para su desarrollo.

7.1 Backend Azure

Como hemos ido mencionando, nuestra aplicación tendrá una implementación en los servidores de Azure.

7.1.1 Base de datos SQL

Para realizar la aplicación, disponemos de una base de datos SQL de 2GB de memoria de almacenamiento.

Se ha configurado una base de datos con las siguientes tablas y sus atributos:

```
CREATE TABLE [dbo].[Actividad] (
    [id]          NVARCHAR (255) NOT NULL,
    [createdAt]   DATETIMEOFFSET (3) NOT NULL,
    [updatedAt]   DATETIMEOFFSET (3) NULL,
    [version]     ROWVERSION         NOT NULL,
    [deleted]     BIT                DEFAULT ((0)) NULL,
    [nombre]      NVARCHAR (MAX)     NULL,
    [descripcion] NVARCHAR (MAX)     NULL,
    [fecha]       DATETIMEOFFSET (3) NULL,
    [precio]      FLOAT (53)         NULL,
    [lugar]       NVARCHAR (MAX)     NULL,
    [ubicacion]   NVARCHAR (MAX)     NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC)
);

CREATE TABLE [dbo].[Registro] (
    [id]          NVARCHAR (255)     NOT NULL,
    [createdAt]   DATETIMEOFFSET (3) NOT NULL,
    [updatedAt]   DATETIMEOFFSET (3) NULL,
    [version]     ROWVERSION         NOT NULL,
    [deleted]     BIT                DEFAULT ((0)) NULL,
    [titulo]      NVARCHAR (MAX)     NULL,
    [descripcion] NVARCHAR (MAX)     NULL,
    [imagen]      NVARCHAR (MAX)     NULL,
    [fecha]       DATETIMEOFFSET (3) NULL,
    [precio]      FLOAT (53)         NULL,
    [modifiedBy]  NVARCHAR (MAX)     NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC)
);
```

```
CREATE TABLE [dbo].[Socio] (
    [id]        NVARCHAR (255)     NOT NULL,
    [createdAt] DATETIMEOFFSET (3) NOT NULL,
    [updatedAt] DATETIMEOFFSET (3) NULL,
    [version]   ROWVERSION         NOT NULL,
    [deleted]   BIT                DEFAULT ((0)) NULL,
    [nombre]    NVARCHAR (MAX)     NULL,
    [dni]       NVARCHAR (MAX)     NULL,
    [telefono]  FLOAT (53)         NULL,
    [email]     NVARCHAR (MAX)     NULL,
    [esUPC]     BIT                NULL,
    [apellidos] NVARCHAR (MAX)     NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC)
);


CREATE TABLE [dbo].[SocioActividad] (
    [id]           NVARCHAR (255)     NOT NULL,
    [createdAt]    DATETIMEOFFSET (3) NOT NULL,
    [updatedAt]    DATETIMEOFFSET (3) NULL,
    [version]      ROWVERSION         NOT NULL,
    [deleted]      BIT                DEFAULT ((0)) NULL,
    [pagado]       BIT                NULL,
    [socio_id]     NVARCHAR (255)     NOT NULL,
    [actividad_id] NVARCHAR (255)     NOT NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC),
    FOREIGN KEY ([socio_id]) REFERENCES [dbo].[Socio] ([id]),
    FOREIGN KEY ([actividad_id]) REFERENCES [dbo].[Actividad] ([id])
);


CREATE TABLE [dbo].[CuotasSocio] (
    [id]        NVARCHAR (255)     NOT NULL,
    [createdAt] DATETIMEOFFSET (3) NOT NULL,
    [updatedAt] DATETIMEOFFSET (3) NULL,
    [version]   ROWVERSION         NOT NULL,
    [deleted]   BIT                DEFAULT ((0)) NULL,
    [year]      NVARCHAR (MAX)     NULL,
    [pagado]    BIT                NULL,
    [id_socio]  NVARCHAR (255)     NOT NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC),
    FOREIGN KEY ([id_socio]) REFERENCES [dbo].[Socio] ([id])
);
```

7.1.2 Mobile App

El api que se ha configurado para el servicio de backend se puede encontrar en el ANEXOII API.

7.2 Aplicación Android

Para desarrollar la aplicación AgentM para los dispositivos, además de utilizar el SDK de Android, se ha utilizado el SDK Azure Mobile App, que nos permite autentificarnos y conectarnos al Mobile App, y el SDK Azure Storage, para la gestión de los *blobs*.

En la organización del proyecto se han utilizado un conjunto de carpetas.

- El *package* **manifest** contiene el fichero *manifest* de la aplicación.
- En la carpeta **com.example.agent** contiene los archivos Activity.java.
- Dentro de **adapters** encontramos los adaptadores de las vistas para crear las listas personalizadas.
- En **model** hay las clases del dominio, estas clases contienen los mismos atributos que las tablas de la base de datos, ya que el SDK de Mobile App también las utilizará como veremos más adelante.
- **Pattern**, contiene las clases factoría, que instancian los componentes de los SDK de Azure.
- En **androidTest**, encontramos los ficheros de *testing* de AgentM.
- En la carpeta **fonts** encontramos las fuentes personalizadas de la aplicación.
- **Drawable** contiene todas las imágenes de la aplicación.
- La carpeta ***layout*** contiene todas las vistas en xml de AgentM.
- En la carpeta **menu** se agrupan todos los xml que definen las vistas de los *toolbar* y *drawer*.
- En **values** hay los ficheros de idioma, color y estilos.

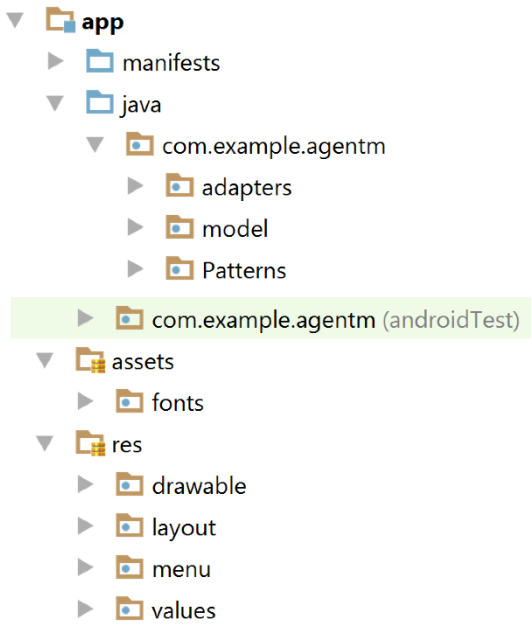


Ilustración 58.Packages Android.

7.2.1 SDK Mobile App

Es necesario inicializar Mobile App para conectarse con el servidor. Para solo tener que configurar una vez este servicio, se ha creado la clase FactoryClientMobile, que será una instancia que contendrá MobileServiceClient.

Al iniciar la aplicación se configura la conexión con MobileApp, donde ***URLMobileApp*** es la url del servicio MobileApp.

Una vez creada la conexión se procederá con la autentificación, donde ***stringAuth*** es una cadena da conexión y ***REQUEST_CODE*** es el código de retorno de onActivyResult.

Para obtener el id y el token la clase MobileServiceClient ofrece las siguientes funciones.

```
login("windowsazureactivedirectory", stringAuth, REQUEST_CODE);
```

```
getCurrentUser().getUserId();
getCurrentUser().getUserToken()
```

En el siguiente código se explica cómo se obtiene un MobileServiceTable para hacer peticiones a la tabla Socio de la base de datos.

```
private MobileServiceClient mClient;
private MobileServiceTable<Socio> mSocioTable;
mClient = FactoryClientMobile.getInstance().getClient();
mSocioTable = mClient.getTable(Socio.class);
```

Una vez tenemos la instancia de MobileServiceTable, podemos realizar peticiones de tipo GET, POST, PATCH y DELETE, un ejemplo de petición GET sería el siguiente:

```
List<Socio> socios = mSocioTable
                     .orderBy("nombre", QueryOrder.Ascending)
                     .execute().get();
```

Hay más información detallada en la documentación oficial de Microsoft Azure [24].

7.2.2 SDK Azure Storage

El siguiente código ilustra cómo se configura la conexión con la cuenta de almacenamiento de Azure. Necesitamos saber que ***storageConnectionString*** es la cadena de conexión a la cuenta y ***storageContainer*** contiene el nombre del contenedor al que queremos acceder.

```
CloudStorageAccount storageAccount =
CloudStorageAccount.parse(storageConnectionString);

CloudBlobClient blobClient =
storageAccount.createCloudBlobClient();

CloudBlobContainer container =
blobClient.getContainerReference(storageContainer);
```

Posteriormente podemos crear u obtener los blobs.

```
CloudBlockBlob blob =
container.getBlockBlobReference("example.jpg");
```

Y actualizarlos con las fotografías.

```
blob.upload(new FileInputStream(source), source.length());
```

8 Validación

En este apartado veremos las pruebas que se han realizado a AgentM.

8.1 Prueba de carga

Comprobación del requisito de latencia y velocidad; se han realizado dos pruebas de carga en nuestro servidor de Azure.

La primera carga tuvo una duración de 1 minuto, con 10 usuarios y se realizó desde Europa. Los resultados nos proporcionaron 4260 peticiones con un tiempo de respuesta medio de 0.04 segundos.

La segunda prueba también duró 1 minuto, esta vez con 250 usuarios y se realizó desde estados unidos. Los resultados que obtuvimos fueron 46223 peticiones resueltas en una media de tiempo de 0.46 segundos.

Dado que esta aplicación se utilizará mayormente en Barcelona y no tendrá más de 10 usuarios activos a la vez, vemos que tenemos muy buenos tiempos de respuesta.

Además, los servicios de Mobile App, aseguran una disponibilidad del 99’95% [25]

8.2 Pruebas unitarias

Para comprobar que las funcionalidades del backend con la aplicación, se han realizado pruebas unitarias con JUnit.

Tememos un conjunto de archivos con un conjunto de pruebas unitarias ordenadas, en las cuales se crea un nuevo elemento y se modifica de diferentes maneras y aseguurar que todo funciona correctamente.

En la gestión de socios, se ha creado un conjunto de test ordenados que realizan las siguientes pruebas:

- CreateSocio: Crea un socio con unos datos que no están en la base de datos.
- GetSocio: Obtiene los socios y comprueba que está el socio anterior.
- UpdateSocio: Se modifican los datos del socio y se comprueba que la modificación sea correcta.
- GetSocioActividades: Se añaden actividades al socio y se comprueba que aparezcan.
- BuscarSocioPorNIF: Se buscan los socios con el parámetro NIF del socio creado y se comprueba que aparezca.
- BuscarSocioPorNombre: Se buscan los socios con el parámetro nombre del socio creado y se comprueba que aparezca
- BuscarSocioPorApellido: Se buscan los socios con el parámetro apellido del socio creado y se comprueba que aparezca.
- DeleteSocio: Se elimina el socio, se piden los socios y se comprueba que la lista no contenga el socio eliminado.

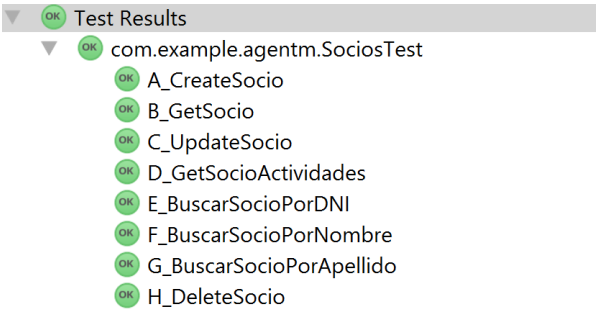


Ilustración 59.JUnit Socios.

En la validación de los registros se han realizado los siguientes test unitarios.

- CrearRegistro: Crea un registro nuevo en la base de datos.
- BuscarRegisto: Se piden todos los registros, y se comprueba que esté el registro que hemos creado con toda la información correcta.
- ModificarRegistro: Modificamos un registro y comprobamos que la información se modifica.
- PedirRegistroFecha: Pedimos los registros en la fecha del registro creado y comprobamos que todos están en esta fecha y se encuentra nuestro registro.
- EliminarRegistro: Eliminamos el registro y comprobamos que se elimina del listado de registros.
- PedirRegistrosNoPagados: Pedimos todos los registros no pagados y comprobamos que ninguno esta pagado.

Ilustración 60.JUnit Registros.

Para validar las cuotas anuales de los socios se han realizado las siguientes pruebas.

- CrearSocio: Creamos un socio nuevo.
- GetCuotaSocio: Obtenemos la cuota del año actual del socio y comprobamos que no está pagada.
- PagarCuota: Marcamos como pagada la cuota del socio y comprobamos que los cambios se realizan.
- DeshacerPago: Eliminamos el pago de la cuota y comprobamos que los cambios se realizan.

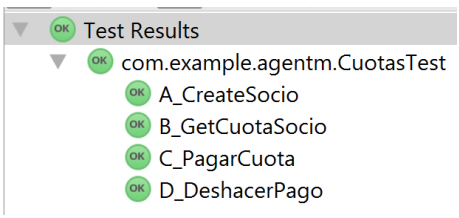


Ilustración 61.JUnit Cuotas.

Para comprobar las funcionalidades de los eventos, se han creado las siguientes pruebas unitarias.

- CrearEvento: Crear un Evento nuevo y comprobar que se crea con los valores indicados.
- ModificarEvento: Modificar el evento y comprobar que los cambios se guardan.
- SetSocio: Añadimos un socio al evento y comprobamos que se añada.
- PagarActividadSocio: Marcamos como pagada la actividad del socio y ver que se realizan los cambios.
- DeshacerPagoActividad: Deshacemos el pago de la actividad del socio y comprobamos que se guardan los cambios.
- QuitarSocio: Quitamos el socio de la actividad y comprobamos que la actividad no tiene socios.
- EliminarEvento: Eliminamos el evento y comprobamos que el evento no está en el listado de eventos.

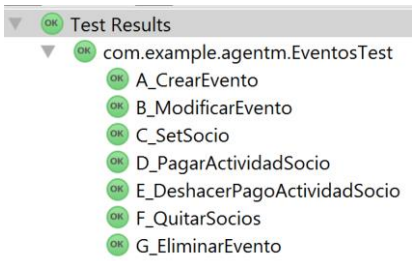


Ilustración 62.Junit Evento.

8.3 Pruebas manuales

Hay bastantes requisitos que se cumplen gracias a restricciones de la interfaz, como no añadir fechas incorrectas, campos vacíos, no poder eliminar socios de eventos que ya han pagado, etc.

La aplicación ha sido testeada probando todos los menús y funcionalidades, intentado introducir datos incorrectos o realizando acciones inesperadas. Comprobando que la aplicación funciona como debe y no tiene ningún comportamiento inesperado.

Además, gracias a estas pruebas se acabaron de validar todos los requisitos funcionales. Para los no funcionales se hicieron pruebas de rendimiento realizando peticiones y viendo que los tiempos eran inferiores a 3 segundos por petición, además, las personas que utilizaron la aplicación se adaptaron rápidamente a su uso y la opinión del diseño siempre fue un cinco sobre cinco.

9 Sostenibilidad

9.1 Dimensión económica

Para ver si el proyecto es sostenible a nivel económico, lo primero es evaluar los costes tanto de personas como de materiales, en el apartado de planificación, descrito anteriormente, se explica en detalle todos los costes de todos los recursos necesarios, estableciendo medidas para controlar imprevistos y poder contener cualquier contratiempo.

El coste total del proyecto se ajusta a la realidad, por lo tanto, cualquiera que quisiera adquirir todos los recursos descritos en el proyecto, tendría que asumir los mismos gastos.

El proyecto cuenta como recurso únicamente con una persona, por lo tanto, se podría recortar el tiempo del proyecto creando un grupo de personas encargadas de realizarlo, pero no se abaratarían costes.

Los servidores de Azure producen un gasto mensual mientras estén activos los servidores, pero la asociación tiene un acuerdo con Microsoft y el programa de estudiantes de Microsoft, con el que reciben bono mensual gratis para poder utilizar los servicios de Azure, por lo tanto, económicamente, es más barato que tener que comprar un ordenador e implementar un servidor, además de tener un espacio dedicado para mantenerlo con conexión a internet, sin hablar del tiempo dedicado a realizar copias de seguridad y arreglar averías del servidor.

Las tareas del proyecto tienen un tiempo asignado razonable a su importancia, se ha intentado utilizar tecnología existente para poder aplicarla al proyecto y así poder reducir tiempos y por lo tanto costes. Un ejemplo muy claro es el de utilizar Azure como *backend*, que nos permite tener un servidor funcional en mucho menos tiempo que si tenemos que instalar, configurar e implementar uno desde cero.

La parte de *backend* tiene prevista ampliarse para poder realizar, en un futuro, una herramienta para los socios de la asociación y poder reutilizar información, como, por ejemplo, el calendario y las actividades que ha realizado cada socio.

9.2 Dimensión social

La necesidad de realizar este proyecto, como ya se ha explicado en anteriores apartados, nace de la necesidad de la junta de la asociación de disponer de una herramienta que se pudiera utilizar en cualquier lugar y momento. Los requisitos del proyecto se han tomado con los propios miembros de la junta, por lo tanto, cuando el programa esté en funcionamiento mejorará la experiencia de los miembros de la junta, ya que podrán realizar las tareas que les correspondan sin tener que estar sentados delante de un ordenador.

9.3 Dimensión ambiental

El proyecto al tratarse de un sistema software, no tendrá casi ninguna repercusión en el medio ambiente, solo en el proceso de creación y en el consumo de los servidores, pero ahora veremos que serán una repercusión inapreciable.

Para realizar el proyecto, se ha utilizado un ordenador y un móvil, que ya habían sido adquiridos y que se utilizaban y se utilizarán para otros fines, a parte de la realización del proyecto. Por lo tanto, no se ha adquirido ningún componente de fabricación que pueda perjudicar al medio ambiente. En la realización solo se tendría en cuenta el gasto energético, que, al utilizar aparatos de bajo consumo, representan un impacto despreciable.

El servidor, está en las instalaciones de Azure, y es una instancia compartida, lo que quiere decir que mientras no se utilice el servidor, se utilizará para otras tareas. A nivel medioambiental, es mucho mejor esta opción, ja que la alternativa seria tener un servidor propio y dedicado sin los servicios de Azure y tendríamos consumo eléctrico en momentos que el servidor no se esté utilizando.

9.4 Matriz de sostenibilidad

	Económica	Social	Ambiental
Planificación	Económicamente viable	Mejora en la experiencia del usuario	Sin repercusión medioambiental
Puntuación	8.5	10	9.5

Tabla 11.Matriz de sostenibilidad.

10 Conclusiones

10.1 Objetivos conseguidos

Una vez finalizado el trabajo, se ha logrado cumplir el objetivo principal de este proyecto: se ha creado una aplicación móvil con almacenamiento online que ayude a la Junta de la asociación a organizar todas sus tareas.

Al finalizar el desarrollo de la aplicación, a pesar de que ha habido un retardo en la finalización del proyecto, se ha conseguido implementar, validar y documentar todos los requisitos definidos.

10.2 Trabajo futuro

Dado que la asociación es una asociación nueva, sin experiencia previa y aun no tiene socios para poder utilizar el 100% del potencial de la aplicación. En estos momentos no podemos decir si la aplicación ha sido todo un éxito o la aplicación no le ha servido a la asociación.

Cuando la asociación tenga socios y realice eventos, la Junta comenzará a utilizar la aplicación e irá viendo que funcionalidades les va bien, cuales no les hacía falta realmente, que funcionalidades nuevas necesitan o cuales tienen que modificar.

Por este motivo AgentM es la base de una aplicación que posiblemente irá evolucionando en el futuro.

10.3 Reflexión personal

Realizar esta aplicación ha sido una experiencia con un balance positivo, ha habido partes que han sido más divertidas que otras, en general me ha gustado mucho poder realizar este proyecto, ya que he podido ayudar a compañeros y he aprendido nuevos conocimientos que me serán útiles en el futuro.

Lo que más me ha gustado ha sido poder realizar la toma de requisitos, ya que requería tener buena comunicación con los encargados, y a pesar de que estuvimos bastantes horas, al final conseguí que se definieran todos.

Otro punto muy interesante e innovador para mí, ha sido poder utilizar algunos de los servicios de Azure, ya que no lo había utilizado nunca y ha estado muy bien poder aprender cómo funcionan.

Lo que menos me ha gustado ha sido tener que realizar la documentación, porque se me ha hecho una tarea muy pesada.

Al final, la mejor satisfacción ha sido poder finalizar un aplicación para móvil (AgentM) que se utiliza.

11 Referencias

[1] Definición de asociación [Online] [Ultima consulta: 22/05/2017]:

http://web.gencat.cat/ca/tramits/tramits-temes/Constitucio-duna-associacio

[2] BOE Num.73/2002 [Online] [Ultima consulta: 22/05/2017]:

http://www.boe.es/boe/dias/2002/03/26/pdfs/A11981-11991.pdf

[3] Formulario del acta fundacional [Online] [Ultima consulta: 22/05/2017]:

http://justicia.gencat.cat/web/.content/documents/formularis/associacions/acta_fund_assoc_cast.pdf

[4] Modelo de estatutos [Online] [Ultima consulta: 22/05/2017]:

http://justicia.gencat.cat/web/.content/documents/formularis/associacions/model_estatuts_assoc_general_cast.doc

[5] Microsoft Student Partner [Online] [Ultima consult: 14/03/2017]:

https://msdn.microsoft.com/es-es/microsoftstudentpartners.aspx

[6] Idesoft, software para asociaciones. [Online] [Ultima consulta: 16/03/2017]:

https://www.idesoft.es/software-para-asociaciones/

[7] MDGsoft, programa para la gestión de socios. [Online] [Ultima consulta: 16/03/2017]: http://www.mdgsoft.com/programas/mdg-socios.htm

[8] Mis Finanzas. [Online] [Ultima consulta: 16/05/2017]

https://play.google.com/store/apps/details?id=com.sevencsolutions.myfinances&hl=es

[9] Meetup. [Online] [Ultima consulta: 16/05/2017]

https://play.google.com/store/apps/details?id=com.meetup&hl=es

[10] Estudio de remuneración Michael Page del 2016 [Online] [Ultima consulta: 16/03/2017]:
http://www.michaelpage.es/sites/michaelpage.es/files/tecnologia2016.pdf

[11] Calculadora de precios Azure [Online] [Ultima consulta: 16/03/2017]:

https://azure.microsoft.com/es-es/pricing/calculator

[12] Scrum, información sobre el proceso y la práctica. [Online] [Ultima consulta: 16/03/2017]:

https://proyectosagiles.org/que-es-scrum/

[13] Metodología clásica en Cascada. [Online] [Ultima consulta: 28/04/2017]:

https://es.scribd.com/doc/35015019/Metodologia-en-Cascada/

[14] Tarjeta de historias de usuario [Online] [Ultima consulta: 19/05/2017]:

http://ronjeffries.com/xprog/articles/expcardconversationconfirmation/

[15] Charles Bradley [Online] [Ultima consulta: 22/05/2017]:

https://www.scrum.org/charles-bradley

[16] Pagina web de Charles Bradley [Online] [Ultima consulta: 24/05/2017]:

http://www.scrumcrazy.com/User+Story+Basics+-+What+is+a+User+Story%3F

[17] *Planning Poker* [Online] [Ultima consulta: 28/05/2017]:

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

[18] Mobile App [Online] [Ultima consulta: 02/06/2017]:

https://docs.microsoft.com/es-es/azure/app-service-mobile/app-service-mobile-value-prop

[19] Base de datos SQL [Online] [Ultima consulta: 02/06/2017]:

https://azure.microsoft.com/es-es/services/sql-database/

[20] Azure Active Directory [Online] [Ultima consulta: 02/06/2017]:

https://docs.microsoft.com/es-es/azure/active-directory/active-directory-whatis/

[21] Storage Acount - blob [Online] [Ultima consulta: 02/06/2017]:

https://docs.microsoft.com/es-es/azure/storage/storage-dotnet-how-to-use-blobs/

[22] OAuth 2.0 [Online] [Ultima consulta: 02/06/2017]:

https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2/

[23] Iconos de Android [Online] [Ultima consulta: 02/06/2017]:

https://material.io/icons/

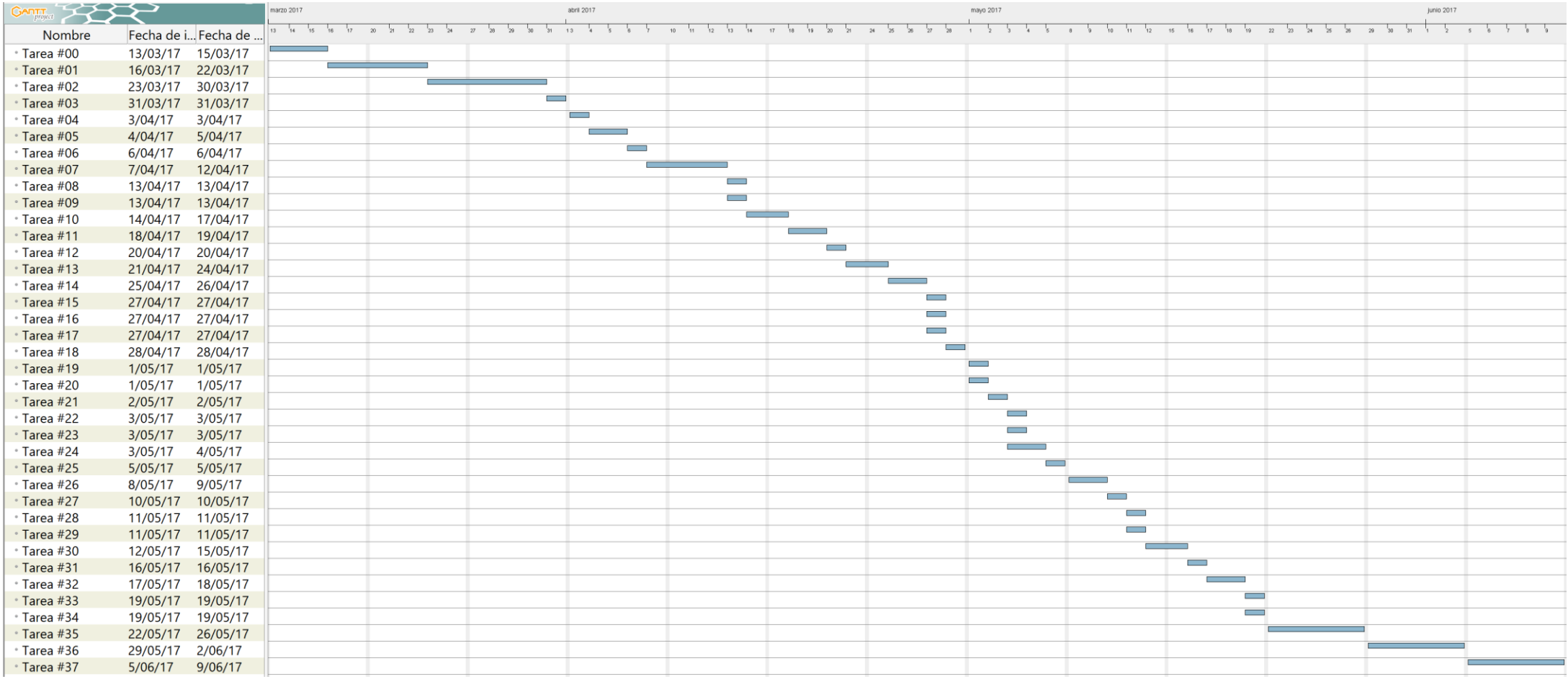
[24] SDK Android Azure Mobile App[Online] [Ultima consulta: 05/06/2017]:

https://docs.microsoft.com/es-es/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library

[24] Contrato App Service[Online] [Ultima consulta: 05/06/2017]:

https://azure.microsoft.com/es-es/support/legal/sla/app-service/v1_4/

Anexo I Diagrama de Gantt



Nombre	Fecha de i...	Fecha de ...
Tarea #00	13/03/17	15/03/17
Tarea #01	16/03/17	22/03/17
Tarea #02	23/03/17	30/03/17
Tarea #03	31/03/17	31/03/17
Tarea #04	3/04/17	3/04/17
Tarea #05	4/04/17	5/04/17
Tarea #06	6/04/17	6/04/17
Tarea #07	7/04/17	12/04/17
Tarea #08	13/04/17	13/04/17
Tarea #09	13/04/17	13/04/17
Tarea #10	14/04/17	17/04/17
Tarea #11	18/04/17	19/04/17
Tarea #12	20/04/17	20/04/17
Tarea #13	21/04/17	24/04/17
Tarea #14	25/04/17	26/04/17
Tarea #15	27/04/17	27/04/17
Tarea #16	27/04/17	27/04/17
Tarea #17	27/04/17	27/04/17
Tarea #18	28/04/17	28/04/17
Tarea #19	1/05/17	1/05/17
Tarea #20	1/05/17	1/05/17
Tarea #21	2/05/17	2/05/17
Tarea #22	3/05/17	3/05/17
Tarea #23	3/05/17	3/05/17
Tarea #24	3/05/17	4/05/17
Tarea #25	5/05/17	5/05/17
Tarea #26	8/05/17	9/05/17
Tarea #27	10/05/17	10/05/17
Tarea #28	11/05/17	11/05/17
Tarea #29	11/05/17	11/05/17
Tarea #30	12/05/17	15/05/17
Tarea #31	16/05/17	16/05/17
Tarea #32	17/05/17	18/05/17
Tarea #33	19/05/17	19/05/17
Tarea #34	19/05/17	19/05/17
Tarea #35	22/05/17	26/05/17
Tarea #36	29/05/17	2/06/17
Tarea #37	5/06/17	9/06/17

Tarea	Descripción	Horas(h)
TAREA #00	**Toma de requisitos de la aplicación.**	**20**
TAREA #01	**Definición de las historias de usuario.**	**40**
TAREA #02	**Formarse en desarrollo con Azure.**	**50**
TAREA #03	**Configuración del entorno de trabajo y de Azure.**	**8**
TAREA #04	**Definición y creación de la base de datos.**	**4**
TAREA #05	**Conexión de la base de datos con la aplicación.**	**16**
TAREA #06	**Crear un socio.**	**8**
TAREA #07	**Listar los socios.**	**32**
TAREA #08	**Modificar un socio.**	**6**
TAREA #09	**Eliminar un socio.**	**2**
TAREA #10	**Ver un socio.**	**8**
TAREA #11	**Buscador de socios por nombre o DNI para el listado de socios.**	**16**
TAREA #12	**Exportador de socios a formato EXCEL.**	**10**
TAREA #13	**Crear la vista de las cuotas anuales (listado de socios conforme si han pagado o no).**	**16**
TAREA #14	**Añadir pago de la cuota de un usuario.**	**16**
TAREA #15	**Modificar el año de visualización de pagos.**	**2**
TAREA #16	**Filtrar solo por impagados.**	**2**
TAREA #17	**Buscador de socios por nombre o DNI para el listado de cuotas.**	**2**
TAREA #18	**Crear un nuevo registro de cuentas.**	**8**
TAREA #19	**Listar el libro de cuentas con los registros.**	**6**
TAREA #20	**Mostrar el saldo de total.**	**2**
TAREA #21	**Ver un registro.**	**8**
TAREA #22	**Modificar un registro.**	**4**
TAREA #23	**Eliminar un registro.**	**2**
TAREA #24	**Exportar el libro de cuentas a formato EXCEL.**	**10**
TAREA #25	**Crear un nuevo evento.**	**8**
TAREA #26	**Listar los eventos.**	**16**
TAREA #27	**Ver un evento.**	**8**
TAREA #28	**Modificar un evento.**	**6**
TAREA #29	**Eliminar un evento.**	**2**
TAREA #30	**Añadir socios al evento.**	**16**
TAREA #31	**Confirmar pago de los usuarios al evento.**	**8**
TAREA #32	**Ver los eventos en un calendario.**	**16**
TAREA #33	**Crear menú de redacción de asunto y contenido email en la aplicación.**	**3**
TAREA #34	**Enviar el email**	**2**
TAREA #35	**Testear y confirmar que la aplicación está finalizada.**	**40**
TAREA #36	**Finalizar y revisar la memoria del trabajo.**	**40**
TAREA #37	**Preparar la presentación.**	**40**

Anexo II API

La API del servidor es la siguiente:

HOST:	https://agentm.azurewebsites.net/tables
VERSION:	ZUMO-API-VERSION 2.0.0

ACTIVIDAD

GET	/actividad	Obtener las actividades

- **Respuesta: json**


```
[
  {
    "id": "163893a4-1f4f-472e-9bd3-432388581445",
    "createdAt": "2017-05-12T03:08:16.519Z",
    "updatedAt": "2017-05-14T18:20:03.455Z",
    "version": "AAAAAAAAEFA=",
    "deleted": false,
    "nombre": "Cursos Xamarin",
    "descripcion": "Introducción al curso de Android",
    "fecha": "2017-05-18T19:52:00.000Z",
    "precio": 10,
    "lugar": "Aula A5S102",
    "ubicacion": "Campus Nord"
  },
  {
    "id": "163893a4-1f4f-472e-9bd3-432388581445",
    "createdAt": "2017-05-12T13:24:57.758Z",
    "updatedAt": "2017-05-17T17:14:45.077Z",
    "version": "AAAAAAAAEIw=",
    "deleted": false,
    "nombre": "Reunión Junt",
    "descripcion":    "Temas    a    tratar:\n\n*Horarios    de
xamarin*\n*Realización curso*",
    "fecha": "2017-05-17T20:30:00.000Z",
    "precio": 0,
    "lugar": "Sala 4, planta 2",
    "ubicacion": "Biblioteca campus nord"
  }
]
```

GET	/actividad/{ActividadId}	Obtener la actividad identificada por la id

- **Parámetro ActividadId: Id de la actividad a devolver.**
- **Respuesta: json**

```
{
    "id": "f83d3d62-e2ac-47e1-9ba5-e6ee35a99e16",
    "createdAt": "2017-05-12T03:08:16.519Z",
    "updatedAt": "2017-05-14T18:20:03.455Z",
    "version": "AAAAAAAAEFA=",
    "deleted": false,
    "nombre": "Cursos Xamarin",
    "descripcion": "Introducción al curso de Android",
    "fecha": "2017-05-18T19:52:00.000Z",
    "precio": 10,
    "lugar": "Aula A5S102",
    "ubicacion": "Campus Nord"
}
```

POST	/actividad	Insertar una nueva actividad

- **Cuerpo: json**

```
{
    "nombre": "Cursos Xamarin",
    "descripcion": "Introducción al curso de Android",
    "fecha": "2017-05-18T19:52:00.000Z",
    "precio": 10,
    "lugar": "Aula A5S102",
    "ubicacion": "Campus Nord"
 }
```

- **Respuesta: json**

```
{
    "id": "f83d3d62-e2ac-47e1-9ba5-e6ee35a99e16",
    "createdAt": "2017-05-12T03:08:16.519Z",
    "updatedAt": "2017-05-14T18:20:03.455Z",
    "version": "AAAAAAAAEFA=",
    "deleted": false,
    "nombre": "Cursos Xamarin",
    "descripcion": "Introducción al curso de android",
    "fecha": "2017-05-18T19:52:00.000Z",
    "precio": 10,
    "lugar": "Aula A5S102",
    "ubicacion": "Campus Nord"
}
```

PATCH /actividad/{ActividadId} Actualiza una actividad por id

- **Parámetro ActividadId: Id de la actividad a actualizar.**
- **Cuerpo: json**

```
{
    "id": "f83d3d62-e2ac-47e1-9ba5-e6ee35a99e16",
    "nombre": "Curso Xamarin",
    "descripcion": "Introducción al curso de android",
    "fecha": "2017-05-18T19:52:00.000Z",
    "precio": 10,
    "lugar": "Aula A5S102",
    "ubicacion": "Campus Nord"
 }
```

- **Respuesta: json**

```
{
    "id": "f83d3d62-e2ac-47e1-9ba5-e6ee35a99e16",
    "createdAt": "2017-05-12T03:08:16.519Z",
    "updatedAt": "2017-05-14T18:20:03.455Z",
    "version": "AAAAAAAAEFA=",
    "deleted": false,
    "nombre": "Cursos Xamarin",
    "descripcion": "Introducción al curso de android",
    "fecha": "2017-05-18T19:52:00.000Z",
    "precio": 10,
    "lugar": "Aula A5S102",
    "ubicacion": "Campus Nord"
}
```

DELETE /actividad/{ActividadId} Eliminar una actividad por id

- **Parámetro ActividadId: Id de la actividad a eliminar.**

REGISTRO

GET /registro Obtener los registros

- **Respuesta: json**


```
[
  {
    "id": "f83d3d62-e2ac-47e1-9ba5-e6ee35a99e16",
    "createdAt": "2017-05-10T02:24:34.302Z",
    "updatedAt": "2017-05-10T02:24:34.302Z",
    "version": "AAAAAAAACHs=",
    "deleted": false,
    "titulo": "Material Oficina",
    "descripcion": null,
    "imagen": null,
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": 30,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
  },
  {
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "titulo": "Material",
    "descripcion": "Folios",
    "imagen": "05062017185542",
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": -22.5,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
  }
]
```

GET /registro/{RegistroId} Obtener el registro identificado por la id

- **Parámetro RegistroId: Id del registro a devolver.**
- **Respuesta: json**


```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "titulo": "Material",
    "descripcion": "Folios",
    "imagen": "05062017185542",
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": -22.5,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
}
```


POST /registro Insertar un nuevo registro

- **Cuerpo: json**


```
{
    "titulo": "Material",
    "descripcion": "Folios",
    "imagen": "05062017185542",
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": -22.5,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
 }
```


- **Respuesta: json**


```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "titulo": "Material",
    "descripcion": "Folios",
    "imagen": "05062017185542",
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": -22.5,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
}
```

PATCH /registro/{RegistroId} Actualiza un registro por id

- **Parámetro RegistroId: Id del registro a actualizar.**
- **Cuerpo: json**


```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "titulo": "Material",
    "descripcion": "Folios",
    "imagen": "05062017185542",
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": -22.5,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
 }
```


- **Respuesta: json**


```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "titulo": "Material",
    "descripcion": "Folios",
    "imagen": "05062017185542",
    "fecha": "2017-05-09T22:00:00.000Z",
    "precio": -22.5,
    "modifiedBy": "sid:67b95238de5730daf3ec99c54013ec51"
}
```


DELETE /registro/{RegistroId} Eliminar un registro por id

- **Parámetro SocioId: Id del socio a eliminar.**

SOCIO

GET /socio Obtener los socios

- **Respuesta: json**


```
[
  {
    "id": "b2225613-8e9d-45b7-8e83-a54535548a8c",
    "createdAt": "2017-05-13T02:14:20.411Z",
    "updatedAt": "2017-05-13T02:14:20.411Z",
    "version": "AAAAAAAADxs=",
    "deleted": false,
    "nombre": "Maria",
    "dni": "12121212M",
    "telefono": 698566985,
    "email": "maria@gmail.com",
    "esUPC": false,
    "apellidos": "Herrera Pardo"
  },
  {
    "id": "b237be7b-c35d-43d5-ac7f-b2ef307342ec",
    "createdAt": "2017-05-14T02:46:49.386Z",
    "updatedAt": "2017-05-14T02:46:49.386Z",
    "version": "AAAAAAAAD44=",
    "deleted": false,
    "nombre": "Juan",
    "dni": "12121212M",
    "telefono": 698563985,
    "email": "juan@hotmail.com",
    "esUPC": true,
    "apellidos": "Palacios Ortega"
  }
]
```

GET /socio/{SocioId} Obtener el socio identificado por la id

- **Parámetro SocioId: Id del socio a devolver.**
- **Respuesta: json**


```
{
    "id": "b237be7b-c35d-43d5-ac7f-b2ef307342ec",
    "createdAt": "2017-05-14T02:46:49.386Z",
    "updatedAt": "2017-05-14T02:46:49.386Z",
    "version": "AAAAAAAAD44=",
    "deleted": false,
    "nombre": "Juan",
    "dni": "12121212M",
    "telefono": 698563985,
    "email": "juan@hotmail.com",
    "esUPC": true,
    "apellidos": "Palacios Ortega"
  }
```


POST /socio Insertar un nuevo socio

- **Cuerpo: json**


```
{
    "nombre": "Juan",
    "dni": "12121212M",
    "telefono": 698563985,
    "email": "juan@hotmail.com",
    "esUPC": true,
    "apellidos": "Palacios Ortega"
 }
```


- **Respuesta: json**


```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "nombre": "Juan",
    "dni": "12121212M",
    "telefono": 698563985,
    "email": "juan@hotmail.com",
    "esUPC": true,
    "apellidos": "Palacios Ortega"
}
```

PATCH /socio/{SocioId} **Actualiza un socio por id**

- **Parámetro SocioId: Id del socio a actualizar.**
- **Cuerpo: json**

```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "nombre": "Juan",
    "dni": "12121212M",
    "telefono": 698563985,
    "email": "juan@hotmail.com",
    "esUPC": true,
    "apellidos": "Palacios Ortega"
 }
```

- **Respuesta: json**

```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "nombre": "Juan",
    "dni": "12121212M",
    "telefono": 698563985,
    "email": "juan@hotmail.com",
    "esUPC": true,
    "apellidos": "Palacios Ortega"
}
```

DELETE /socio/{RegistroId} **Eliminar un registro por id**

- **Parámetro RegistroId: Id del registro a eliminar.**

SOCIOACTIVIDAD

GET /socioactividad Obtener los socioactividades

- **Respuesta: json**


```
[
  {
    "id": "ead6d2bd-c6be-428a-8744-aaf793fbbf05",
    "createdAt": "2017-05-13T11:52:23.104Z",
    "updatedAt": "2017-05-13T11:52:23.104Z",
    "version": "AAAAAAAAD0U=",
    "deleted": false,
    "pagado": true,
    "socio_id": "824b93ae-7364-48b9-9ded-1655a169953b",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
  },
  {
    "id": "331d06f9-91c1-44df-82c0-b1accc5788fc",
    "createdAt": "2017-05-13T12:50:05.295Z",
    "updatedAt": "2017-05-13T12:50:05.295Z",
    "version": "AAAAAAAAD0g=",
    "deleted": false,
    "pagado": true,
    "socio_id": "e83fa8af-1a14-47d1-b10f-fc69e00ff21e",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
  }
]
```


GET /socioactividad/{Id} Obtener el socioactividad por id

- **Parámetro Id: Id de la socioactividad a devolver.**
- **Respuesta: json**


```
{
    "id": "331d06f9-91c1-44df-82c0-b1accc5788fc",
    "createdAt": "2017-05-13T12:50:05.295Z",
    "updatedAt": "2017-05-13T12:50:05.295Z",
    "version": "AAAAAAAAD0g=",
    "deleted": false,
    "pagado": true,
    "socio_id": "e83fa8af-1a14-47d1-b10f-fc69e00ff21e",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
}
```

POST /socioactividad Insertar un nuevo socioactividad

- **Cuerpo: json**

```
{
    "pagado": true,
    "socio_id": "e83fa8af-1a14-47d1-b10f-fc69e00ff21e",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
 }
```

- **Respuesta: json**

```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "pagado": true,
    "socio_id": "e83fa8af-1a14-47d1-b10f-fc69e00ff21e",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
}
```

PATCH /socioactividad/{Id} Actualiza un socioactividad por id

- **Parámetro ID: Id del socioactividad a actualizar.**
- **Cuerpo: json**

```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "pagado": true,
    "socio_id": "e83fa8af-1a14-47d1-b10f-fc69e00ff21e",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
 }
```

- **Respuesta: json**

```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "createdAt": "2017-05-10T02:26:50.947Z",
    "updatedAt": "2017-05-10T02:26:50.947Z",
    "version": "AAAAAAAACH0=",
    "deleted": false,
    "pagado": true,
    "socio_id": "e83fa8af-1a14-47d1-b10f-fc69e00ff21e",
    "actividad_id": "163893a4-1f4f-472e-9bd3-432388581445"
}
```

DELETE /socioactividad/{Id} Eliminar un socioactividad por id

- **Parámetro Id: Id del socioactividad a eliminar.**

CUOTASSOCIO

GET /cuotassocio Obtener las cuotassocio

- **Respuesta: json**


```
[
  {
    "id": "668a6ff0-8d94-43b4-9f65-fed3cc664159",
    "createdAt": "2017-05-13T00:32:25.709Z",
    "updatedAt": "2017-05-14T03:14:56.530Z",
    "version": "AAAAAAAAEAQ=",
    "deleted": false,
    "year": "2017",
    "pagado": true,
    "id_socio": "824b93ae-7364-48b9-9ded-1655a169953b"
  },
  {
    "id": "83766467-c929-4901-ab3b-d7cd421fb812",
    "createdAt": "2017-05-13T00:40:41.699Z",
    "updatedAt": "2017-05-13T00:40:41.699Z",
    "version": "AAAAAAAADwU=",
    "deleted": false,
    "year": "2017",
    "pagado": false,
    "id_socio": "39cdcfdc-53ae-4b6b-9ef4-d70b9504cb26"
  }
]
```

GET /cuotassocio/{Id} Obtener el cuotassocio por id

- **Parámetro Id: Id de la cuotassocio a devolver.**
- **Respuesta: json**


```
{
    "id": "83766467-c929-4901-ab3b-d7cd421fb812",
    "createdAt": "2017-05-13T00:40:41.699Z",
    "updatedAt": "2017-05-13T00:40:41.699Z",
    "version": "AAAAAAAADwU=",
    "deleted": false,
    "year": "2017",
    "pagado": false,
    "id_socio": "39cdcfdc-53ae-4b6b-9ef4-d70b9504cb26"
}
```


POST /cuotassocio Insertar un nuevo cuotassocio

- **Cuerpo: json**


```
{
    "year": "2017",
    "pagado": false,
    "id_socio": "39cdcfdc-53ae-4b6b-9ef4-d70b9504cb26"
}
```


- **Respuesta: json**


```
{
    "id": "83766467-c929-4901-ab3b-d7cd421fb812",
    "createdAt": "2017-05-13T00:40:41.699Z",
    "updatedAt": "2017-05-13T00:40:41.699Z",
    "version": "AAAAAAAADwU=",
    "deleted": false,
    "year": "2017",
    "pagado": false,
    "id_socio": "39cdcfdc-53ae-4b6b-9ef4-d70b9504cb26"
}
```

PATCH /cuotassocio/{Id} Actualiza un cuotassocio por id

- **Parámetro ID: Id del cuotassocio a actualizar.**
- **Cuerpo: json**

```
{
    "id": "69c866bd-4a08-46cb-99ca-4c5040607d26",
    "year": "2017",
    "pagado": false,
    "id_socio": "39cdcfdc-53ae-4b6b-9ef4-d70b9504cb26"
 }
```

- **Respuesta: json**

```
{
    "id": "83766467-c929-4901-ab3b-d7cd421fb812",
    "createdAt": "2017-05-13T00:40:41.699Z",
    "updatedAt": "2017-05-13T00:40:41.699Z",
    "version": "AAAAAAAADwU=",
    "deleted": false,
    "year": "2017",
    "pagado": false,
    "id_socio": "39cdcfdc-53ae-4b6b-9ef4-d70b9504cb26"
}
```

DELETE /socioactividad/{Id} Eliminar un socioactividad por id

- **Parámetro Id: Id del socioactividad a eliminar.**