

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de una aplicación Android que unifique eventos y actividades en la UAM (J-Event)

Javier Fernández Riobos
Tutor: Alejandro Bellogín Kouki
Ponente: Iván Cantador Gutiérrez

JUNIO 2017

Desarrollo de una aplicación Android que unifique eventos y actividades en la UAM (J-Event)

AUTOR: Javier Fernández Riobos

TUTOR: Alejandro Bellogín Kouki

PONENTE: Iván Cantador Gutiérrez

**Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2017**

Resumen

La gestión de actividades que acontecen en una institución es de vital importancia a la hora de dar a conocer y promocionar los eventos llevados a cabo en ésta con el fin de que todo aquel que lo desee pueda participar. La solución actual reside en mostrar información relevante acerca del contenido de dichas actividades en lugares donde el público objetivo sea capaz de visualizarlo, entre los que se encuentran puntos de paso común o subsecciones de páginas web de la institución designadas a tal efecto.

Sin embargo, en la mayoría de los casos no se logra atraer a tanto público objetivo como se desearía, en parte debido a que los eventos no están disponibles de forma clara e integrada en un ámbito donde cualquiera podría encontrar información de su interés, sino que se encuentran aislados en pocos puntos de paso, o en páginas web que no son consultadas frecuentemente.

Es clara la intencionalidad de mejorar aspectos de este tipo en instituciones de ámbito global como empresas multinacionales o negocios donde el principal objetivo es la formación de empleados o la participación de éstos en actividades de claro fin lucrativo para la propia institución. Pero si nos centramos en el ámbito académico, estas posibilidades se reducen notablemente, hasta el punto de encontrar pocas alternativas intuitivas y de gran uso para el alumnado.

El objetivo principal de este Trabajo de Fin de Grado es el desarrollo de J-Event, una aplicación para dispositivos móviles centrada en la unificación de eventos que acontecen en las principales instituciones académicas. De esta forma, se proporciona al usuario una herramienta que proporciona intuitivamente todas aquellas actividades que puedan ser de su interés, no solo de carácter académico, sino también en aquellas de carácter extracurricular.

Las ventajas de J-Event con respecto a las demás aplicaciones del sector engloban desde una mayor sencillez a la hora de acceder a eventos que puedan ser de interés, como una mayor unificación de todas aquellas actividades relevantes propiciada por la capacidad que tienen los usuarios de subir sus propios eventos y categorizarlos, además de estar disponible para todo aquél que lo desee, aunque no sea integrante de la institución académica.

El desarrollo de esta aplicación se ha realizado por medio del lenguaje Android, basado en Java, tecnología recomendada para el desarrollo de aplicaciones en dispositivos móviles, junto con el gestor de bases de datos SQLite para integrar la información.

Palabras clave

Eventos académicos/extra-académicos, aplicaciones para dispositivos móviles, Android, SQLite, J-Event.

Abstract

Management of activities that take place in an institution is really important to announce and promote the events carried out in it so everyone could take part if they want to. The current solution, typically, involves showing relevant information of the activities' content in places where the target audiences are able to visualize it, for instance, common waypoints or subparts of institution websites designated for that aim.

However, in the majority of cases we cannot attract as much target audience as we would desire to, due to the fact that the events are not available in an understandable and integrated way in a scope where everyone could find information of their interest, but they are isolated in few waypoints, or in websites that are not frequently visited.

The intentionality of overcoming aspects of this category is obvious in global scope institutions like multinational companies or businesses where the main objective is employees training or their participation in activities for the institution profit. But if we focus ourselves in the academic context, these possibilities are reduced notably, to the point of finding few intuitive and useful alternatives.

The main objective for this Bachelor Thesis is the development of J-Event, an application for mobile devices focused on the events unification that takes place in most academic institutions. Thus, we provide the user a tool which offer intuitively all those activities that may be for his or her interest, not only in an academic context, but also those with extra-academic specifications.

The advantages of J-Event with regard to other industry applications encompass from a major simplicity when accessing events that may be for the user's interest, as a higher relevant activities unification propitiated by the capacity that users have to upload their own events and categorise them, in addition to be available for everyone that wants to use it, although they may not act as members of the academic institution.

The development of this application has been done through the Android language, based on Java, recommended technology for the mobile devices application development, along with the database manager SQLite to integrate the information.

Keywords

Academic, extra-academic events, mobile devices applications, Android, SQLite, J-Event.

Agradecimientos

Agradecer a gente de confianza que ha tenido contacto con la aplicación con el objetivo de mejorarla.

Al tutor que me ha prestado la ayuda que necesitaba en cada momento, y se ha mostrado siempre dispuesto a hacer un producto profesional y de calidad, sin perder los objetivos que pretendía.

A profesores de las numerosas disciplinas de las que se compone el proyecto, gracias a los cuales he aprendido a engranar todos los aspectos necesarios para la formalización de la idea que se pretendía al principio.

A mi compañera Patricia, por ayudarnos mutuamente y ser ese apoyo que se nota en esos momentos de agobio y desesperación.

A ti, mamá, por haber estado siempre orgullosa de mí y apoyándome, a ti Papá, por exigirme siempre superarme, y a ti Sara, por ser uno de los pilares sobre los que me sustentó.

A mis compañeros de la carrera, gracias a la cual me ha permitido conocer a bellísimas personas que me acompañarán el resto de mi vida.

Gracias al resto de familia y amigos, por apoyarme en esta aventura, desde principio y hasta el final, confiando en mí.

Y sobre todo a ti, Paula, por aguantar mis quejas y angustias y saber que puedo contar contigo para esto y mucho más. Va por ti, va por nosotros.

INDICE DE CONTENIDOS

INDICE DE FIGURAS	iii
1 Introducción.....	1
1.1 Motivación	1
1.2 Objetivos	1
1.3 Organización de la memoria	2
2 Estado del arte	5
2.1 Análisis de aplicaciones similares a un gestor de eventos académicos	5
2.1.1 Conclusión de aplicaciones similares a un gestor de eventos académicos ..	8
2.2 Análisis de tecnologías de desarrollo en Android.....	9
2.2.1 Otras tecnologías empleadas	12
2.2.2 Conclusión sobre el estudio de tecnologías de desarrollo en Android	15
3 Diseño.....	17
3.1 Ciclo de vida	17
3.2 Base de datos.....	18
3.2.1 Diagrama Entidad – Relación	18
3.2.2 Entidades de la base de datos	19
3.3 Diagrama de secuencia.....	20
4 Desarrollo	23
4.1 Lógica de la aplicación.....	24
4.1.1 <i>Bean</i> usuario (user)	24
4.1.2 <i>Beans</i> evento (event) y categoría (category).....	25
4.1.3 <i>Bean</i> comentario (commentary).....	25
4.1.4 Actividad de log in (LoginActivity).....	25
4.1.5 Actividad de registro (RegisterActivity).....	25
4.1.6 Preferencias del usuario (JEventPreferenceActivity)	26
4.1.7 Actividad inicial y lista de eventos (EventListActivity).....	26
4.1.8 Actividad de eventos (EventActivity).....	26
4.1.9 Barra de acción y menú.....	26
4.2 Modelo de datos	27
4.2.1 Base de datos local	27
4.2.2 Modelo de datos en el servidor	29
4.3 Vista	29
4.3.1 Pantallas de carga (splash_screen) y de Log in (activity_login).....	29
4.3.2 Pantalla de registro (activity_register)	29
4.3.3 Pantalla inicial y lista de eventos (fragment_event_list)	29
4.3.4 Pantalla de eventos (fragment_event)	30
4.3.5 Barra de acción y menú.....	30
5 Integración, pruebas y resultados	33
5.1 Pruebas funcionales.....	33
5.1.1 Pruebas unitarias	33
5.1.2 Pruebas de componentes	34
5.1.3 Pruebas de integración	34
5.1.4 Pruebas de sistema	35
5.2 Pruebas no funcionales.....	35
5.2.1 Pruebas de compatibilidad	35
5.2.2 Pruebas de usabilidad.....	36
5.2.3 Pruebas de escalabilidad	36

6 Conclusiones y trabajo futuro.....	37
6.1 Conclusiones	37
6.2 Trabajo futuro.....	37
Referencias	39
Glosario	41
Anexos.....	- 1 -
A Manual de usuario	- 1 -
B Análisis de requisitos.....	- 7 -
Requisitos funcionales.....	- 7 -
Requisitos no funcionales.....	- 8 -
C Maquetas de la aplicación.....	- 11 -
D Diseño gráfico de actividades y flujo entre ellas.....	- 13 -

INDICE DE FIGURAS

FIGURA 1: APLICACIÓN UCM EVENTOS	5
FIGURA 2: APLICACIÓN IUPA EVENTOS.....	7
FIGURA 3: COMPARATIVA ENTRE LOS DISTINTOS SISTEMAS OPERATIVOS PARA DISPOSITIVOS MÓVILES [3]	9
FIGURA 4: DISTINTAS VERSIONES DISPONIBLES DE ANDROID A FECHA DE 19 DE ENERO DE 2017 [25]	10
FIGURA 5: ENTORNO ANDROID STUDIO.....	10
FIGURA 6: ENTORNO ECLIPSE	11
FIGURA 7: ENTORNO BASIC4ANDROID.....	12
FIGURA 8: GESTOR DE BASES DE DATOS SQLITE.....	12
FIGURA 9: LOGO DE LA BIBLIOTECA VOLLEY	13
FIGURA 10: LOGO DE GOOGLE MAPS.....	14
FIGURA 11: LOGO Y ESLOGAN DE DROPWIZARD	14
FIGURA 12: CICLO DE VIDA EN CASCADA CON REALIMENTACIÓN.....	17
FIGURA 13: DIAGRAMA ENTIDAD RELACIÓN DE LA APLICACIÓN.....	18
FIGURA 14: DIAGRAMA DE SECUENCIA DE LA APLICACIÓN.....	21
FIGURA 15: ARQUITECTURA CLIENTE-SERVIDOR [9]	23
FIGURA 16: PATRÓN MODELO-VISTA-CONTROLADOR (MVC) [15].....	23

1 Introducción

1.1 Motivación

Hoy en día las universidades constituyen focos de actividad creciente si atendemos a los eventos que se desarrollan dentro y fuera de ellas [26]. Es por esto, que todo personal relacionado directa o indirectamente con cualquier universidad necesita acceder de forma rápida y sencilla al abanico de posibilidades que se le ofrecen cuando forma parte de dicha institución. Actualmente, a este tipo de eventos se accede mediante la página web de la facultad o universidad donde se desarrollan o a través de anuncios y panfletos situados en éstas (lo más frecuente).

Sin embargo, la gran parte de la información anteriormente mencionada queda desatendida por un sector mayoritario de la población estudiantil y de profesorado, conllevando a la ignorancia de éstos hacia actividades que podrían ser de su interés. Además, es frecuente observar información relativa a eventos relacionados con la propia carrera de la facultad donde se estudia, u otras actividades más genéricas en puntos de paso de la universidad, pero no información de otras facultades y entornos que perfectamente podrían servir de rédito hacia todo aquel que se preste a simplemente echarlas un vistazo.

Producto de todo esto, surge la necesidad de una aplicación intuitiva y completa que unifique dichas actividades de carácter académico y extracadémico que sirvan al alumnado y profesorado como herramienta focalizada en mostrar información relevante de todo aquello que acontece en la universidad elegida, dando pie a todo aquel que lo desee a poder formar parte de dichas actividades o eventos, o simplemente consultar dicha información con el fin de tener conocimiento de ella.

1.2 Objetivos

El objetivo fundamental de la aplicación es hacer accesible toda aquella información de carácter académico o extracadémico disponible a lo largo de todo el campus universitario que pueda ser de relevancia para cualquier estudiante o profesor, dando pie al fomento de la concienciación de todos aquellos eventos y actividades desarrollados en la universidad.

El hecho de englobar y unificar toda la información anteriormente citada conllevará una mayor facilidad para encontrar aquellos eventos a los que se desee acudir, o simplemente a tener constancia de todo aquello que ocurre en el ámbito universitario.

Así las principales funcionalidades que debe tener en cuenta la aplicación son las siguientes:

- **Gestión del acceso por parte de los usuarios:** todo aquel que desee acceder a la aplicación podrá hacerlo, sin embargo, para la inscripción de eventos será necesario introducir los datos académicos del usuario, con el objetivo de controlar y moderar los eventos académicos que se introduzcan.

- **Clasificación de la información:** los eventos y actividades se organizarán de acuerdo al ámbito (académico o extracadémico) o facultad al que pertenezcan, de forma que se pueda acceder a ellos de forma más intuitiva.
- **Gestión de la inscripción a un evento/actividad:** la aplicación dispondrá de información detallada de todas las actividades integradas, donde se clarifican el lugar y horario de realización, además de información de contacto con el fin de que sea más sencillo inscribirse en dicho evento.
- **Gestión de la incorporación/modificación de información:** así mismo, la aplicación facilita la inclusión o modificación de información por parte del usuario, referente a eventos o actividades que éstos deseen compartir con el objetivo de que estén disponibles para todo aquel que lo desee. Para realizar dicha funcionalidad, se requiere la introducción de datos académicos por parte del usuario (nombre del correo de estudiante). A continuación se enviará un correo a la dirección introducida para verificar que se ha realizado desde un usuario válido. Esta información, antes de ser integrada o modificada en la aplicación, será aprobada o rechazada por el moderador del sistema.
- **Gestión de la actualización de información:** la aplicación periódicamente comprobará la caducidad de todos los eventos y actividades, con el fin de actualizar la información de forma coherente.
- **Gestión de la interfaz de usuario:** la aplicación contará con una interfaz intuitiva y amigable donde se dispondrán los eventos clasificados de la forma anteriormente descrita, y tendrá en cuenta todas las funcionalidades aquí señaladas.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte**
En este capítulo se presenta un análisis exhaustivo acerca de diversas aplicaciones en entornos similares al pretendido con este trabajo, además de un análisis en detalle acerca de las tecnologías disponibles para el desarrollo de un producto como el que se pretende. Se valoran puntos a tener en cuenta, funcionalidades y herramientas que podrían encajar en la aplicación en desarrollo.
- **Diseño**
En este capítulo se presenta la fase de diseño del ciclo de vida de la aplicación correspondiente a este trabajo, en la que se muestran con detalle el ciclo de vida y metodología utilizados, el diagrama entidad relación de la base de datos y el diagrama de secuencia, además de explicaciones previas al desarrollo, que facilitan la consecución de éste.
- **Desarrollo**
En este capítulo se presenta la fase de desarrollo y codificación del ciclo de vida de la aplicación correspondiente a este trabajo, en la que se detalla la arquitectura y patrón utilizados para realizarla, además de la explicación de cada una de las partes que constituyen dicho patrón de forma exhaustiva.
- **Integración, pruebas y resultados**
En este capítulo se presenta la fase de pruebas del ciclo de vida de la aplicación correspondiente a este trabajo, en la que se realizan una serie de tentativas que

analizan las diversas partes que conforman el proyecto, y después la unión de dichas partes, extrayendo los resultados que arrojan.

- **Conclusiones y trabajo futuro**

En este último capítulo se exhibe una recapitulación de todas las partes que conforman el presente proyecto, además de las propuestas de mejora y optimización futuras que se llevarían a cabo en una fase posterior a la actual.

2 Estado del arte

2.1 Análisis de aplicaciones similares a un gestor de eventos académicos

Antes de comenzar el desarrollo de la aplicación se procedió a realizar un análisis exhaustivo de aplicaciones similares en entornos académicos, que sirvieron de base para concretar los elementos que podrían incluirse en la aplicación a desarrollar, además de algunos aspectos mejorables y de cambio que podrían dar lugar a partes de la aplicación innovadoras en el campo de las aplicaciones de eventos en ámbitos académicos. Así, se analizaron algunas de las siguientes aplicaciones:

- *Eventos UCM*



Figura 1: Aplicación UCM Eventos

Eventos UCM es una aplicación desarrollada exclusivamente para la Universidad Complutense de Madrid, que alberga los eventos y actividades que acontecen dentro de la institución, además de conocer y gestionar todo tipo de actividades organizadas por la universidad anteriormente mencionada [11].

Al entrar en dicha aplicación, lo primero que salta a la vista es la sencillez en la organización de la interfaz, basada en tres elementos principales, clasificados en pestañas: “próximos”, “mis eventos” y “avisos”, los cuales se describirán en detalle a continuación. Sin embargo, cabe destacar la escasez de imaginación a la hora de elaborar una interfaz más dinámica, que otorgue un poco más de color y realidad a la app, con tonos predominantes básicos como blanco y negro y monotonía en la caligrafía, este es, claramente, un aspecto mejorable dentro de la susodicha aplicación.

La pestaña “próximos” se asemeja a una aplicación de tipo agenda en la que se nos muestran las actividades que se realizan día a día, con un título identificativo breve y la dirección exacta de google maps, que otorga un nivel de detalle preciso y claro, donde el estudiante puede directamente saber el lugar donde se realiza de forma inequívoca. Si accedemos al evento en cuestión, se nos muestra una nueva ventana en la que aparece un

resumen de la actividad, nuevamente la ubicación precisa, la fecha y hora del evento (además de las fechas de inscripción, de inicio y de fin), el programa detallado si está disponible y los ponentes de la actividad. Además se nos da la oportunidad de inscribirnos al evento o de acceder a la web de éste, ambos si están disponibles. Sendas acciones te redireccionan a la página web de la universidad, donde se gestionará de forma directa la inscripción y se darán más detalles acerca del evento. Por último cabe destacar la posibilidad de compartir el evento a través de las aplicaciones sociales cotidianas, hecho que otorga a la aplicación una gran capacidad de extensión.

La pestaña “Mis eventos” a priori alberga todos aquellos eventos a los que el usuario pretende asistir (funcionalidad no implementada ya que se pide acceder por correo para poder interactuar), pero el objetivo está claro, seleccionar todos aquellos eventos de interés y organizarlos.

La pestaña “Avisos” lista todos aquellos eventos que se han ido integrando en la aplicación de forma consecutiva, dando un título descriptivo y una breve sinopsis acerca de la actividad en cuestión.

Como funcionalidad adicional, se puede observar (deslizándose hacia arriba en la pantalla), la capacidad de actualización de la aplicación al instante y un apartado de ajustes en el que se manejarán las opciones de notificación disponibles en la app, esto es, notificaciones generales de nuevos eventos y notificaciones de eventos en donde el usuario está inscrito.

En general, la aplicación cubre los aspectos esenciales que se buscan en este tipo de apps, y se conforma como una de las más completas del género, con funcionalidades específicas para el caso que se está tratando: una aplicación de eventos académicos. Quizás se echaría en falta (además de la interfaz más dinámica explicada con anterioridad) la capacidad del usuario para subir nuevos eventos, ya que en principio tan sólo podrían publicar actividades los desarrolladores o moderadores de la aplicación, y no los usuarios, además de un apartado de ajustes más extenso que diera más opciones de cambio como poder organizar los eventos en temáticas o similar a gusto del usuario, y por último una característica indispensable es que la aplicación es de libre acceso, cualquiera puede ver los eventos que acontecen en la universidad, dando una sensación de descontrol al no exigir un nombre de usuario y un correo institucional (ya que se usa el de google play). Además de esto se echan en falta los eventos pasados y comentarios que el usuario pudiera hacer dentro de cualquier actividad.

- *IUPa*



Figura 2: Aplicación iUPa Eventos

IUPa es una aplicación desarrollada por la universidad de Málaga que ofrece un amplio abanico de los eventos culturales y científicos organizados en las universidades públicas andaluzas (Sevilla, Málaga, Cádiz, Córdoba, Granada, Almería, Jaén y Huelva), organizadas en diversas categorías [14].

Al iniciar la aplicación, se nos muestra una pantalla en la que debemos de seleccionar la provincia a la que deseamos acceder y las categorías que nos interesen (mediante sendas listas desplegadas). Una vez introducidos los datos anteriormente mencionados, aparecerá un listado de eventos (nombre identificativo y fecha y hora exacta) ordenado por la proximidad horaria a la consulta realizada y un botón de búsqueda para localizar un evento en concreto.

Así, una vez obtenido el listado de eventos, nos aparecerán cinco iconos en la parte inferior de la página, cada uno conteniendo funcionalidad que se explicará a continuación:

- **Favoritos:** que ofrece un listado con aquellas actividades que hayan sido clasificadas como favoritos previamente hasta un máximo de 35.
- **Recientes:** que ofrece un listado con aquellas actividades que hayan sido consultadas últimamente hasta un máximo de 35.
- **Alfabéticamente:** que ofrece un listado de eventos ordenado de forma alfabética.
- **Calendario:** podemos visualizar un calendario mensual donde es posible acceder a un día en concreto y listar los eventos asociados a dicha fecha.
- **Categorías:** que permite seleccionar nuevamente todas aquellas categorías de interés, que pueden ser diferentes a las escogidas en un principio.

Si accedemos a un evento en concreto, nos aparece una nueva ventana en la que se puede apreciar el título descriptivo, imagen de la universidad promotora, descripción breve, fecha y lugar de realización y cuatro iconos y una funcionalidad adicional que a continuación se explican:

- **Favoritos:** cada evento ofrece la posibilidad de ser marcado como favorito para después aparecer en la lista anteriormente descrita.
- **Ubicación:** cada evento ofrece la posibilidad de encontrar la ubicación exacta y precisa de donde se vaya a realizar (funcionalidad ofrecida por Google Maps).
- **Inscripción:** cada evento ofrece (si hay posibilidad de obtenerla) la reserva de entradas e inscripción a la actividad mediante un gestor de reservas externo habilitado para tal caso.
- **Compartición:** cada evento permite ser compartido con los contactos del usuario, hecho que le otorga gran extensibilidad.
- **Más Info:** cada evento ofrecerá una descripción detallada acerca de sí mismo enlazando con la noticia de la actividad original de la universidad a la que pertenece.

La funcionalidad ofertada en esta app es indudablemente completa y bien organizada, mostrando distintas opciones de clasificación de eventos y descripciones precisas e información contrastada acerca de ellos. Además ofrece gran variedad de universidades y categorías con las que el usuario puede acceder de forma rápida y sencilla. Sin embargo, presenta una interfaz algo pobre visualmente (una lista de eventos monótona y con colores apagados) y el rendimiento e interacción con el usuario es más bien bajo (se cuelga demasiado y a veces se sale forzosamente de la app). Además se debe de considerar una mejora en la funcionalidad de clasificación ya que existen demasiados botones dedicados a esta misma función (recientes, favoritos y alfabéticamente podrían coexistir en una lista desplegable hacia arriba si mantenemos pulsado, por ejemplo). Por otro lado, los eventos se actualizan diariamente y a través de los moderadores del sistema (se podría realizar al instante), hecho que no otorga privilegios al usuario para poder subir sus propias actividades. Por último recalcar que el usuario accede sin identificarse, lo que da lugar a la accesibilidad global a la aplicación, y un menor control sobre los usuarios que interactúan con la misma.

2.1.1 Conclusión de aplicaciones similares a un gestor de eventos académicos

Tras el estudio de dos de las más influyentes aplicaciones en este ámbito se puede concluir que ambas aplicaciones contienen funcionalidad referente a eventos de forma sistemática y bien estructurada, dando pie a afirmar que se ha dado mucha mayor importancia al contenido que a la forma, pensando más en ofrecer al usuario una experiencia que le sirva aunque para ello tenga que lidiar con una interfaz menos amigable.

Es por ello que muchas de las tecnologías presentes en cada una de ellas podrían ser reutilizables en mayor o menor medida, como por ejemplo la búsqueda de eventos, la posibilidad de su clasificación o categorización, el poder compartirlos mediante otras apps o una sección de eventos del usuario que guarde todos aquellos que le interesen, entre otros.

Por otro lado, ambas apps gozan de buena reputación en sus respectivos entornos (universidades que las suministran), y esto permite dar un visto bueno a la consecución de una aplicación de este tipo en ámbitos académicos puesto que ya se han desarrollado antes en otras comunidades universitarias dando como resultado la plena satisfacción del usuario.

Sin embargo, ambas aplicaciones carecen de un sistema de control de usuarios que permita monitorizar el uso de la app y gestionar su utilización en un ámbito académico, además de tener una interfaz visual algo pobre en dinamismo y lo más importante, no permitir al usuario compartir eventos que él mismo pueda encontrar o incluso proponer, por lo que siempre tendrían que ser actualizados a través del moderador o administrador del sistema. Además, es preciso mencionar la falta de interacción con el usuario en cuanto a comentarios se refiere, con los que se puedan compartir opiniones y pensamientos acerca de un evento determinado.

2.2 Análisis de tecnologías de desarrollo en Android

Antes de proceder a analizar las distintas tecnologías de desarrollo, habría que preguntarse el porqué de escoger Android como sistema operativo en vez de otros disponibles para el mercado de tecnologías móviles como IOS o Windows Phone. La razón principal es la versatilidad y facilidades que ofrece Android a los desarrolladores con herramientas donde poder implementar aplicaciones que varían desde lo simple y sencillo hasta lo complejo e intrincado, tanto si se trata de desarrolladores profesionales como estudiantes o iniciados.

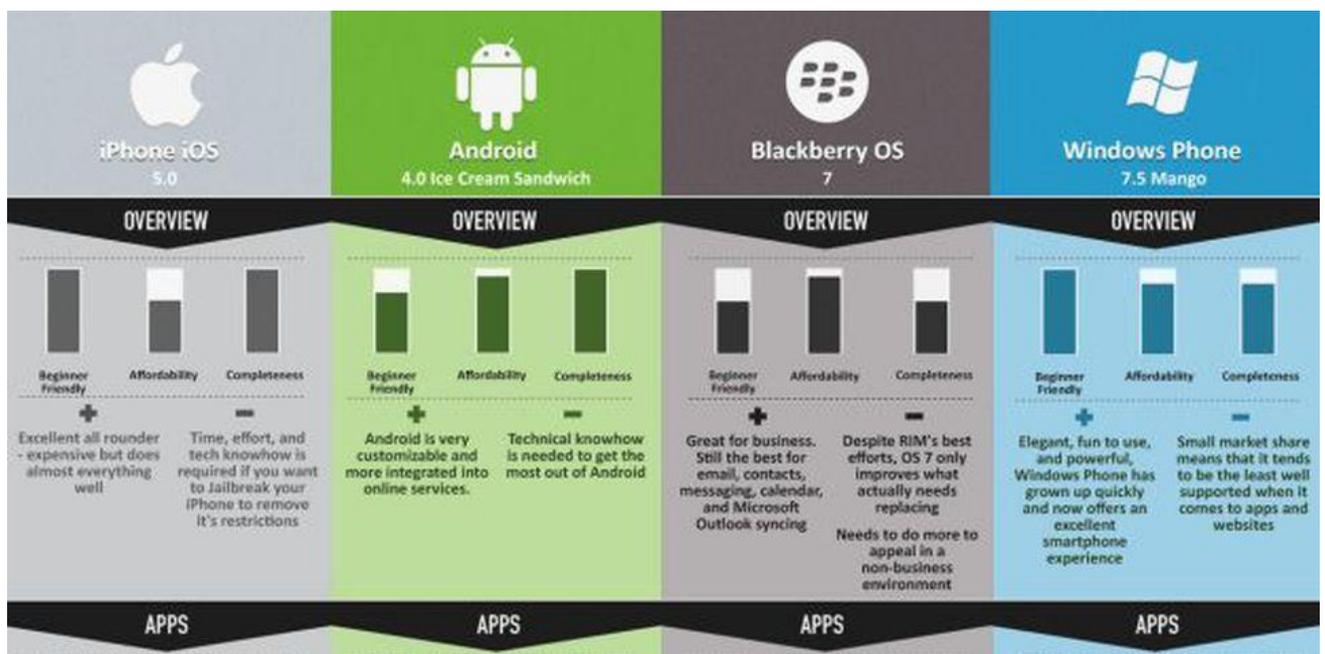


Figura 3: Comparativa entre los distintos sistemas operativos para dispositivos móviles [3]

Otro aspecto a tener en cuenta es la versión de Android sobre la que se implementará la aplicación. En principio se elegirá la última versión disponible compatible con el entorno de desarrollo, que en este caso se trata de Android Marshmallow. A continuación se muestra un gráfico detallado de las versiones disponibles de Android, así como su distribución y uso antes de comenzar la parte de desarrollo de la aplicación.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%

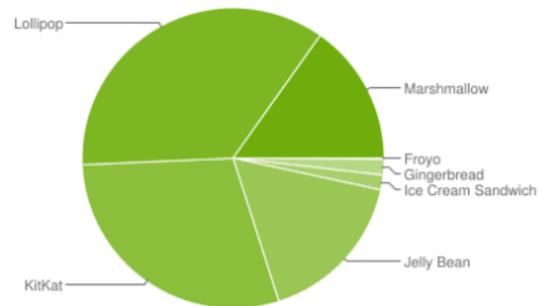


Figura 4: Distintas versiones disponibles de Android a fecha de 19 de enero de 2017 [25]

A continuación, en este apartado, se realiza un estudio exhaustivo acerca de todas aquellas tecnologías disponibles para desarrollar aplicaciones en Android, con el objetivo de seleccionar de entre este amplio abanico aquellas que mejor se adapten a lo que se pretende conseguir con la aplicación en desarrollo.

Así, disponemos de las siguientes herramientas tecnológicas para desarrollar aplicaciones en Android:

- **Android Studio**



Figura 5: Entorno Android Studio

Es el entorno de desarrollo oficial e integrado para la plataforma Android que está en continua actualización [6].

Entre sus características fundamentales, podemos citar que contiene la versión más reciente y actualizada de Android SDK (Software Development Kit, conjunto de herramientas de desarrollo para la plataforma Android) y Android SDK Tools and platform-tools (conjunto de herramientas encargadas del proceso de depuración y testeo de las apps implementadas

en Android), y un sistema de imagen para el emulador Android (que permite crear y probar las aplicaciones Android en diferentes dispositivos virtuales) [5].

Es el entorno multiplataforma de desarrollo predilecto para la implementación de aplicaciones Android, utiliza una licencia de software libre Apache 2.0, y está programado en Java. Además, se trata del IDE oficial de Google en colaboración con JetBrains (compañía especializada en desarrollos de IDEs) y permite la creación de módulos dentro de un mismo proyecto. Como desventajas podemos citar una menor cantidad de plugins con respecto a Eclipse, el sistema de creación de proyectos mediante Gradle puede ser tedioso inicialmente, y el entorno se encuentra en constante desarrollo, lo que podría desembocar en fallos inesperados sin resolución [4].

- Eclipse



Figura 6: Entorno Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma, cuyo uso más común se basa en actuar como plataforma para desarrollar entornos de desarrollo integrados. Los creadores lo definen como una herramienta universal, además de un entorno abierto y extensible [2].

Entre sus características principales podemos mencionar la versatilidad que ofrece para la elaboración de aplicaciones en Android al basarse en la inclusión de plugins funcionales si son requeridos por el programador, y ofrece todas las funcionalidades requeridas por la plataforma Android para el desarrollo de aplicaciones.

Sin embargo, presenta varias desventajas. La primera y más fundamental es que ya no se considera el entorno de desarrollo oficial y la mayoría de programadores han dejado de utilizarlo como IDE para la consecución de proyectos Android. Además, no ofrece las funcionalidades de construcción de paquetes .apk que ofrece el entorno mencionado con anterioridad, Android Studio, y es considerada como una herramienta de menor potencial con respecto a éste [4].

- **Basic4Android**



Figura 7: Entorno Basic4Android

Es una herramienta de desarrollo rápido de aplicaciones Android alternativa a la programación mediante Java y Android SDK, que utiliza el lenguaje de programación Visual Basic, en contraposición a los demás entornos de desarrollo anteriormente mencionados [7].

Sus características principales es que se puede observar el proceso de desarrollo de las aplicaciones que se estén implementando (desarrollo rápido de aplicaciones) y simplifica el proceso de construir interfaces de usuario que atañen a dispositivos con diferente tamaño de pantalla, además de que no es necesario utilizar ningún otro framework para la consecución de proyectos Android.

Sin embargo, no presenta las ventajas dependientes del entorno de desarrollo oficial ni ofrece la funcionalidad completa que éste ofrece, limitándose tan solo a la implementación de aplicaciones de forma rápida. Además, se trata de una plataforma minoritaria alternativa a la utilización del lenguaje predominante Java, que no alcanza la misma cobertura que ofrecen los entornos de desarrollo explicados con anterioridad.

2.2.1 Otras tecnologías empleadas

- **SQLite**



Figura 8: Gestor de bases de datos SQLite

Es un sistema de base de datos relacional contenida en una pequeña biblioteca escrita en el lenguaje de programación C. Se considera como uno de los sistemas gestores de bases de datos relacionales (RDBMS) punteros en la actualidad, al estar accesible en la mayoría de lenguajes de programación genéricos [1].

SQLite presenta como características básicas el almacenamiento de los datos en simples ficheros, accesibles mediante la librería anteriormente mencionada, la compatibilidad con el desarrollo de aplicaciones móviles al incorporar dicha librería en la librería estándar de Android (con algunas clases de ayuda en Java) y a que es utilizado como el sistema prioritario a la hora de desarrollar aplicaciones en Android que incorporen bases de datos relacionales.

Como inconvenientes podemos citar la gran cantidad de esfuerzo que requiere conocer toda la funcionalidad referente al sistema para desarrollar bases de datos relacionales de gran consistencia y extensión, y la obligación de entender el lenguaje Java en un nivel medio-avanzado para poder integrarlo en aplicaciones Android.

- **Biblioteca Volley**

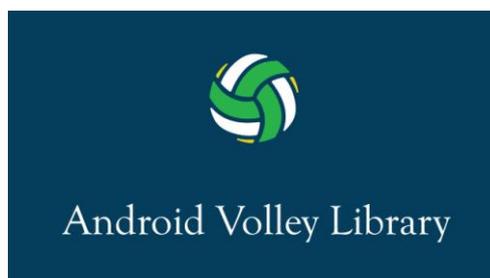


Figura 9: Logo de la biblioteca Volley

La biblioteca Volley es una solución de Networking que ha acelerado y simplificado considerablemente la tarea de comunicaciones, permitiendo de forma sencilla acceder a una entidad remota [10].

Entre sus ventajas podemos destacar la programación automática de peticiones, las conexiones de red concurrentes múltiples, el API de cancelación de peticiones y la posibilidad de personalización para programar reintentos. Como desventajas, *Volley* no es adecuado para descargas grandes de cantidades de información ya que mantiene todas las respuestas en memoria mientras éstas se procesan.

En este caso, se utiliza la biblioteca para conectar y realizar solicitudes y peticiones POST y GET al servidor, de forma que se convierte en el elemento utilizado para establecer las comunicaciones con éste de forma unilateral.

- API de Google Maps



Figura 10: Logo de Google Maps

La API de Google Maps es la encargada de ofrecer servicios relacionados con la ubicación y geolocalización que potencian la experiencia de usuario de forma considerable, dando paso a la identificación de un lugar determinado de forma precisa [13].

Entre sus ventajas podemos encontrar la inclusión de un elemento utilizado en millones de sitios WEB y aplicaciones o la posibilidad de ubicar un elemento determinado de forma precisa. Como desventajas, la utilización de la API en su totalidad resulta muy compleja si es necesario hacer un uso exhaustivo de ésta.

En este caso, se utiliza la API para localizar los eventos en el espacio de forma precisa, al pulsar en el icono de Google Maps situado en la pantalla del evento en cuestión. Así, hacemos uso de la propia aplicación de Maps para geolocalizar una actividad de forma exacta.

- REST Webservices & Dropwizard



Figura 11: Logo y eslogan de Dropwizard

Los servicios web son una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, entre los que se encuentra REST [23].

REST (Representational State Transfer) es un estilo de arquitectura software para sistemas hipermedia distribuidos como la WWW, que especifica restricciones tales como una interfaz uniforme, que aplicada a un servicio web induce propiedades deseadas como rendimiento o escalabilidad, que permiten a dichos servicios trabajar de forma óptima [24].

Dropwizard es un Framework o conjunto de librerías de Java que sirve para desarrollar servicios REST de forma sencilla e intuitiva por medio de etiquetas, ofreciendo un servicio web (*web service*) de calidad en el menor tiempo posible [12].

En este caso, se utiliza en el lado del servidor para atender y tratar las peticiones llegadas a través del cliente por medio de ficheros JSON, desembocando en el almacenamiento de información en el servidor a través de ficheros, elemento indispensable en la aplicación.

2.2.2 Conclusión sobre el estudio de tecnologías de desarrollo en Android

Tras el estudio de varias tecnologías predominantes en el campo de desarrollo de aplicaciones Android, concluimos afirmando que existen varios entornos propicios para la consecución de proyectos en dicha plataforma, pero que no presentan tanta incidencia como el analizado en primera instancia, Android Studio, que presenta toda la funcionalidad requerida para elaborar los productos software anteriormente mencionados, tratándose de la herramienta prácticamente universal utilizada para este propósito. Otras aproximaciones interesantes desembocan en la utilización de Eclipse o Basic4Android, sin embargo, no alcanzan tanto éxito debido a una limitación en sus capacidades a la hora de realizar proyectos para plataformas móviles.

Por otro lado, se estudiaron tecnologías complementarias con respecto a las bases de datos que soportará la aplicación. Sin embargo, la más eficiente en cuanto a creación de bases de datos relacionales a media y gran escala es SQLite. Ésta ofrece gran acoplamiento para el desarrollo de aplicaciones en Android ya que dispone de una librería especializada que, integrándose en la aplicación, desemboca en el acceso a la funcionalidad relacional directamente, ofreciendo sencillez y eficacia. Es por eso que se considera el sistema gestor de base de datos predominante a la hora de implementar aplicaciones móviles.

Por último, se hace uso de tres librerías complementarias que simplifican de forma considerable tareas concretas relacionadas con el Networking, localización y tratamiento de información en el servidor, como son la biblioteca Volley, la API de Google Maps y el *framework* Dropwizard, respectivamente. A través de dichos elementos, la conexión y almacenamiento de datos en el servidor se vuelve mucho menos complejo, desembocando en una aplicación fluida y completa. Además, la tarea de geolocalización se vuelve muy sencilla al delegar su funcionalidad a la aplicación Google Maps, que desemboca en un producto que utiliza una de las mejores APIs profesionales en tareas de localización.

3 Diseño

3.1 Ciclo de vida

Para el desarrollo de la aplicación se ha seguido un modelo de ciclo de vida basado en cascada con realimentación, como el mostrado en la siguiente figura:

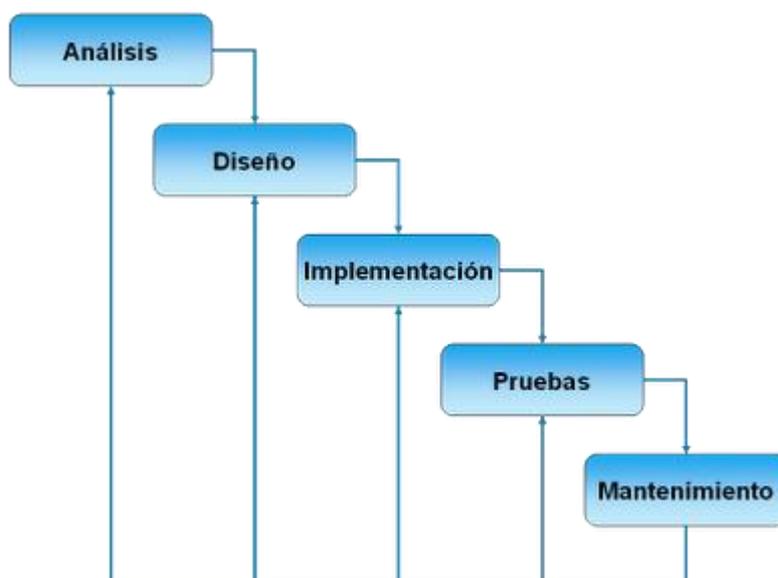


Figura 12: Ciclo de vida en cascada con realimentación

A continuación se detallan las diversas fases que componen el ciclo de vida anteriormente mencionado, donde se tiene en cuenta que cada fase es supervisada por el equipo que conforma el proyecto:

- **Análisis:** en esta etapa se procedió al estudio de aplicaciones similares del entorno académico reflejadas anteriormente ([véase Sección 2.1](#)) además de un estudio de las tecnologías que podrían ser necesarias para el desarrollo y que mejor encajarían con lo que se pretendía conseguir con el proyecto ([véase Sección 2.2](#)). En adición a esto, se realizó un análisis de requisitos funcionales y no funcionales a partir de las necesidades de los usuarios finales [\[ANEXO B\]](#).
- **Diseño:** en esta etapa se procedió a realizar diagramas entidad-relación referentes a la base de datos, un diagrama de secuencia que detecta las tareas que solicitará el cliente al servidor y éste a la base de datos y las maquetas de la aplicación [\[ANEXO C\]](#).
- **Implementación:** en esta etapa se procedió a realizar la codificación de la aplicación a través del lenguaje Android, y la implementación de la base de datos subyacente.
- **Pruebas:** esta etapa se realizó de forma conjunta con la implementación a medida que se avanzaba y también al finalizar su desarrollo, englobando pruebas unitarias, de integración y de validez y verificación.
- **Mantenimiento:** etapa fuera del alcance de este TFG, se considera la corrección de errores y mejora del proyecto una vez ha finalizado la entrega de este al usuario final.

3.2 Base de datos

En esta sección se describirá en detalle el proceso de creación de la base de datos correspondiente a la aplicación J-Event, necesaria para almacenar toda aquella información relevante que contiene. Entre los datos a almacenar se encuentran los usuarios que se registran, los eventos o actividades y sus características, así como las categorías de evento que crea el usuario. Para llevar esto a cabo se opta por almacenar los datos en una base de datos dinámica, actualizada con cada actividad que se realiza dentro de la app.

3.2.1 Diagrama Entidad – Relación

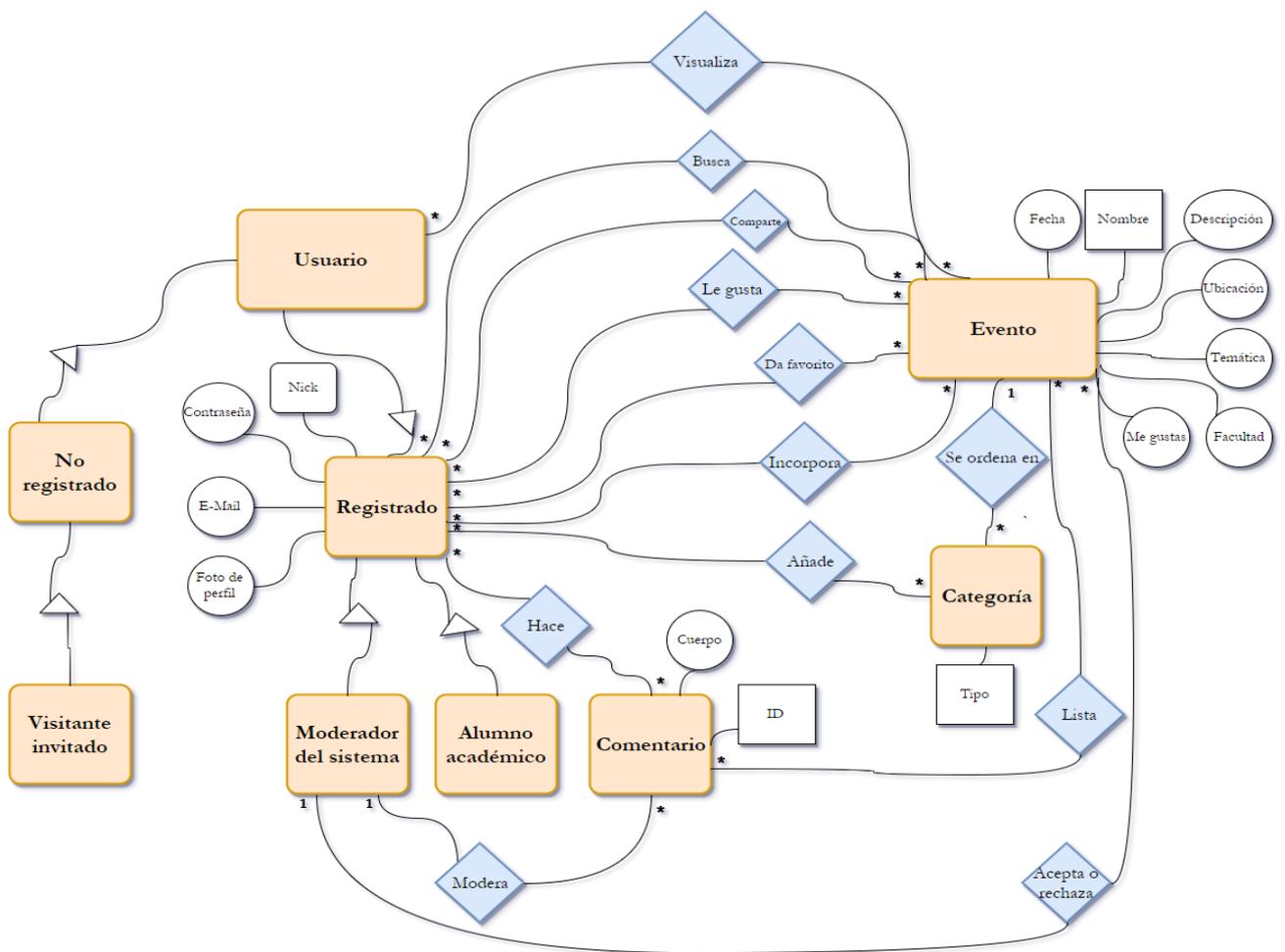


Figura 13: Diagrama entidad relación de la aplicación

Este diagrama refleja las entidades y relaciones que abarcaría la aplicación en su base de datos, comentadas en detalle en la siguiente subsección (3.2.2).

3.2.2 Entidades de la base de datos

En la figura 10 se muestra el diagrama entidad-relación correspondiente a la base de datos de la aplicación desarrollada e implementada en el gestor de base de datos SQLite. A continuación se listan las entidades, así como sus relaciones y atributos, de forma detallada:

- **Usuario:** representa todos aquellos actores que pueden formar parte de la aplicación e interactuar con ella. Así mismo, como se vio en el apartado de análisis de requisitos un usuario de la aplicación se divide a su vez en usuarios registrados y no registrados. Cualquier usuario (tanto registrado como no) puede visualizar los eventos en la aplicación, representado esto mediante la relación correspondiente y la cardinalidad (*...*).
- **Usuario no registrado:** que deriva del usuario anteriormente mencionado. Éste tan solo contiene la funcionalidad de visualizar eventos ya comentada. Como derivación directa, surge la figura del visitante invitado.
- **Visitante invitado:** que deriva del usuario no registrado anteriormente mencionado, compartiendo misma funcionalidad.
- **Usuario registrado:** que deriva del usuario anteriormente mencionado. Contendrá los atributos de Nick de usuario (que se comportará como clave de la entidad), contraseña asociada a éste, e-mail académico que se verificará y foto de perfil identificativa. Entre las acciones que puede realizar esta entidad se encuentran la posibilidad de buscar, compartir, darle me gusta, dar favorito (**NOTA más abajo**) o incorporar eventos, añadir una categoría nueva de evento o realizar un comentario. Todas estas relaciones implican que muchos usuarios registrados sean capaces de realizar muchas de estas acciones, representado esto con la cardinalidad muchos a muchos (*...*) en la relación correspondiente. Del usuario registrado derivan el moderador del sistema y el alumno académico perteneciente a institución.
- **Moderador del sistema:** que deriva del usuario registrado anteriormente mencionado. Tan sólo habrá uno en todo el sistema, encargado de realizar funcionalidad específica de moderar los comentarios que se efectúen, así como los eventos que se intenten incorporar a la aplicación, además de poder eliminar los eventos cuya fecha haya sobrepasado la actual. Esto está representado mediante la relación correspondiente y la cardinalidad uno a muchos (1...*).
- **Alumno académico:** que deriva del usuario registrado anteriormente mencionado, compartiendo su misma funcionalidad.
- **Comentario:** entidad que representa aquel aporte textual que realiza un usuario registrado acerca de información relevante correspondiente a un evento dado. Como atributos posee un identificador (que se corresponderá con la clave de la entidad) y un cuerpo que contendrá la información textual anteriormente citada. Antes de ser incluido en la aplicación deberá ser moderado por el moderador del sistema.
- **Evento:** que representa las actividades y principal foco de interés de la aplicación. Como atributos posee un nombre que lo identifique (que actuará como clave de la entidad), la fecha en la que tendrá lugar, una descripción precisa acerca del contenido y demás información relevante, la ubicación donde tendrá lugar, la temática asociada a éste, la facultad o entidad organizadora y la popularidad del evento (que se correspondería con la cantidad de me gustas obtenidos). Los distintos eventos se ordenan en categorías, y contienen comentarios, representadas estas dos acciones mediante las relaciones correspondientes y la cardinalidad muchos a muchos (*...*). Antes de ser incluidos en la aplicación deben de ser aceptados o rechazados por el moderador del sistema.

- **Categoría:** que representa la clasificación y ordenación de los eventos anteriormente mencionados en diversos tipos. Contiene el atributo tipo que actúa como clave de la entidad.

NOTA: Es importante recalcar la diferencia entre los siguientes conceptos: *dar me gusta* es una propiedad inherente del evento que es visualizada por el resto de usuarios, con el fin de listar los eventos más populares y que la gente pueda ver cuánto de populares son en cada momento. *Dar favorito a un evento* simplemente lo añade a una categoría propia del usuario donde se listan sus eventos de interés, con el fin de accederlos más rápida y fácilmente en el futuro.

3.3 Diagrama de secuencia

A continuación se muestra el diagrama de secuencia que refleja las tareas que el cliente solicita al servidor, y a su vez las que éste solicita a la base de datos. Este diagrama clarifica aspectos relacionados con la conexión entre los usuarios y el servidor y base de datos de la aplicación, haciendo más sencillo el proceso de codificación. Resultaría impracticable abarcar toda la funcionalidad de la aplicación en este diagrama, por lo que se ha decidido incluir en el diagrama las tareas de mayor importancia dentro de la aplicación.

En él se pueden distinguir varias tareas referentes a la funcionalidad en el apartado de análisis mencionado:

- **Inicio de sesión o registro** por parte de todos aquellos usuarios registrados, en los cuales se conecta primero con el servidor y después con la base de datos para incluir los datos del usuario en el caso de registro y comprobar que no existen unos iguales o para comprobar que los datos de usuario se han introducido de forma correcta en el caso del inicio de sesión.
- **Modificación de los datos de usuario** en el caso de usuarios registrados, en los cuales se conecta primero con el servidor y después con la base de datos para actualizar los datos pertinentes.
- **Aceptar/rechazar eventos** por parte del moderador del sistema, en los cuales se conecta primero con el servidor y después con la base de datos para incluir los datos pertinentes a dicho evento si es aceptado
- **Eliminar evento** por caducidad por parte del moderador del sistema, en el cual se conecta primero con el servidor y después con la base de datos para actualizar el evento y eliminarlo.
- **Dar me gusta, favorito o hacer comentario en evento** por parte de los usuarios registrados del sistema, en los cuales se conecta primero con el servidor y después con la base de datos para actualizar los datos pertinentes en el evento correspondiente.
- **Añadir una categoría de evento** por parte de los usuarios registrados del sistema, en la cual se conecta primero con el servidor y después con la base de datos para actualizar los datos pertinentes a las categorías de evento existentes.
- **Retorno a la pantalla principal** cuando el usuario se encuentra en las pantallas de evento o categorías de evento.
- **Añadir un evento** pendiente de moderar al sistema.
- **Hacer log out** por parte de usuarios tanto registrados como no registrados.

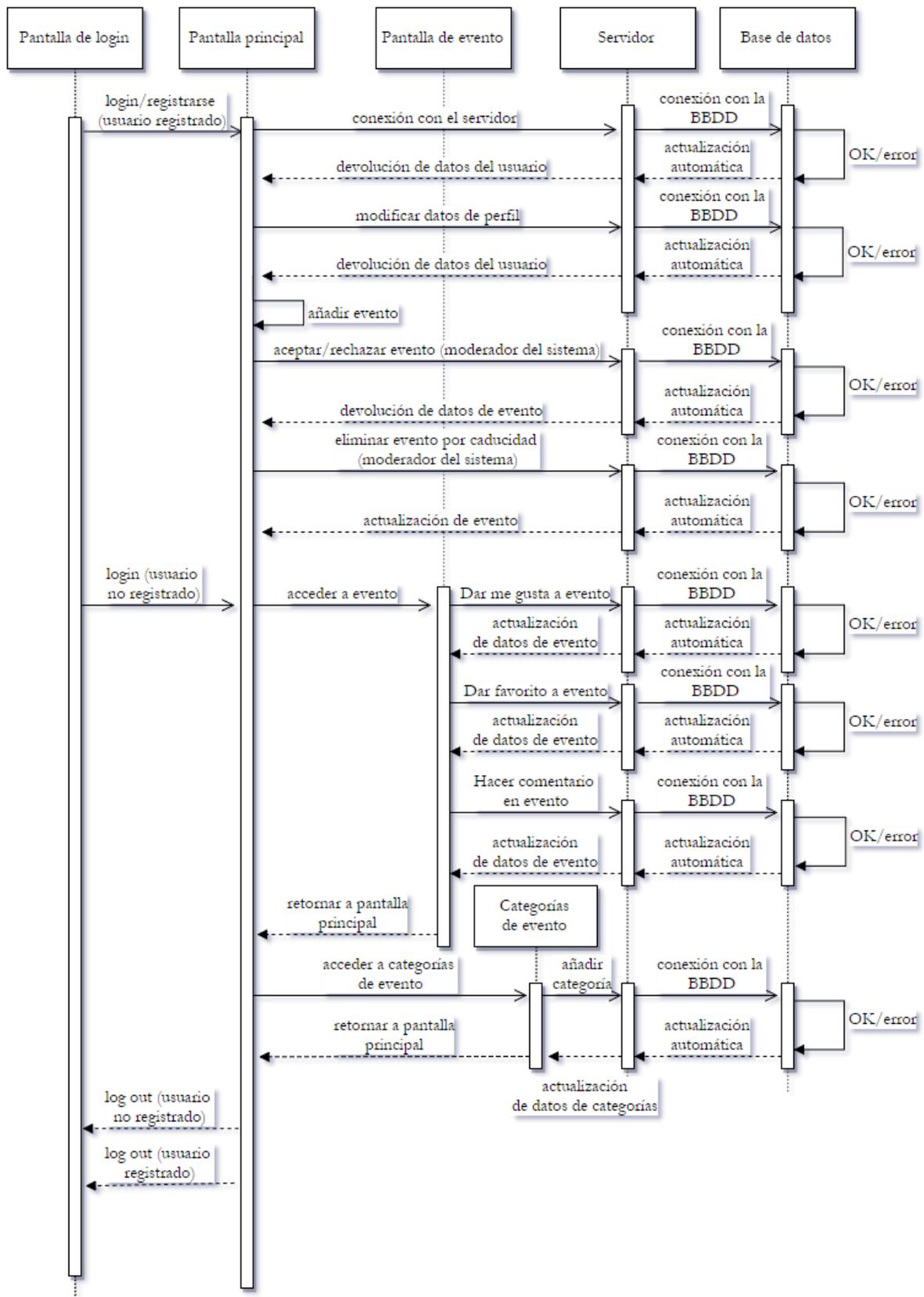


Figura 14: Diagrama de secuencia de la aplicación

4 Desarrollo

La fase de desarrollo conlleva la utilización de una arquitectura cliente-servidor, en la que se distingue de forma plausible los usuarios que interactúan y realizan peticiones a la aplicación, del servidor web que es el encargado de recibir dichas solicitudes, procesarlas y dar una respuesta lógica al demandante. Entre las tareas fundamentales del servidor, se encuentran el procesamiento de consultas por parte del usuario al interactuar con la aplicación, y la actualización de información con el objetivo de que ésta se muestre en todo momento de forma consistente.

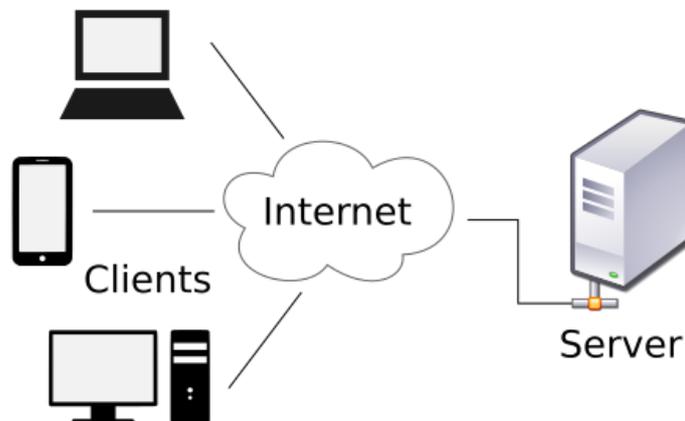


Figura 15: Arquitectura Cliente-Servidor [9]

El desarrollo se lleva a cabo siguiendo el patrón Modelo-Vista-Controlador, que separa el manejo y utilización de los datos, así como la lógica de aplicación o negocio, de la representación de dichos datos en forma de interfaz de usuario.

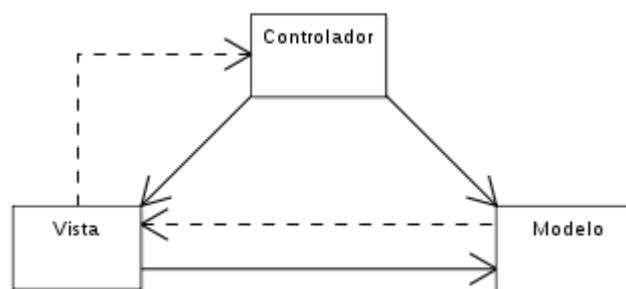


Figura 16: Patrón Modelo-Vista-Controlador (MVC) [15]

Así, la labor de manejo, acceso y manipulación de información se lleva a cabo a través de una base de datos implementada a través del gestor SQLite, que contiene toda la funcionalidad de representación de los datos que constituyen la aplicación. Esta base de datos se constituye de forma local en el dispositivo del usuario a la hora de arrancar la aplicación, y se va actualizando y modificando su contenido a medida que se interactúa con ésta en puntos específicos explicados más adelante.

Posteriormente, se implementó una base de datos a través del gestor PostgreSQL, que se utilizará en versiones futuras del proyecto cuando el volumen de información disminuya de forma considerable el rendimiento de la aplicación. Esta nueva base de datos sustituirá a la anteriormente citada y estará implementada en el lado del servidor, que será el encargado de gestionar su acceso y manejo, en lugar de la base de datos local implementada en el dispositivo de cada usuario.

La lógica de negocio o aplicación se implementa fundamentalmente en Java, lenguaje integrado para el desarrollo de aplicaciones para dispositivos móviles, y se constituye como controlador del patrón MVC anteriormente mencionado, actuando como intermediario realizando solicitudes de información al modelo de datos cuando el usuario realiza una petición, y realizando peticiones a la vista cuando se solicita un cambio en la forma en la que se presenta la información del modelo.

Por último, la vista se implementa de forma completa a través de Android, al tratarse de una aplicación enfocada a dispositivos móviles, constituida por las diversas interfaces asociadas a cada una de las pantallas que se presentan en la aplicación.

4.1 Lógica de la aplicación

En una primera instancia, se comienza el desarrollo de la parte lógica de la aplicación teniendo en cuenta que posteriormente ésta irá integrada en Android, actuando como controlador accediendo a la base de datos y servidor cuando se necesite acceder a la información o actualizarla, y accediendo a las vistas que constituyen la aplicación para presentarlas con la información adecuada en cada momento.

Así, inicialmente tan sólo se crean las clases necesarias cuyos atributos y métodos cumplen los requisitos analizados en la fase de análisis, constituyéndose como *beans* sencillos que reflejan la funcionalidad indispensable de la que se hará uso en la aplicación.

4.1.1 Bean usuario (user)

Este *bean* abstrae la funcionalidad referida al usuario de la aplicación, que se define como la entidad principal que hará uso de la funcionalidad dispuesta con el fin de interactuar con los elementos que se vayan presentando. Así, se define una interfaz de usuario que engloba a todos aquellos clientes que harán uso de la aplicación, y se definen dos clases que la implementan: usuario registrado y usuario no registrado. El usuario registrado es todo aquel estudiante o miembro de la comunidad universitaria que ha cumplido un proceso de inscripción en la aplicación de forma satisfactoria, un caso especial de este tipo de usuario es el administrador del sistema. El usuario no registrado es todo aquel cliente que entra de paso a la aplicación y cuya funcionalidad dentro de ésta se predispone de forma limitada.

4.1.2 *Beans* evento (event) y categoría (category)

Estos *beans* abstraen la funcionalidad referida a todas aquellas actividades o eventos que se integran dentro de la aplicación, que se constituyen como el elemento central de ésta, albergando todas aquellas noticias de carácter informativo para el miembro de la comunidad universitaria o cliente que haga uso de la aplicación, así como cada una de las categorías de evento en las que puede encontrarse, como por ejemplo *Meeting*, *Reunion*, *Learning*, *Sport*, o *Culture*, entre otras categorías.

4.1.3 *Bean* comentario (commentary)

Este *bean* refleja la funcionalidad de realizar comentarios dentro de un evento determinado por parte de un usuario de la aplicación, ya sea registrado o no registrado; en este último caso de forma anónima.

A continuación se integran todos estos *beans* en el proyecto Android, que actuarán como fuentes base de información a las que accederán cada una de las actividades o fragmentos correspondientes a las pantallas de la aplicación. El flujo entre actividades se puede observar en el diagrama incluido en el [\[ANEXO D\]](#).

La lógica de aplicación como tal en Android se implementa partiendo del diagrama de actividades realizado en el apartado de diseño algo más escueto que el presentado en el [\[ANEXO D\]](#), que simplifica la labor de forma considerable, dando pie únicamente a la codificación e integración de éstas de tal manera que funcione sin incidencias. Así se dispone la lógica en forma de clases controladoras que accederán al modelo de datos y vistas consecuentemente.

4.1.4 Actividad de log in (LoginActivity)

La lógica parte de una primera pantalla de log in en la que se nos muestran sendos campos de introducción de los datos de usuario (pseudónimo o *nick* y contraseña), respectivamente.

Cuando se introducen debidamente los datos, éstos se envían al servidor por medio de una petición POST a través de la librería *Volley*, que es quien los procesa, y comprueba que existe un usuario con los datos introducidos, dando paso al inicio de sesión, y almacenando los datos en las preferencias del usuario. En caso contrario se muestra un mensaje informativo.

4.1.5 Actividad de registro (RegisterActivity)

Cuando un usuario desee registrarse en la aplicación, deberá cumplimentar debidamente los datos que se especifican, teniendo en cuenta que no puede haber un usuario con un mismo *nick* (informando la aplicación sobre este error si así ocurriera) e introduciendo un correo válido. Una vez introducidos los datos y validados, se envía una petición POST al servidor para que la procese.

NOTA: en principio se realiza un control simple del correo, en versiones futuras se propone realizar un control que envíe un correo de confirmación al usuario con el que pueda entrar en la aplicación.

4.1.6 Preferencias del usuario (JEventPreferenceActivity)

Esta actividad se encarga de gestionar los datos relativos al usuario necesarios para todas las actividades de la aplicación, como son sus datos de inicio de sesión o si se trata de un usuario registrado, no registrado o el administrador del sistema. Se almacenan en el elemento preferencia de Android que de forma consistente albergará una cantidad de información mínima usable en cada una de las actividades y fragmentos de la aplicación.

4.1.7 Actividad inicial y lista de eventos (EventListActivity)

Esta actividad inicial se encarga de listar de forma dinámica todos aquellos eventos incluidos en la aplicación que no estén pendientes de moderación. Los eventos son tomados del servidor a través de una solicitud GET, que alberga una lista con todos aquellos eventos disponibles en la aplicación.

4.1.8 Actividad de eventos (EventActivity)

En esta actividad se nos muestra la información relativa al evento accedido, su nombre, la fecha donde tendrá lugar, la dirección, la entidad organizadora, la categoría a la que pertenece, el usuario que ha subido el evento, los me gusta que tiene, su descripción y su lista de comentarios dispuesta de manera dinámica.

Una funcionalidad dispuesta en esta actividad es la de subir un comentario a dicho evento, en donde deberemos cumplimentar debidamente los datos, tras lo cual se enviará el comentario a través de una solicitud POST al servidor en forma de pendiente de moderación. Una vez realizado esto, el administrador del sistema podrá aceptar o denegar el comentario e incluirlo o eliminarlo de la aplicación, cuando acceda a la pantalla de pendientes.

4.1.9 Barra de acción y menú

Dispondremos de un menú en la barra de acción que contiene las funcionalidades de añadir un evento a la aplicación, mostrar la ayuda, o eliminar los eventos fuera de plazo (este último solo posible para el administrador del sistema).

- **Añadir evento** (*UploadEvent*): consistente en una actividad en la que se deben cumplimentar debidamente todos los campos de carácter obligatorio relativos al evento que se desee incluir en la aplicación. Una vez pulsado el botón de *upload*, el evento se envía al servidor a través una solicitud POST, tras lo cual el administrador del sistema lo moderará e incluirá o denegará el evento. Además, el evento se incluye dentro de la base de datos local, con el objetivo de acceder a los datos de una forma mucho más rápida, en casos específicos.
- **Eliminar eventos fuera de plazo**: el administrador del sistema, en cualquier momento, podrá eliminar todos aquellos eventos que hayan sobrepasado la fecha actual, con el

objetivo de actualizar la información de forma consistente. Esto se realiza accediendo al servidor a través de una petición POST eliminando todos los eventos deseados.

Además es posible acceder al menú lateral a través del botón situado en la esquina superior izquierda de la pantalla, donde se nos muestra la imagen y datos del usuario, además de las funcionalidades de acceso a las pantallas de perfil (*profile*), filtrar (*filter*), favoritos (*favourites*), y log out (*log out*), además de la pantalla de pendientes (*pending*) en caso de que el usuario tenga permisos de administrador.

- **Perfil** (*profile*): solo accesible por usuarios registrados, en esta pantalla se muestran los datos de usuario que ha iniciado sesión, tomados de las preferencias de usuario.
- **Filtrar** (*filter*): en esta pantalla se elige una de las categorías de evento posibles y el sistema automáticamente lista todos aquellos eventos pertenecientes a la misma.
- **Favoritos** (*favourites*): solo accesible por usuarios registrados, en esta pantalla se listan todos los eventos que el usuario ha marcado como favoritos.
- **Pendientes** (*pending*): solo accesible por el administrador del sistema, se listan de forma dinámica los eventos y comentarios pendientes de moderación a través de sendas solicitudes GET. Pulsando cualquier evento o comentario se procede a su aceptación, y pulsando sendos botones flotantes se eliminan eventos o comentarios, respectivamente, haciendo solicitudes correspondientes al servidor en un caso u otro.
- **Log out**: se eliminan las preferencias del sistema asociadas al usuario y se cierra la sesión dentro de la aplicación.

4.2 Modelo de datos

El modelo de datos representa exclusivamente el tratamiento de información, es decir, el acceso, modificación y actualización de los datos. Es el encargado de enviar a la vista aquella parte de la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan a este modelo por parte del controlador.

En esta aplicación, se definen dos tipos fundamentales de modelos de datos:

4.2.1 Base de datos local

La base de datos local se implementa a través del sistema gestor de bases de datos SQLite. Esta base de datos se constituye a la hora de arrancar la aplicación, formando las tablas y atributos de éstas al comienzo.

Se implementa siguiendo estrictamente el diagrama entidad-relación desarrollado en el apartado de diseño. Las diversas tablas se van rellenando con entradas a medida que se hace uso de la aplicación y se accede a funcionalidades concretas.

En la aplicación se crean diversas clases que manejan la base de datos desde su creación hasta su destrucción, pasando por su manipulación y actualización:

- Se dispone de un esquema de base de datos (*EventDataBaseSchema*) que almacena los campos principales que constituyen la fuente de información. Está formado por las siguientes tablas:
 - **Tabla categoría** (*CategoryTable*): correspondiente a las diversas categorías a las que puede pertenecer un evento. Tiene como atributos el *tipo* de categoría (como clave primaria) y el *creador* de la misma (que actúa como clave foránea).
 - **Tabla comentarios** (*CommentTable*): correspondiente a los comentarios que realizan los usuarios sobre los eventos dispuestos en la aplicación. Tiene como atributos el *ID* del comentario (como clave primaria), su *cuerpo*, el *usuario* que lo realiza (como clave foránea), el *evento* al que pertenece (como clave foránea) y si está *pendiente* de moderación.
 - **Tabla eventos** (*EventTable*): correspondiente a los distintos eventos que se van incluyendo en la aplicación. Tiene como atributos la *fecha* donde tiene lugar el evento, el *nombre* (como clave primaria), su *descripción*, la *ubicación* donde tendrá lugar, la *entidad* organizadora que lo gestiona, el número de *me gustas* obtenido, la *categoría* a la que pertenece (como clave foránea), el *creador* del evento (como clave foránea) y si está *pendiente* de moderación.
 - **Tabla usuarios** (*UserTable*): correspondiente a los usuarios que ingresan en la aplicación, además del administrador del sistema. Tiene como atributos el *ID* de usuario (como clave primaria), su *Nick*, su contraseña o *password*, su correo o *email* y su *tipo*: registrado o no registrado.
 - **Tabla eventos favoritos** (*UserFavouriteEventTable*): correspondiente a los eventos favoritos de los usuarios. Tiene como atributos un *ID*, el *nick* del usuario (como clave foránea) y el *nombre* del evento (como clave foránea).
- Para cada una de las tablas anteriormente mencionadas se dispone de su *wrapper* correspondiente, que envolverá la funcionalidad de parseo de las entradas de la tabla correspondiente, utilizando el elemento cursor de Android, para acceder a la base de datos de forma sencilla.
- Repositorios (*repository*) para cada una de las tablas anteriormente mencionadas, junto con la factoría de creación del repositorio (siguiendo el Patrón Factory Method), que albergan los métodos de acceso a la base de datos, de forma sencilla y que estarán implementados por la clase principal.
- Por último, la clase principal (*JEventDataBase*) encargada de englobar toda la funcionalidad anteriormente mencionada de creación, acceso y modificación de la base de datos. Dispone de métodos específicos para ello como *onCreate* y *onUpgrade* que se ejecutan al iniciarse la aplicación y al actualizarse los datos respectivamente. Además, dispone de métodos implementados de cada uno de los repositorios mencionados con anterioridad, que accederán a la base de datos por medio de sentencias SQL y los wrapper anteriormente citados que facilitarán la conexión por medio de los cursores.

4.2.2 Modelo de datos en el servidor

Además de la base de datos local, es preciso un mecanismo de actualización y almacenamiento de información del lado del servidor, de forma que, cuando éste recibe alguna petición, se actualiza su estructura interna y permite acceder a los datos actualizados en cada momento.

Esto se ha conseguido a través, primero y fundamentalmente de la librería *Volley*, que facilita la interacción entre cliente y servidor al proporcionar mecanismos de envío de peticiones para Android sin ninguna dificultad y del framework *DropWizard*, que hace uso de los servicios web REST para acceder a información del lado del servidor de forma sencilla. La información se guarda en el servidor, y se accede por medio de la aplicación cuando sea necesario, obteniendo su contenido y actualizando el modelo de la aplicación en consecuencia (a través de solicitudes al servidor GET), o enviando información con el objetivo de actualizar los datos de forma consistente (a través de solicitudes al servidor POST).

4.3 Vista

Las vistas de la aplicación se definen como la parte visual en forma de interfaz con la que el usuario podrá interactuar en todo momento desde la aplicación. En este caso se implementan por medio de recursos Android en forma de *layouts* que disponen la información de forma amigable e intuitiva para el usuario.

Ligadas íntimamente al controlador y al modelo, éstas se inflan (es decir, se asocian a la actividad y se muestra visualmente) por medio de las actividades explicadas con anterioridad, tomando los datos del modelo de datos visto en el apartado anterior.

4.3.1 Pantallas de carga (*splash_screen*) y de Log in (*activity_login*)

Al arrancar la aplicación aparece una primera pantalla de carga en la que se muestra el logotipo de la aplicación, dando paso a la pantalla de inicio de sesión. En la pantalla de log in se nos muestran sendos campos de introducción de los datos de usuario (nick y contraseña, respectivamente), además de las opciones de entrar si se han introducido de forma válida los datos, entrar como visitante, salir, o la posibilidad de registrarse en el sistema.

4.3.2 Pantalla de registro (*activity_register*)

Cuando un usuario desee registrarse en la aplicación, deberá acceder desde la pantalla de inicio de sesión (*Not account yet? Create a new one*). Una vez en la pantalla de registro, se nos muestran todos aquellos datos necesarios para la inclusión de un nuevo usuario en el sistema, nick, contraseña y correo.

4.3.3 Pantalla inicial y lista de eventos (*fragment_event_list*)

Esta pantalla inicial, junto con su fragmento asociado, se encarga de listar de forma dinámica todos aquellos eventos incluidos en la aplicación que no estén pendientes de moderación, pudiendo acceder a la pantalla de evento específico pulsando en cualquier actividad. Esto se consigue a través de una vista recicladora que nos muestra en cada

momento la información que el usuario solicite, ocultando el resto y optimizando así el rendimiento.

4.3.4 Pantalla de eventos (*fragment_event*)

Cuando pulsamos cualquiera de los eventos listados en la pantalla inicial, accedemos a la pantalla de evento, donde se nos muestra la información relativa al evento accedido, su nombre, la fecha donde tendrá lugar, la dirección, la entidad organizadora, la categoría a la que pertenece, el usuario que ha subido el evento, los me gusta que tiene, su descripción y su lista de comentarios dispuesta de manera dinámica. Además se completa la funcionalidad de localización pulsando en la imagen del mapa, que redirigirá al usuario a la aplicación Google Maps mostrándole la ubicación exacta donde tendrá lugar el evento.

Otra funcionalidad dispuesta en esta actividad es la de subir un comentario a dicho evento, que nos redirigirá a la pantalla de subir comentario (*UploadComment*) en donde tan sólo se dispone de un campo a rellenar, el cuerpo del comentario.

Además, se dispone de los botones de dar me gusta al evento, marcarlo como favorito, o solo para el administrador, borrar todos los comentarios del evento.

4.3.5 Barra de acción y menú

Dispondremos de un menú en la barra de acción que contiene los botones de añadir un evento a la aplicación, mostrar la ayuda, o eliminar los eventos fuera de plazo (este último solo posible para el administrador del sistema).

- **Añadir evento** (*activity_upload_event*): consistente en una pantalla en la que se deben se nos muestran los campos relativos al evento que se desee incluir en la aplicación, tales como su nombre, la fecha, la descripción, la ubicación, la entidad organizadora y la categoría a la que pertenece.
- **Ayuda** (*activity_help*): actividad que nos muestra la funcionalidad básica que se puede llevar a cabo dentro de la aplicación.
- **Eliminar eventos fuera de plazo**: el administrador del sistema, en cualquier momento, podrá eliminar todos aquellos eventos que hayan sobrepasado la fecha actual, con el objetivo de actualizar la información de forma consistente.

En el menú lateral encontramos los siguientes botones:

- **Perfil** (*fragment_menu_profile*): en esta pantalla se muestran los datos de usuario que ha iniciado sesión.
- **Pendientes** (*fragment_menu_pending*): solo accesible por el administrador del sistema, se listan de forma dinámica los eventos y comentarios pendientes de moderación. Pulsando cualquier evento o comentario se procede a su aceptación, y pulsando sendos botones flotantes se eliminan eventos o comentarios, respectivamente.
- **Filtrar** (*fragment_menu_filter*): en esta pantalla se muestra un menú desplegable a través del cual se podrá seleccionar una categoría de evento y filtrar las actividades por dicho tema.
- **Favoritos** (*fragment_menu_favourites*): en esta pantalla se muestran los eventos marcados como favoritos

5 Integración, pruebas y resultados

En este apartado, se especifican las investigaciones empíricas y técnicas que se han llevado a cabo a lo largo de todo el ciclo de vida de desarrollo de la aplicación, en la que se comprueban la validez y verificación de cada una de las partes que la componen, además de la correcta integración de éstas para formar el sistema completo que otorgan al proyecto fiabilidad y solidez de cara a la respuesta ante problemas o fallos que se puedan cometer al usar o interactuar con la aplicación.

Así, distinguimos varios tipos de pruebas que se enumeran a continuación, y que han llevado a la consecución de un producto software eficiente.

5.1 Pruebas funcionales

Son aquellas pruebas basadas en la correcta comprobación y validez de los requisitos estipulados en la fase de análisis, en este documento presentes en el [\[ANEXO B\]](#) [22]. Atendiendo a su clasificación, nos encontramos con diversos tipos.

5.1.1 Pruebas unitarias

Son aquellas pruebas de caja blanca que comprueban la validez y verificación de cada unidad de código por separado, sin tener en cuenta las demás [18].

En esta aplicación, se ha sometido a cada unidad de código a un proceso de comprobación de corrección de parámetros, validez en las estructuras de datos usadas, así como la correcta secuenciación del programa, es decir, que haga lo que se espere de ella de forma que siga un proceso secuencial.

Esto se lleva a cabo en cada una de las clases Java que componen el proyecto, a excepción de los *beans* mencionados en el apartado de diseño, ya que éstos contienen funcionalidad básica que no precisa de comprobación (están compuestos tan sólo de constructores y métodos setter y getter).

Las pruebas se realizan a través del paso de parámetros inválidos en una primera instancia, comprobando mediante ejecución del programa paso a paso, analizando qué resultado arrojan y si es el esperado, además de cerciorar si pueden producir algún tipo de error que concluya con la finalización anormal del programa (y por ende, con la finalización no deseada de la aplicación).

A continuación se pasan parámetros de carácter válido y se observa en la ejecución paso a paso los valores que van tomando cada una de las variables del programa y si son las esperadas o no, teniendo en cuenta el funcionamiento independiente de cada unidad de código.

5.1.2 Pruebas de componentes

Son aquellas pruebas de caja negra que comprueban la validez y verificación de unidades de código que funcionarán de forma conjunta en la aplicación y que por tanto están íntimamente ligadas, distinguiendo claramente su funcionalidad de aquella de otros componentes que no poseen relación con ésta [16].

En esta aplicación se somete a cada actividad y fragmento asociado (si es que lo posee), a una serie de comprobaciones en tiempo de ejecución, que sirven para cerciorarnos si se está obteniendo resultados esperados y si no se produce ningún tipo de error a la hora de hacer un uso normal o anormal de la aplicación.

Así, por ejemplo, sometemos a la actividad de Login a una serie de pruebas relacionadas con la introducción de datos para comprobar su validez, como la introducción de datos de un usuario que no existe, la no introducción de algún dato, o la introducción de algunos datos correctos y otros incorrectos, que arrojan un mensaje informativo concluyendo que los datos introducidos no son los esperados.

En la actividad de Register se procede de igual forma, introduciendo en este caso nombres de usuario que ya existen (mostrándonos un mensaje de error que alega que ese nombre de usuario ya está en uso), no introduciendo algún dato, así como correos electrónicos inválidos, que concluyen de igual forma mostrando un mensaje informativo en el que se dictamina que se ha producido algún tipo de error.

A la hora de subir un evento o comentario, se procede de igual forma a la hora de rellenar los datos requeridos por la aplicación.

El resto de actividades o unidades de código se prueban de forma separada observando en la ejecución paso a paso si se muestran los datos correctos, así como si al realizar cualquier tipo de acción se obtiene el resultado esperado y no cualquier error de carácter grave.

5.1.3 Pruebas de integración

Son aquellas pruebas de caja negra que se realizan una vez aprobadas las unitarias y de componentes, y lo que prueban es que todas las unidades de código independientes o componentes que componen el software, funcionen juntos correctamente probándolos en grupo [21].

En esta aplicación se somete a cada una de las actividades a pruebas en las que se verifica si los datos pasados de una actividad a otra son los esperados, si se produce algún tipo de error al utilizar alguna preferencia guardada en el sistema, o si al realizar cualquier tipo de acción en una actividad y ésta transita a otra o termina, no se produzca ningún tipo de fallo y se obtiene el resultado esperado.

Así, por ejemplo, se comprueba que tras realizar login, los datos se almacenan de forma correcta en las preferencias del usuario al navegar hasta la pantalla de perfil, se comprueba que tras subir un evento o comentario, éstos están disponibles para el administrador del sistema en la sección de pendientes, además de comprobar que al aceptar o rechazar un evento o comentario éste aparece o no aparece en el listado de eventos o comentarios

correspondiente. También se comprueba que tras pulsar un evento en la pantalla inicial de eventos, la pantalla del evento concreto posee los datos correspondientes al pulsado de forma correcta.

5.1.4 Pruebas de sistema

Son las pruebas de caja negra que comprueban la validez y verificación del sistema al completo, integrando todos los componentes de forma conjunta.

Así, en este caso se prueba a través de la ejecución de la aplicación, que en cada pantalla se realice la funcionalidad pertinente y esperada y se guarden datos que van a ser utilizados en cualquier módulo, así como se introducen datos de forma correcta en la base de datos y servidor, y se accede a éstos sin incidencias y de forma aceptable en tiempo de ejecución de la aplicación.

Por ejemplo, se comprueba que las preferencias del usuario se mantienen de forma correcta a lo largo de toda su sesión, o que los datos introducidos, actualizados o borrados de base de datos y servidor son consistentes atendiendo a las diversas funcionalidades que requieren de la interacción con éstos accediendo a los datos proporcionados y comprobando su validez.

5.2 Pruebas no funcionales

Son aquellas pruebas cuyo objetivo es la validez y verificación de un requisito que puede usarse para juzgar la operación de un sistema, atendiendo a diversas clasificaciones.

5.2.1 Pruebas de compatibilidad

Son aquellas pruebas que validan y verifican el funcionamiento del sistema en varios entornos [19].

En este caso, se ha probado a lanzar la aplicación en diversos dispositivos móviles, y comprobando que cada funcionalidad específica se lleva a cabo con normalidad y sin ningún tipo de incidencia.

Además se ha probado la aplicación en dispositivos móviles físicos y virtuales y en dispositivos de tipo tableta, obteniendo resultados y experiencia similar a los obtenidos en los demás, ofreciendo una aplicación versátil y multiplataforma.

5.2.2 Pruebas de usabilidad

Son pruebas definidas para evaluar un producto software a través de los propios usuarios, obteniendo información directa acerca de cómo usuarios reales interactúan con el sistema y qué respuestas concluyen [17].

En este caso, se experimenta a través de diversos usuarios que interactúan con la aplicación, y se obtiene la respuesta emocional de éstos ante la interacción con la misma, focalizándose así en los puntos a mejorar de la aplicación en cuanto a interacción con el usuario, qué tareas son menos intuitivas, qué aspectos no están claros o son menos sencillos de llevar a cabo dentro de la aplicación, ... que concluyen con una idea general acerca de los aspectos a mejorar y puntos donde centralizar la atención para optimizar la experiencia de usuario. Entre estos aspectos, merece la pena mencionar una interfaz de usuario con un tema más amigable visualmente, la presencia de un menú lateral en conjunción con el menú de la barra de acción en lugar de todo en uno mismo, o la utilización de la API de Google Maps para encontrar la localización exacta de un evento.

Por otro lado, se tiene en cuenta la opinión de diversos usuarios avanzados del sistema en forma de expertos, como son el tutor del presente trabajo y el profesor de Desarrollo de Aplicaciones para Dispositivos Móviles, que cuentan con gran experiencia en el sector, reafirmando aspectos que pueden ser mejorados u optimizados. Entre estos aspectos, se puede destacar la utilización de fragmentos que mejoran el rendimiento a nivel global, la utilización de bibliotecas de apoyo para conectar e interactuar con el servidor o la sustitución de almacenamiento total en una base de datos local del usuario, por almacenamiento de datos en el servidor.

5.2.3 Pruebas de escalabilidad

Son aquellas pruebas que permiten determinar el grado de escalabilidad de un sistema agregando cierta información al sistema y observando cómo éste responde [20].

En este caso, se procede a aumentar de forma progresiva el número de usuarios y eventos de forma que el sistema responde de forma menos rápida ya que al hacer una petición de más información del lado del servidor, ésta tarda más en llegar.

Por otro lado se ha comprobado que el rendimiento de la base de datos ante tal efecto es menor, por ejemplo, al marcar muchos eventos como favoritos, el rendimiento de acceso a la base de datos permanece casi constante, con lo que no disminuye apenas.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Tras la realización del presente trabajo se puede afirmar concluyendo que ha sido una experiencia muy enriquecedora, sobre todo en el aspecto académico, en el que se han planteado muchos problemas de diversos caracteres a medida que se ha ido asentando la idea, y que se han resuelto de forma completa obteniendo una formación y experiencia que convierten este tipo de proyectos de gran envergadura en una gran fuente de aprendizaje y conocimiento.

La estructura del proyecto se ha ido particularizando a lo largo de todo su ciclo de vida, formándose en una primera instancia como una simple idea, formalizándose más adelante a través de la fase de análisis donde se abstraían los requisitos necesarios para concretar dicho pensamiento inicial, pasando por una fase de diseño donde ya se dejaban entrever los mecanismos internos que iban a formar parte de la aplicación en la cual se implementaban diagramas que facilitaban de forma considerable el aspecto de codificación posterior, siguiendo con la fase de desarrollo como la más completa y a la que se ha dedicado más tiempo, con el objetivo de no dejar nada al libre albedrío, sino que todo tenga su por qué y su necesidad de realizarlo de la forma en la que finalmente se ha propuesto, y finalmente una fase de pruebas final que afirma con creces la validez y corrección del producto desarrollado.

Además, mencionar la seriedad dedicada en la realización del presente trabajo, en el que cada sección ha sido cuidada hasta el más mínimo detalle, pasando por una primera fase de reflexión acerca de qué se requería en cada sección, después desarrollando la idea cuidadosamente, para finalmente revisar hasta más de dos veces e ir corrigiendo cada uno de los aspectos que podrían ser mejorables u optimizados.

Recalcar la gran cantidad de esfuerzo dedicada no solo a la implementación de la idea en sí, sino también la atención prestada y dedicada a cada una de las partes que componen el proyecto, desde que se presentaba como una simple idea, hasta la consecución del más mínimo detalle, que desembocan en un producto software fiable y completo donde se pretende que todo aquel que haga uso de él obtenga una experiencia agradable e intuitiva.

6.2 Trabajo futuro

Como trabajo futuro se pretende seguir manteniendo los estándares de calidad que se han conseguido con el presente trabajo, pero optimizando aspectos de forma regular con el objetivo de que el usuario disfrute de una experiencia única a la hora de usar la aplicación. Así, se proponen diversos focos de mejora, que se detallan a continuación:

- Inicialmente, se pretende aumentar la seguridad de los datos almacenados en bases de datos y servidor, almacenando la información con algún tipo de codificación que disminuya la vulnerabilidad de los datos que utiliza la aplicación en caso de ataque, bien podría ser cifrando con algún tipo de algoritmo como MD5 o similar, o con algún otro tipo de solución.

- Se pretende también sustituir la base de datos local y el almacenamiento de información por una base de datos asociada al propio servidor, con el objetivo de dotar al sistema de más consistencia y flexibilidad a la hora de acceder y utilizar los datos. En principio no sería muy costoso al tener la base de datos ya implementada a través del sistema gestor PostgreSQL, tan sólo habría que modificar el tratamiento de las peticiones en el lado del servidor.
- Se pretende hacer uso del sistema de notificaciones de Android a la hora de la subida de un nuevo evento o comentario, de carácter informativo, y con el objetivo de hacer más profesional la interfaz de usuario.
- Se precisa de la utilización de otro host para el servidor al que conectar la aplicación, puesto que el actual quedará inutilizado a lo largo del tiempo, y para ello, se requiere de alguna máquina dispuesta a alojar dicho contenido.
- Se pretende aumentar funcionalidad específica de la aplicación como que el usuario pueda subir sus propias categorías, funcionalidad de eliminar comentarios o eventos específicos, ordenar por nombre los eventos, etc.
- Por último, seguir observando al usuario y considerar sus preferencias y solicitudes a la hora de interactuar con la aplicación, además de tener en cuenta consejos de mejora y prestar atención a la fase de mantenimiento, que desembocarán en tareas de optimización de la aplicación.

Referencias

- [1] About SQLite (sqlite.org), web.
<https://www.sqlite.org/about.html>
Accedido por última vez en Mayo 2017.
- [2] About the Eclipse Foundation (eclipse.org), web.
<https://eclipse.org/org/>
Accedido por última vez en Mayo 2017.
- [3] Android, iOS and Windows Phone compared in infographic (cnet.com), web.
<https://www.cnet.com/news/android-ios-and-windows-phone-compared-in-infographic/>
Accedido por última vez en Mayo 2017.
- [4] Android Studio o Eclipse, opinión de un desarrollador de aplicaciones (Todoandroid.com), web.
<http://www.todoandroid.es/index.php/faq-de-android/65-versiones/1698-android-studio-o-eclipse-opinion-de-un-desarrollador-de-aplicaciones.html>
Accedido por última vez en Mayo 2017.
- [5] Android Studio v1.0: características y comparativa con Eclipse (academiaandroid.com), web.
<http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/>
Accedido por última vez en Mayo 2017.
- [6] Android Studio IDE oficial para Android (developer.android.com), web.
<https://developer.android.com/studio/index.html?hl=es-419>
Accedido por última vez en Mayo 2017.
- [7] B4A – The simple way to develop native Android apps (Anywhere software), web.
<https://www.b4x.com/b4a.html>
Accedido por última vez en Mayo 2017.
- [8] Bill Phillips, Chris Stewart, Brian Hardy & Kristin Marsicano, “Android Programming the big nerd ranch guide 2nd edition”, Big Nerd Ranch, 2016.
- [9] Cliente servidor (wikipedia.org), web.
<https://es.wikipedia.org/wiki/Cliente-servidor>
Accedido por última vez en Mayo 2017.
- [10] Desarrollo de aplicaciones para dispositivos móviles. Tema 19. Moodle.uam.es.
Accesible vía Web.
https://moodle.uam.es/pluginfile.php/1218727/mod_label/intro/19.networking.pdf
Accedido por última vez en Mayo 2017.
- [11] Eventos UCM (ucm.es), web.
<https://www.ucm.es/apps>
Accedido por última vez en Mayo 2017.
- [12] Getting Started (dropwizard.io), web.
<http://www.dropwizard.io/1.1.0/docs/getting-started.html>
Accedido por última vez en Mayo 2017.
- [13] Google Maps para cada plataforma (developers.google.com), web.
<https://developers.google.com/maps/?hl=es-419>
Accedido por última vez en Mayo 2017.
- [14] iUPa - app (uma.es, web).
<http://www.uma.es/servicio-cultura/noticias/iupa-app/>
Accedido por última vez en Mayo 2017.

- [15] Modelo vista controlador wikipedia.org (wikipedia.org), web.
<https://es.wikipedia.org/wiki/Modelo%20%80%93vista%20%80%93controlador>
Accedido por última vez en Mayo 2017.
- [16] Prueba de componente (testingbaires.com), web.
<http://testingbaires.com/prueba-de-componente/>
Accedido por última vez en Mayo 2017.
- [17] Prueba de usabilidad (wikipedia.org), web.
https://es.wikipedia.org/wiki/Prueba_de_usabilidad
Accedido por última vez en Mayo 2017.
- [18] Prueba unitaria (wikipedia.org), web.
https://es.wikipedia.org/wiki/Prueba_unitaria
Accedido por última vez en Mayo 2017.
- [19] Pruebas de compatibilidad (wikipedia.org), web.
https://es.wikipedia.org/wiki/Pruebas_de_compatibilidad
Accedido por última vez en Mayo 2017.
- [20] Pruebas de escalabilidad (wikipedia.org), web.
https://es.wikipedia.org/wiki/Pruebas_de_escalabilidad
Accedido por última vez en Mayo 2017.
- [21] Pruebas de integración (wikipedia.org), web.
https://es.wikipedia.org/wiki/Pruebas_de_integraci%C3%B3n
Accedido por última vez en Mayo 2017.
- [22] Pruebas funcionales (wikipedia.org), web.
https://es.wikipedia.org/wiki/Pruebas_funcionales
Accedido por última vez en Mayo 2017.
- [23] Servicio web (wikipedia.org), web.
https://es.wikipedia.org/wiki/Servicio_web
Accedido por última vez en Mayo 2017.
- [24] The Java EE 6 Tutorial (docs.oracle.com), web.
<http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>
Accedido por última vez en Mayo 2017.
- [25] Versiones de la plataforma (developer.android.com), web.
<https://developer.android.com/about/dashboards/index.html>
Accedido por última vez en Mayo 2017.
- [26] Vida universitaria (universia.es), web.
<http://universitarios.universia.es/tiempo-libre/cultura-universidades/>
Accedido por última vez en Mayo 2017.

Glosario

API	Application Programming Interface (Interfaz de programación de aplicaciones), es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
BEAN	Son un modelo de componentes para la construcción de aplicaciones en Java que se usan para encapsular varios objetos en uno único para hacer uso de un solo objeto en lugar de varios más simples.
BIBLIOTECA	Es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.
FRAMEWORK	Entorno de trabajo o marco de trabajo, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
REST	Representational State Transfer (Transferencia de estado representacional) es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.
SGDB	Sistema gestor de bases de datos, es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos.
WWW	World Wide Web, es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet.

Anexos

A Manual de usuario



Logo de la aplicación JEvent

¡Bienvenido a JEvent! A través de esta aplicación ya podrás disponer de todos los eventos y actividades llevados a cabo en la Universidad Autónoma de Madrid, y enterarte de todas las novedades que se encuentran en cada momento dentro del campus, además de poder subir los tuyos propios.

Al iniciar la aplicación, se mostrará algo como esto:



Pantallas de carga e inicio de sesión

En esta primera pantalla de inicio de sesión, podremos entrar en la aplicación si introducimos nuestros datos de forma correcta, y pulsamos en el botón de “*Login*”. Podremos entrar como invitados pulsando en el botón “*Enter as visitor*”, pero en este caso dispondremos de funcionalidad limitada a los usuarios registrados.

Si deseamos crearnos una nueva cuenta para entrar en JEvent, pulsaremos en el texto “*Not account yet? Create a new one*”, con lo que se nos mostrará la pantalla de registro, que vemos a continuación:

Pantalla de registro

En esta pantalla, podremos crearnos una nueva cuenta en JEvent cumplimentando debidamente los datos que se solicitan, e introduciendo un e-mail válido. Una vez introducidos los datos, pulsaremos el botón “*Create account*”, hecho que nos redirigirá a la pantalla de inicio de sesión.

Una vez introducidos correctamente los datos de inicio de sesión, o entremos como invitados, accederemos a la pantalla principal de listado de eventos, que se muestra a continuación:



Pantalla principal de listado de eventos

Desde esta pantalla podremos visualizar todos los eventos incluidos en la aplicación, pulsando y arrastrando hacia abajo para ir mostrando los que inicialmente no aparecen. En cada evento se nos muestra su título, fecha, descripción, entidad organizadora y categoría (en forma de icono al lado de la descripción).

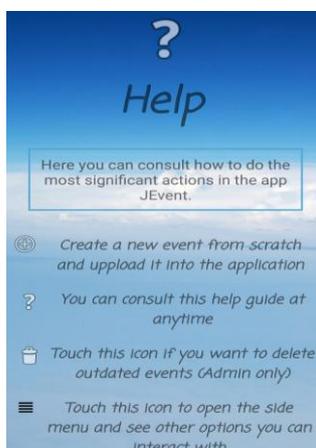
Si pulsamos en el icono  de la barra de acción, accederemos a la pantalla de subir evento, mostrada en la siguiente figura:



Pantalla de subir evento

En esta pantalla, se deberán rellenar los datos solicitados, ya que son todos de carácter obligatorio, pulsando en el botón de “Event date” para cambiar la fecha, en el botón “Select category” para cambiar la categoría y en el botón “Upload” para subirlo (NOTA: los eventos no se subirán directamente, sino que estarán pendientes de moderar por el administrador del sistema).

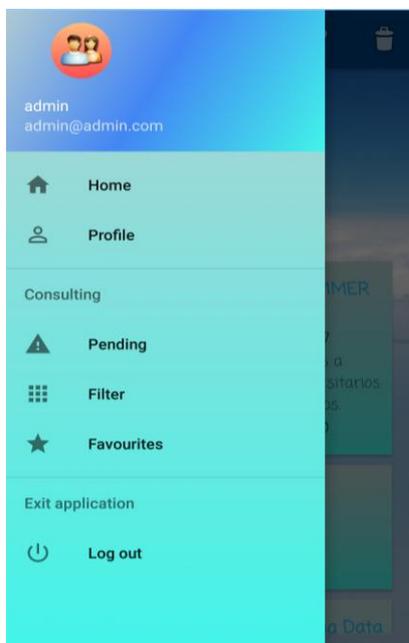
Desde la pantalla principal, también podremos acceder a la ayuda pulsando el icono  de la barra de acción, con lo que se nos mostraría lo siguiente:



Pantalla de ayuda

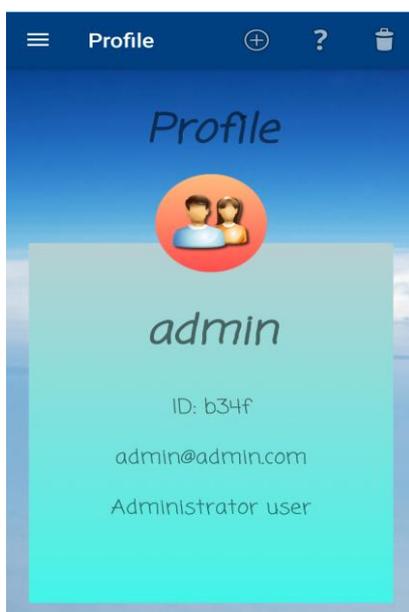
En la pantalla de ayuda podremos consultar las principales funcionalidades que se pueden llevar a cabo en JEvent.

Si presionamos el icono de menú  situado en la barra de acción en la pantalla principal, accederemos al menú lateral que se muestra a continuación:



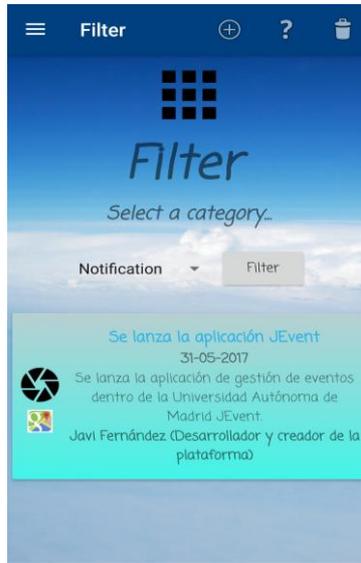
Menú lateral de JEvent

Desde este menú podremos acceder a la funcionalidad de consultar el perfil , donde se muestran los datos del usuario que ha iniciado sesión:



Pantalla de perfil

Desde el menú también se podrá acceder a la pantalla de eventos filtrados , donde podremos seleccionar una categoría para filtrar los eventos disponibles según ésta:



Pantalla de eventos filtrados

Desde el menú también se podrá acceder a la pantalla de favoritos , donde se listarán todos aquellos eventos marcados por el usuario como favoritos:



Pantalla de favoritos

Por último, desde el menú se podrá acceder nuevamente a la pantalla principal de listado de eventos pulsando el icono “Home” , consultar los eventos y comentarios pendientes en caso de ser administrador en “Pending” , o salir de la sesión a través del icono de “Log out” .

Si pulsamos cualquiera de los eventos situados en la pantalla principal de listado de eventos, accederemos a la pantalla de detalle de evento que se refleja a continuación:



Pantalla de detalle de evento

En esta pantalla se nos muestran de forma detallada los atributos del evento que hemos seleccionado, así como los comentarios que se han hecho sobre el mismo. Pulsando en el mapa accederemos a la localización exacta de realización del evento a través de Google Maps. Pulsando en el botón  daremos me gusta al evento, pulsando el botón  lo marcaremos como favorito, pulsando  en caso de ser administrador borraremos todos los comentarios del evento, y pulsando  accederemos a la pantalla de subir comentario que se muestra a continuación:



Pantalla de subir comentario

En esta pantalla, una vez rellenado el cuerpo del evento, se subirá el comentario a la lista de comentarios pendientes de moderación, a la que tan sólo el administrador podrá acceder para aceptar o rechazar el comentario.

B Análisis de requisitos

Requisitos funcionales

La aplicación soporta el actor siguiente:

Usuario, subdividido a su vez en:

Moderador del sistema (*)

- Registrarse en la aplicación como tal (**ID 01**)
- Hacer login en la aplicación como tal (**ID 02**)
- Consultar eventos pendientes de aceptación/rechazo (**ID 03**)
- Aceptar evento (**ID 04**)
- Rechazar evento (**ID 05**)
- Eliminar eventos caducados (**ID 06**)
- Aceptar comentario en evento (**ID 07**)
- Rechazar comentario en evento (**ID 08**)
- Eliminar usuario inactivo o reiteradamente ofensivo (**ID 09**)

Alumno académico perteneciente a institución (**)

- Registrarse en la aplicación (**ID 10**)
- Hacer login en la aplicación (**ID 11**)
- Consultar perfil (**ID 12**)
- Modificar perfil (**ID 13**)
- Subir evento (**ID 14**)
- Hacer comentario en evento (**ID 15**)
- Darse de baja en el sistema (**ID 16**)

Visitante invitado

- Consultar listado de eventos (**ID 17**)
- Filtrar por categoría el listado de eventos (**ID 18**)
- Obtener información de evento (**ID 19**)
- Clasificar evento (**ID 20**)
- Consultar eventos clasificados (**ID 21**)
- Hacer comentario en evento (**ID 22**)
- Compartir evento (**ID 23**)
- Buscar evento (**ID 24**)
- Marcar evento como favorito (**ID 25**)
- Consultar eventos favoritos (**ID 26**)

(*) El moderador del sistema podrá realizar todas las tareas que pueda realizar un visitante y alumno cualquiera de la aplicación; así, se tiene en cuenta que todos los

requisitos que atañen éstos se incluyen de forma implícita en los requisitos del moderador del sistema, a excepción de los dos primeros (**ID's 09 y 10**) y el (**ID 16**).

(**) El alumno académico podrá realizar todas las tareas que pueda realizar un visitante invitado cualquiera de la aplicación; así, se tiene en cuenta que todos los requisitos que atañen al visitante invitado se incluyen de forma implícita en los requisitos del alumno académico.

Requisitos no funcionales

Operacionales

- Posibilidad para el usuario (tanto alumnos como moderador del sistema) de recuperar sus datos de acceso (**ID 27**).
- [Opcional] Posibilidad para el moderador del sistema de recuperar eventos eliminados (**ID 28**).

Seguridad

- Cada alumno debe autenticarse antes de poder realizar las funcionalidades de subir un evento y hacer comentarios (**ID 29**).
- El moderador del sistema debe autenticarse antes de poder realizar las funcionalidades que le atañen (**ID 30**).
- Todos los comentarios realizados con respecto a un evento serán moderados por el moderador del sistema antes de estar visibles para el resto de los usuarios (**ID 31**).
- Todos los eventos que un alumno pretenda incorporar a la aplicación serán moderados y comprobados por el moderador del sistema (**ID 32**).
- Cada usuario registrado dispondrá de un nombre de perfil único y privado de su elección (**ID 33**).
- Cada usuario deberá registrarse con un correo académico válido; para comprobar esta validez se le enviará un correo de confirmación de registro (**ID 34**).

Mantenibilidad y portabilidad

- La aplicación estará disponible para el sistema operativo Android en esta primera versión (**ID 35**).

Recursos

- El sistema de almacenamiento se basará en un sistema de base de datos que guardará todos aquellos eventos que se vayan incluyendo en la aplicación, además de los nombres de usuario y sus perfiles (**ID 36**).
- La capacidad de almacenamiento será, en principio, indefinida, ya que los eventos se irán actualizando periódicamente y existirán usuarios que deseen darse de baja o sean eliminados por inactividad o comentarios ofensivos reiterados (**ID 37**).
- La longitud máxima de los comentarios será de 300 caracteres (**ID 38**).

- En la subida de eventos, la longitud máxima del título será de 50 caracteres y la longitud máxima de la descripción será de 1000 caracteres (**ID 39**).

Rendimiento

- El tiempo de respuesta será el mínimo razonable para cada actividad realizada sobre la aplicación (**ID 40**).

Interfaz y usabilidad

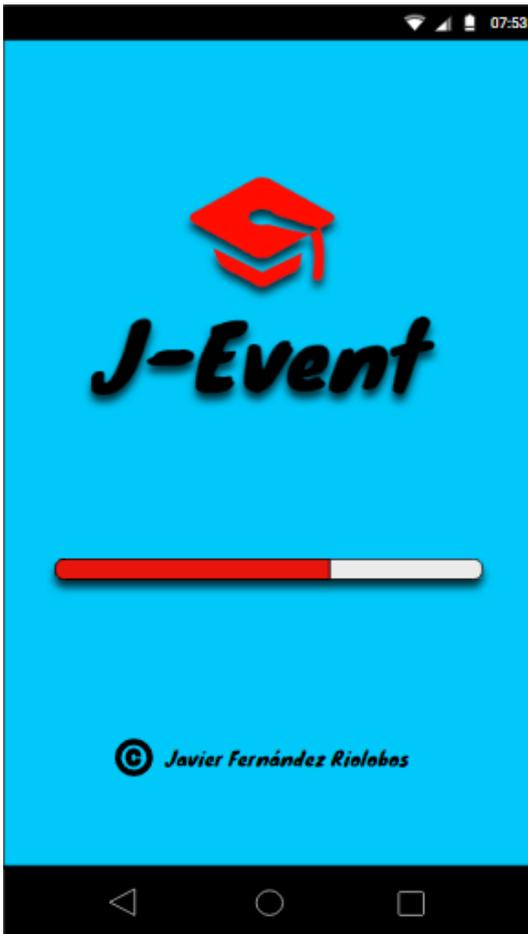
- La interfaz debe ser amigable para cada usuario que utiliza la aplicación y está basada en varias pantallas de fácil acceso e interconectadas de forma simple (**ID 41**).

Fiabilidad y verificación

- Si el alumno o el moderador se equivocase a la hora de autenticarse, se mostrará un mensaje de error, y se le permitirá volver a introducir sus datos de inicio de sesión (**ID 42**).
- En caso de no encontrar un evento a la hora de buscarlo, se imprimirá un aviso en el que figurará que el evento buscado no se encuentra en el sistema (**ID 43**).

C Maquetas de la aplicación

A continuación se presentan cuatro maquetas de la aplicación, que servirán para tener una idea acerca de la interfaz gráfica de usuario de J-Event, además de abarcar la funcionalidad anteriormente descrita en el apartado de análisis del ciclo de vida y en los apartados de objetivos y conclusión de aplicaciones similares del presente trabajo. Estas maquetas servirán para clarificar aspectos relacionados con la UI y con las tareas que desempeñará la aplicación, además de facilitar la labor en la fase de codificación e implementación posterior.



Maqueta 1: Pantalla inicial



Maqueta 2: Pantalla de eventos inicial

En estas dos primeras maquetas se muestran la pantalla inicial de carga que se nos muestra al iniciar la aplicación, y la pantalla de eventos inicial una vez que la carga ha finalizado, mostrando un listado de los eventos de forma genérica, a los que podremos acceder tocando en la actividad correspondiente. Además podremos deslizarlos por todos los eventos a través de la barra de scrolling, ordenar la visualización de eventos por categorías o consultar los eventos favoritos, y buscarlos por palabras clave. Por último se nos muestra el icono de usuario y si desea consultar su perfil o hacer log out.



Maqueta 3: Pantalla de categorías

Maqueta 4: Pantalla de evento

Estas dos últimas maquetas representan la pantalla de visualización de eventos ordenados por categorías mostrada en la Maqueta 2, en la que se podrán visualizar los eventos de acuerdo a la facultad o entidad que los organiza, la ubicación en la que se encuentren, la temática de la que traten (conferencia, formación, excursión, fiesta...) y los más populares o los que tengan más “Me gusta”, y una pantalla de evento a la que se accede tras tocar un evento cualquiera. En esta pantalla (Maqueta 4) se nos muestra la ubicación donde se realiza el evento, la temática a la que pertenece, la facultad o entidad que lo organiza y los “Me gusta” que tiene. Además contiene una descripción detallada del evento, comentarios de los usuarios y dos botones que nos muestran más información (como las fechas exactas o enlaces a páginas web que contengan más información) y el proceso de inscripción.

D Diseño gráfico de actividades y flujo entre ellas

En este anexo se muestran las diversas actividades que constituyen la aplicación, así como el flujo entre ellas a lo largo de la utilización de *JEvent* por parte del usuario, y en qué actividades se accede a la base de datos o se conecta al servidor.

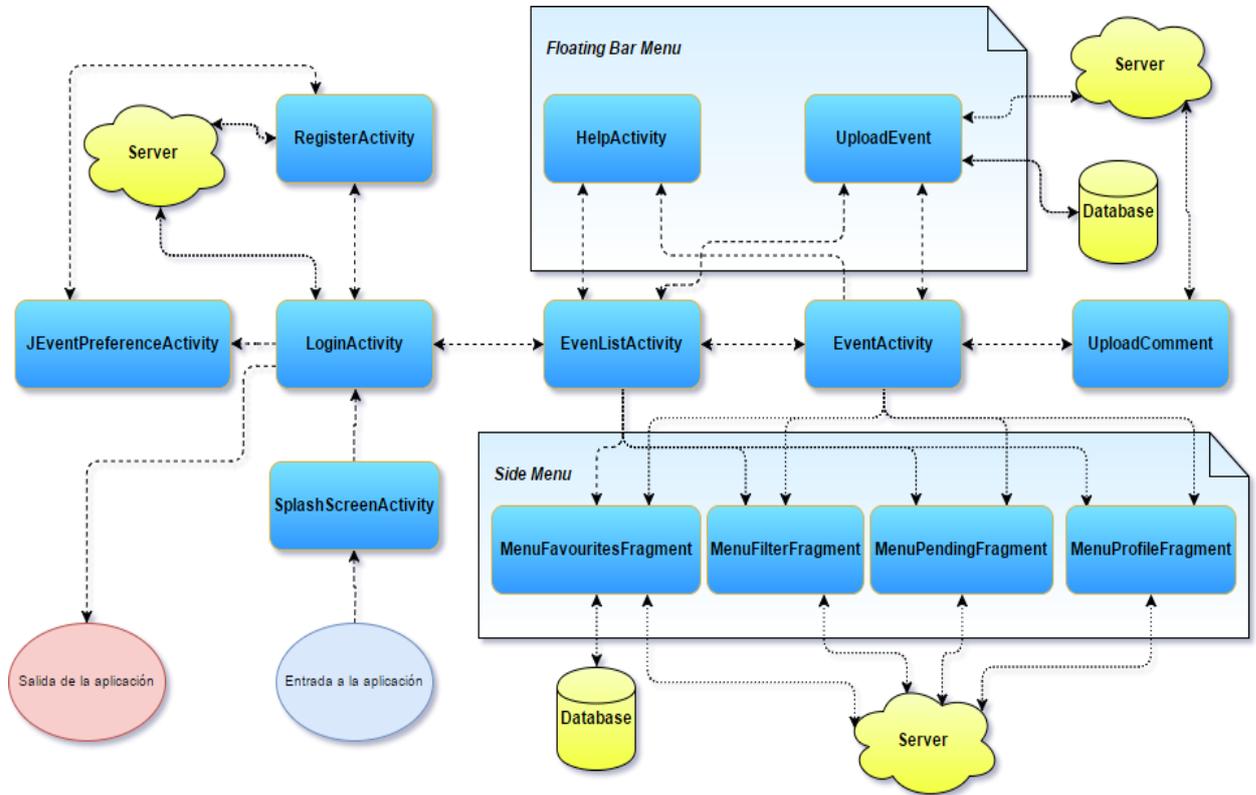


Diagrama de actividades y sus flujos