Eventizer.

Nombre Estudiante: Fernando Álvarez Crespo.
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles.

Nombre Consultor: Francesc D'Assís Giralt Queralt.

Fecha de entrega: 06/06/2018.

FICHA DEL TRABAJO FINAL.

Título del trabajo:	*Eventizer.*
Nombre del autor:	*Fernando Álvarez Crespo.*
Nombre del consultor:	Francesc D'Assís Giralt Queralt.
Fecha de entrega:	06/06/2018
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles.*

Resumen del Trabajo (máximo 250 palabras):

En este trabajo final de máster se propone la creación de una aplicación para dispositivos Android cuyo objetivo reside en facilitar la creación y organización de eventos grupales.

Eventizer permitirá a los usuarios crear eventos, compartirlos con amigos y poder discutir detalles organizativos de los mismos, mediante un chat con opciones de moderación de mensajes asociados a cada evento.

Por último, la aplicación permitirá exportar estos eventos a Google Calendar.

Abstract (in English, 250 words or less):

The main purpose of this master's thesis is to develop an application for Android devices, whose goal is to facilitate the creation and organization of group events.

Eventizer will allow users to create events, share them with friends and be able to discuss organizational details of them through a chat with message moderation options associated with each event.

Finally the application will allow you to export these events to Google Calendar.

Palabras clave (entre 4 y 8):

APP, Evento, Grupo, Organizar, Compartir

Índice.

Lista de figuras.

1. Introducción.

1.1 Contexto y justificación del Trabajo.

La idea de este TFM surge a partir de la observación de la tendencia de las personas de crear grupos de WhatsApp para organizar eventos.

Esta tendencia tiene una limitación muy importante: Una vez que los detalles del evento están cerrados y reflejados en forma de uno o varios mensajes en un grupo de WhatsApp, resulta muy poco intuitivo y lento acceder a esos mensajes “clave”. Esto se complica todavía más cuando entra gente interesada en el grupo después de que se escribieron los mensajes que reflejan los aspectos definitivos del evento. Los integrantes del grupo se ven obligados a copiar y reenviar los mensajes importantes para que los nuevos miembros puedan acceder a dicha información.

Eventizer permite a los usuarios crear eventos y compartirlos con distintos contactos. Cada evento tiene un chat asociado donde los integrantes pueden discutir los pormenores de los eventos, pero con una sutil diferencia respecto a WhatsApp, el administrador o administradores del evento pueden poner chinchetas a los mensajes más representativos para que aparezcan en la ficha del evento y eliminar aquellos mensajes que no son relevantes o que pueden llevar a equívocos. También es importante recalcar que los eventos de Eventizer son exportables a la aplicación de calendario por excelencia en Android: Google Calendar, instalada por defecto en numerosos dispositivos.

1.1.1 Estudio de mercado.

El siguiente paso después de plantear la idea inicial del TFM, es realizar una búsqueda de aplicaciones similares en la Google Play Store. Las aplicaciones que se pueden encuentran que más se parecen a la idea inicial son Looping [1] y TimeTree [2].

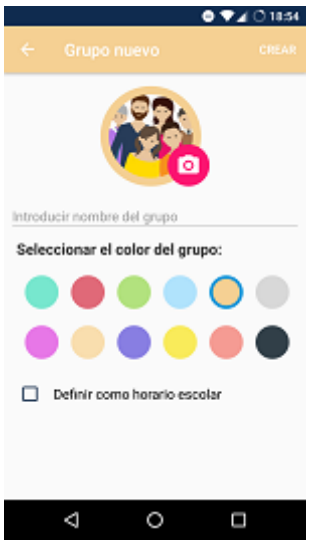

Ilustración 1: Captura de Looping

Ilustración 2: Captura de TimeTree

Características de las dos aplicaciones analizadas:

Las dos aplicaciones nos permiten mantener varios calendarios compartidos de eventos con chat integrado.

Desventajas respecto a Eventizer:

- Las dos aplicaciones buscan substituir cualquier aplicación de calendario por completo, ya que sus eventos no son exportables a otras aplicaciones.
- Sólo permiten el login social usando Facebook. Eventizer permite a mayores loguearse con Google y Twitter.
- Los chats asociados a los eventos no permiten moderación ni resaltado de mensajes importantes.

1.2 Objetivos del Trabajo.

El objetivo de este TFM es crear una aplicación con las siguientes características:

- Login social a través de Facebook, Twitter y Google. A mayores, poder registrarse usando un correo electrónico y nombre de usuario.
- Poder crear eventos compartidos y poder exportarlos a Google Calendar.
- Disponer de un chat con funciones de moderación y resaltado de mensajes importantes asociado a cada evento.

1.3 Enfoque y método seguido.

1.3.1 Plataforma de desarrollo.

El desarrollo de Eventizer se realiza en Android Studio usando Kotlin. Este TFM es una oportunidad muy interesante para probar este nuevo lenguaje de programación creado por Jetbrains orientado a la JVM, ya que durante las distintas PEC's de las asignaturas de Android, usé Java en el 100% de los casos.

1.3.2 Estrategia.

Para la consecución de este proyecto, se tiene que crear de cero una aplicación nativa de Android con las características expuestas anteriormente.

1.3.3 Metodología.

En un primer momento, se realiza una preselección de 2 metodologías de ciclo de vida de proyectos y a partir de ahí, se elige la que mejor se adapta a este proyecto en particular.

Cascada:
Este modelo se caracteriza por ordenar rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior [3].

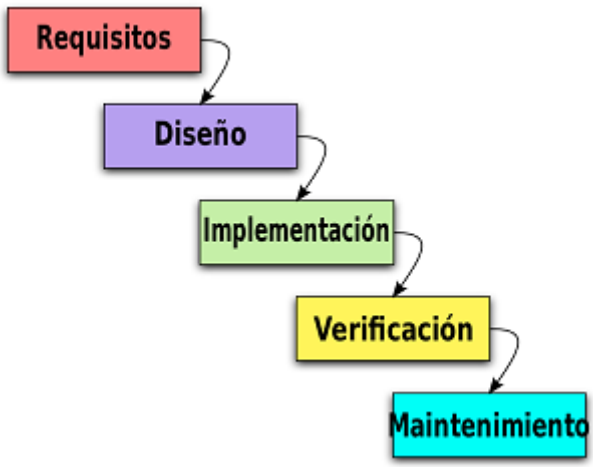

Ilustración 3: Desarrollo en cascada

Incremental:
Este modelo se caracteriza en definir n incrementos, donde se dividen en cada uno, las funcionalidades/requerimientos del proyecto. En cada incremento se realizan las mismas etapas que en el ciclo de vida en cascada. Este modelo de ciclo de vida es muy adecuado en entornos donde los requisitos no están claros desde un principio [4].

La metodología o modelo de ciclo de vida elegido para este proyecto es el desarrollo en cascada.

La elección viene justificada por las características organizativas del TFM, atendiendo a la disposición de las distintas PEC que coinciden con las fases de esta metodología y porque los requisitos están claros desde un primer momento.

1.4 Planificación del Trabajo.

1.4.1 Recursos.

Para la realización de este TFM, cuento con los siguientes recursos hardware:

Un ordenador de sobremesa con un procesador Intel I5 4690K, 16 GB de memoria RAM y W10 de 64 bits, como equipo de desarrollo

Un smartphone Nexus 5 con Android 7.1.2 para la realización de pruebas.

Para la realización de este TFM, cuento con los siguientes recursos software:

Microsoft Word 2016 para la realización de la memoria.

GanttProject para la realización del diagrama de Gantt.

Balsamic Mockups 3 para la realización de los prototipos.

StarUml para la realización de los distintos diagramas de UML presentes en este proyecto.

1.4.2 Planificación temporal.

A continuación, se detalla mediante un diagrama de Gantt, la planificación temporal de este TFM.

Nombre	Fecha de inicio	Fecha de fin	Costo
PEC 1 - Plan de trabajo	21/02/18	14/03/18	26
Selección de tema	21/02/18	27/02/18	15
Definición de objetivos	28/02/18	5/03/18	4
Definición del enfoque y método de trabajo	6/03/18	9/03/18	2
Realización de la planificación	10/03/18	13/03/18	4
Entrega PEC1	14/03/18	14/03/18	1
PEC 2 - Diseño	15/03/18	4/04/18	52
Análisis de requisitos	15/03/18	18/03/18	8
Diseño arquitectura	19/03/18	22/03/18	8
Diseño conceptual	23/03/18	28/03/18	15
Diseño interfaz usuario	29/03/18	3/04/18	20
Entrega PEC2	4/04/18	4/04/18	1
PEC 3 - Implementación	5/04/18	16/05/18	106
Desarrollo de la estructura e interfaz gráfica	5/04/18	19/04/18	30
Desarrollo funcionalidad	20/04/18	9/05/18	60
Pruebas y depuración	10/05/18	15/05/18	15
Entrega PEC3	16/05/18	16/05/18	1
Entrega Final	17/05/18	6/06/18	52
Revisión memoria	17/05/18	23/05/18	15
Creación presentación	24/05/18	1/06/18	30
Creación vídeo aplicación	2/06/18	5/06/18	6
Entrega Final	6/06/18	6/06/18	1

Ilustración 4: Tareas diagrama Gantt

La planificación temporal consta de un total de 236 horas a dividir en 106 días que constituyen la duración de los distintos plazos de las PECS del TFM. Esta división resulta en una media de 2,23 horas cada día si contamos los festivos y fines de semana como días normales de trabajo.

1.5 Breve sumario de productos obtenidos.

Al Entregar este TFM se incluyen los siguientes elementos:

- Esta memoria.
- El código Fuente de la aplicación desarrollada.
- La APK de la aplicación desarrollada.
- Un vídeo de la presentación del proyecto.
- Un vídeo demostrando el funcionamiento de la aplicación.
- Archivo con el contenido de la presentación.
- Pruebas: Este capítulo recoge un resumen de las pruebas realizadas y su resultado.
- Conclusiones: En este capítulo se intenta resumir y valorar lo conseguido durante la realización de este TFM. Se plantean algunas funcionalidades de ampliación de las características básicas cubiertas en este TFM.

2. Análisis.

En este capítulo se analizan los requisitos funcionales de la aplicación. Primeramente, se especifican los actores de la aplicación, para posteriormente, plasmarlos en los diagramas con sus casos de uso asociados.

2.1 Identificación de los usuarios y sus contextos de uso.

En esta sección se identifican los dos grupos de usuarios a los que va destinada la aplicación, junto con fichas de un usuario de cada tipo. Por último, se definen dos escenarios donde se muestra la utilidad de la aplicación.

La aplicación está dirigida a aquellos usuarios que habitualmente usan aplicaciones de calendario y de mensajería de forma separada y que pueden encontrar útil tener relacionados eventos de calendario, compartidos con otros usuarios, junto con mensajes de los participantes en dicho evento en una sola aplicación y además proporciona funcionalidades de ubicación.

Cualquier usuario con un manejo aceptable de aplicaciones muy comunes como Whatsapp o Telegram y Google Calendar no tendrá ningún problema en desenvolverse con soltura en las funcionalidades propuestas por Eventizer.

Fichas de personas y escenarios.

Nombre: Tomás Tortosa Yáñez
Edad: 29 años
Profesión: Trabajador de banca
Lugar de residencia: Barcelona

Nombre: Noelia Cruz Íñigo
Edad: 30 años
Profesión: Programadora web
Lugar de residencia: A Coruña

Escenario 1

Tomás y Noelia se conocen desde la adolescencia, eran integrantes del mismo grupo de amigos con el que realizaban juntos todo tipo de actividades de ocio.
Por motivos laborales, Tomás abandonó su localidad natal, A Coruña, para irse a trabajar a Barcelona. En este verano de 2018, Noelia quiere organizar una comida o una cena en un nuevo restaurante de moda, próximo a A Coruña, para reunir a todo su grupo de amigos de la adolescencia, ya que hace mucho que no coinciden todos juntos debido a que están viviendo en diversas ciudades de España y del extranjero.

Noelia opta por usar Eventizer para crear este evento. Tras invitar a todos los asistentes, en el chat del evento discuten sobre el día más idóneo para el mismo, intentando que le coincida bien al mayor número de amigos.
También deciden que es mejor hacer una cena para facilitar la asistencia a las personas que tienen que realizar un viaje de una duración considerable.

Escenario 2

El día del evento, cuando Tomás se está aproximando a la provincia de A Coruña, se da cuenta de que no conoce cuál es el itinerario óptimo que tiene que seguir para llegar al restaurante. Esto no le va a suponer un problema, ya que, en la ficha del evento en Eventizer se encuentra un marcador de Google Maps con las coordenadas del restaurante. También se incluye un botón para abrir la aplicación Google Maps para obtener las indicaciones para llegar a su destino.

2.2 Identificación de actores.

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad.

Tras una evaluación de los dos escenarios desarrollados anteriormente, se pueden apreciar dos actores distintos.

- Un usuario administrador (Noelia), que cuenta con la capacidad de crear eventos, invitar a los contactos que considere oportunos y moderar mensajes, esto es, poder eliminar posibles mensajes que ya no aportan nada a un evento o resaltar alguno sobre los demás. También puede modificar la ficha de cualquier evento creado.

- Un usuario "normal" (Tomás), que sólo puede acceder a la ficha de un evento, leer y participar en el chat asociado. Este actor también puede añadir usuarios de la aplicación a su lista de contactos para posteriormente poder añadir integrantes a un evento cuando actúa como usuario administrador.

Cabe resaltar que, por defecto, cuando se crea un evento, el propio usuario creador pasa a ser administrador de ese evento.

2.3 Diagrama y descripción de casos de uso.

Una vez identificados los actores, intentamos encontrar los casos de uso que modelan la funcionalidad de nuestra aplicación [5].

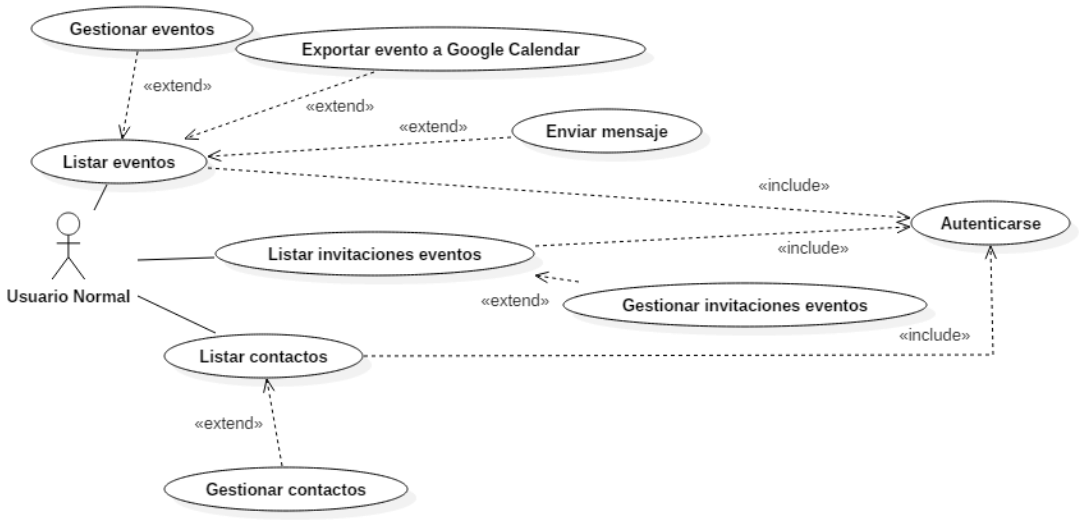

Ilustración 5: Diagrama casos de uso usuario normal

Caso de uso:	Autenticarse.
Precondición:	Ninguna.
Descripción:	El usuario utiliza unas credenciales válidas (correo/contraseña) propias de Eventizer o utiliza alguno de los logins sociales disponibles: Google, Twitter o Facebook.
Postcondición:	El usuario accede a la aplicación.

Caso de uso:	Listar invitaciones de eventos.
Precondición:	Estar autenticado en la aplicación.
Descripción:	Al usuario le aparecen las invitaciones a eventos a los que ha sido invitado.
Postcondición:	Ninguna.

Caso de uso:	Gestionar invitaciones a eventos.
Precondición:	El usuario tiene que tener alguna invitación a un evento que todavía no haya aceptado/rechazado.
Descripción:	El usuario acepta o rechaza alguna de las invitaciones a un evento.
Postcondición:	-Si rechaza la solicitud: Esta desaparece de la lista de solicitudes -Si acepta la invitación a un evento, este aparecerá en la lista de eventos del usuario y desaparece de la lista de solicitudes.

Caso de uso:	Listar contactos.
Precondición:	Estar autenticado en la aplicación.
Descripción:	Al usuario le aparecen, listados por orden alfabético, los contactos que tiene en su lista.
Postcondición:	Ninguna.

Caso de uso:	Gestionar contactos.
Precondición:	El usuario tiene que tener algún contacto en su lista de contactos.
Descripción:	El usuario puede eliminar de su lista a algún usuario agregado previamente o añadir uno nuevo.
Postcondición:	-En el primer caso, el contacto desaparece de la lista. -En el segundo caso, el nuevo contacto aparece reflejado en la lista.

Caso de uso:	Listar eventos.
Precondición:	Estar autenticado en la aplicación.
Descripción:	Al usuario le aparecen listados todos los eventos de los que forma parte.
Postcondición:	Ninguno.

Caso de uso:	Gestionar eventos.
Precondición:	El usuario tiene que tener algún evento en su lista de eventos.
Descripción:	El usuario puede eliminar de su lista algún evento que ya no le interese o crear uno nuevo. Para ello, tendrá que introducir un título, una descripción, una fecha y una dirección GPS.
Postcondición:	-En el primer caso, el evento desaparece de la lista. -En el segundo caso, el nuevo evento aparece reflejado en la lista.

Caso de uso:	Exportar evento a Google Calendar.
Precondición:	El usuario tiene que tener algún evento en su lista de eventos.
Descripción:	El usuario exporta un evento de su lista a Google Calendar.
Postcondición:	El evento aparece reflejado en su APP de Google Calendar.

Caso de uso:	Enviar mensaje.
Precondición:	El usuario tiene que tener algún evento en su lista de eventos.
Descripción:	El usuario selecciona el chat de un evento y escribe un mensaje.
Postcondición:	El mensaje aparece reflejado en el chat del evento en cuestión.

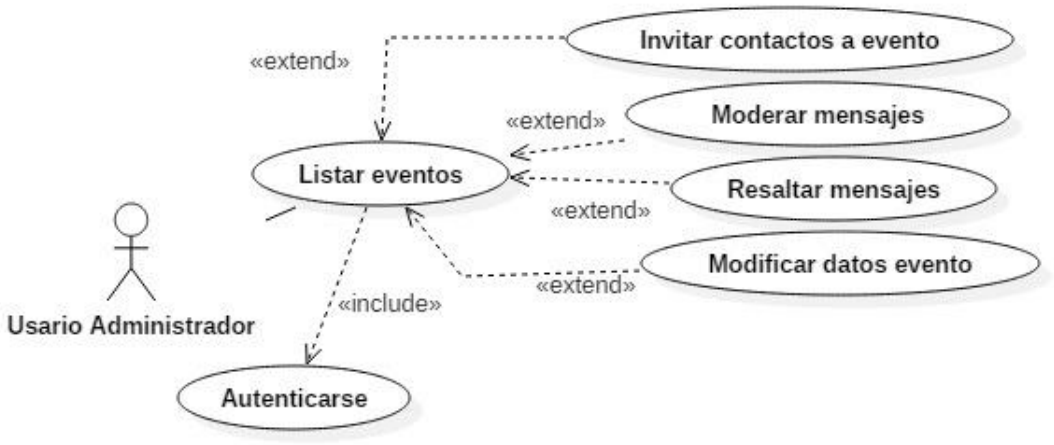

Ilustración 6: Diagrama de casos de uso usuario administrador

Caso de uso:	Invitar contactos a evento.
Precondición:	El usuario es administrador del evento en el que quiere invitar a sus contactos.
Descripción:	El usuario administrador elige a uno o más contactos de su lista.
Postcondición:	Los contactos seleccionados reciben una invitación al evento seleccionado por el administrador.

Caso de uso:	Moderar mensajes.
Precondición:	La existencia de mensajes en el chat de un determinado evento elegido por el usuario administrador.
Descripción:	El usuario administrador puede eliminar mensajes antiguos del chat del evento que considere que puedan generar “ruido” o alguno que considere poco apropiado.
Postcondición:	Los mensajes desaparecen del chat y ningún usuario podrá leerlos.

Caso de uso:	Resaltar mensajes.
Precondición:	La existencia de mensajes en el chat de un determinado evento elegido por el usuario administrador.
Descripción:	El usuario administrador puede resaltar ciertos mensajes para que aparezcan en la ficha principal del evento.
Postcondición:	Los mensajes resaltados por el usuario administrador aparecen en la ficha principal del evento.

Caso de uso:	Modificar datos evento.
Precondición:	Haber creado algún evento.
Descripción:	El administrador cubre un formulario con los nuevos datos del evento que desea cambiar.
Postcondición:	Los nuevos datos introducidos quedan guardados en el evento.

3. Diseño.

3.1 Diseño arquitectónico.

A la hora de abordar la elección del patrón de arquitectura para Eventizer, se valoraron dos alternativas:

MVC (Modelo vista controlador)
Este patrón de arquitectura cuenta con tres capas con distintas responsabilidades:

Modelo: la capa de datos responsable de gestionar la lógica empresarial y la comunicación con las capas de red y bases de datos.
Vista: La capa de interfaz de usuario, es decir, una visualización de los datos del modelo.
Controlador: La capa lógica que recibe notificaciones del comportamiento del usuario y actualiza el Modelo según sea necesario.

MVP (Modelo Vista Presentador).
Este patrón de arquitectura tiene también tres capas con distintas responsabilidades:

Modelo: La capa de datos responsable de gestionar la lógica empresarial y la comunicación con las capas de red y bases de datos.
Vista: La capa de interfaz de usuario muestra los datos y notifica al presentador sobre las acciones del usuario.

Presentador: Recupera los datos del modelo, aplica la lógica de la interfaz de usuario y gestiona el estado de la vista, decide qué mostrar y reacciona a los eventos de entrada del usuario desde la vista.

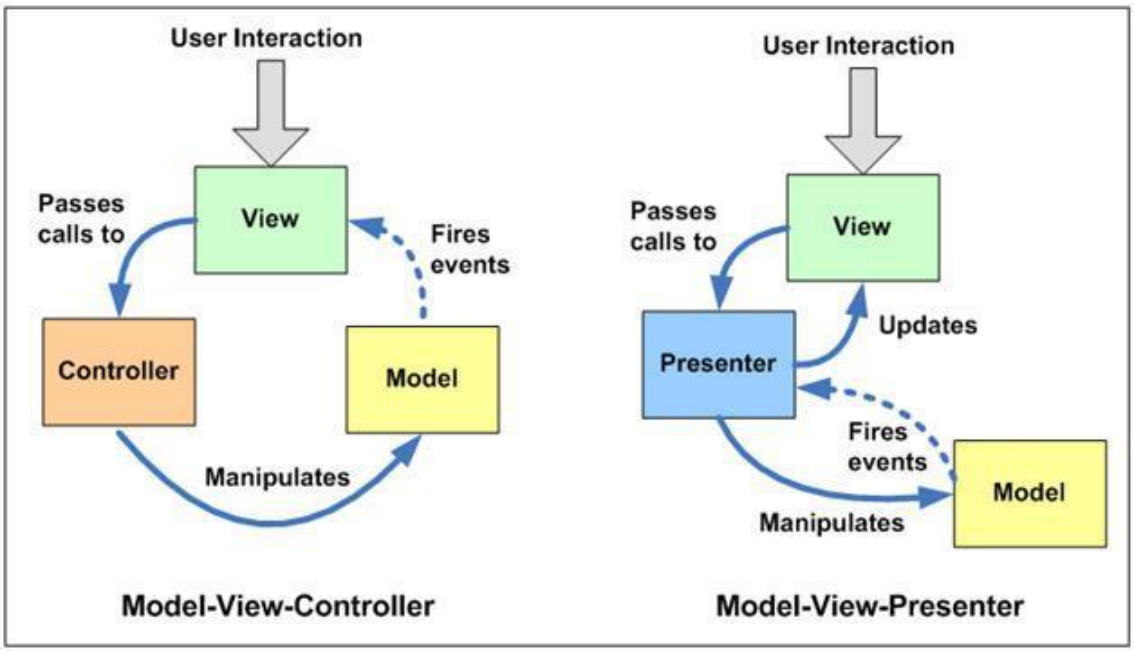

Ilustración 7: MVC vs MVP

Como se puede observar, la mayor diferencia radica en que, en la arquitectura MVP, la vista está completamente desacoplada del modelo, (los cambios siempre tienen que pasar por el presentador). Este es el principal motivo de la elección de la arquitectura MVP para Eventizer [6] [7].

3.2 Backend.

Como backend para la aplicación se elige Firebase [8]. Un servicio de Google que ofrece muchas funcionalidades desde el minuto cero, muy útiles para desarrollos móviles: base de datos en tiempo real, login, notificaciones…
Las funcionalidades de Firebase utilizadas en Eventizer, se comentan de una forma más extensa en el capítulo 4.

Esta elección viene justificada por el poco tiempo disponible para desarrollar un backend propio dentro de la ventana temporal de la PEC3.

3.3 Prototipos de las interfaces.

A continuación, se repasan los prototipos de las pantallas representativas de la aplicación:

1. Pantalla identificación/login.

Ilustración 8: Pantalla login

En esta pantalla el usuario podrá introducir sus credenciales de Eventizer, usar el login social a través de Google, Facebook o Twitter o registrarse en la aplicación.

2. Pantalla registro.

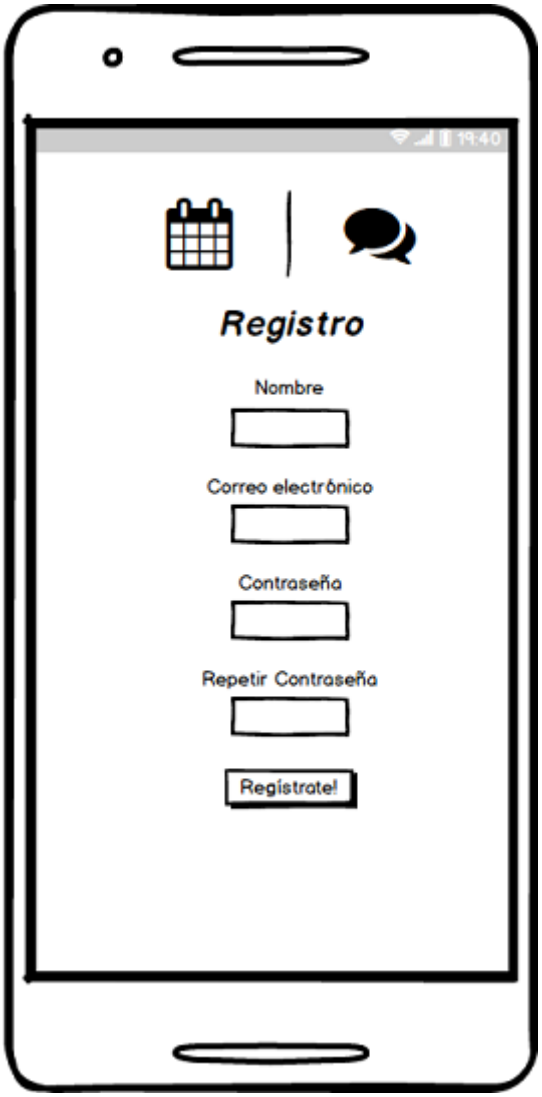

Ilustración 9: Pantalla registro

En esta pantalla el usuario podrá registrarse en la aplicación introduciendo un nombre de usuario, su correo electrónico y una contraseña.

3. Pantalla listado invitaciones eventos.

Ilustración 10: Pantalla invitaciones eventos

En esta pantalla el usuario podrá visualizar las invitaciones a eventos de los que todavía no forma parte.

4. Pantalla listado eventos.

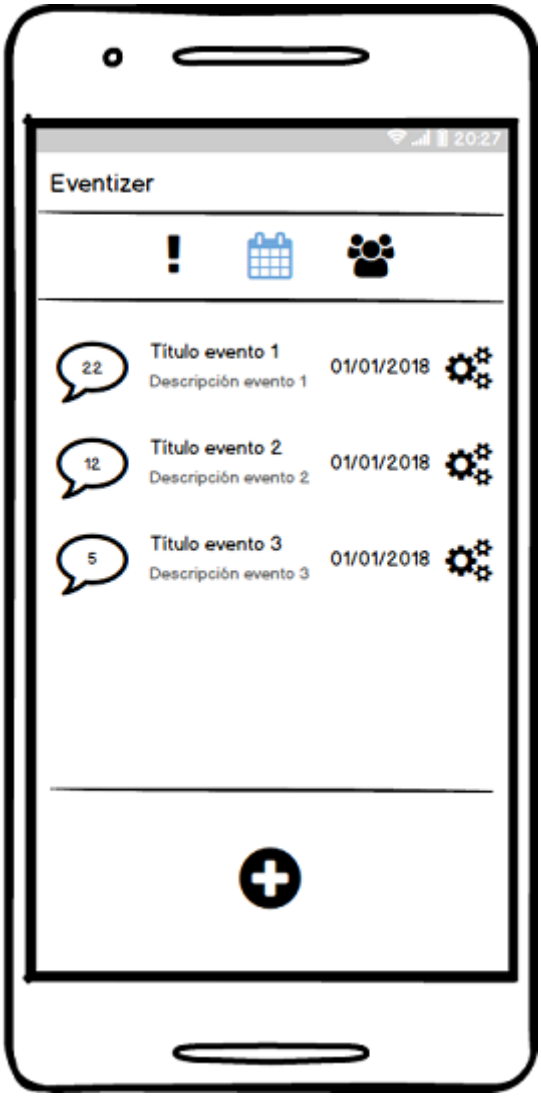

Ilustración 11: Pantalla listado eventos

En esta pantalla el usuario podrá visualizar todos los eventos de los que forma parte, visualizando el número de mensajes sin leer en cada uno de ellos. Pulsando en la rueda de opciones, el usuario podrá eliminar el evento o exportar el evento a Google Calendar.
Pulsando en la parte del título, el usuario accederá a la ficha de detalle del evento.
Usando el botón de la parte inferior, el usuario podrá crear un nuevo evento.

5. Pantalla listado contactos.

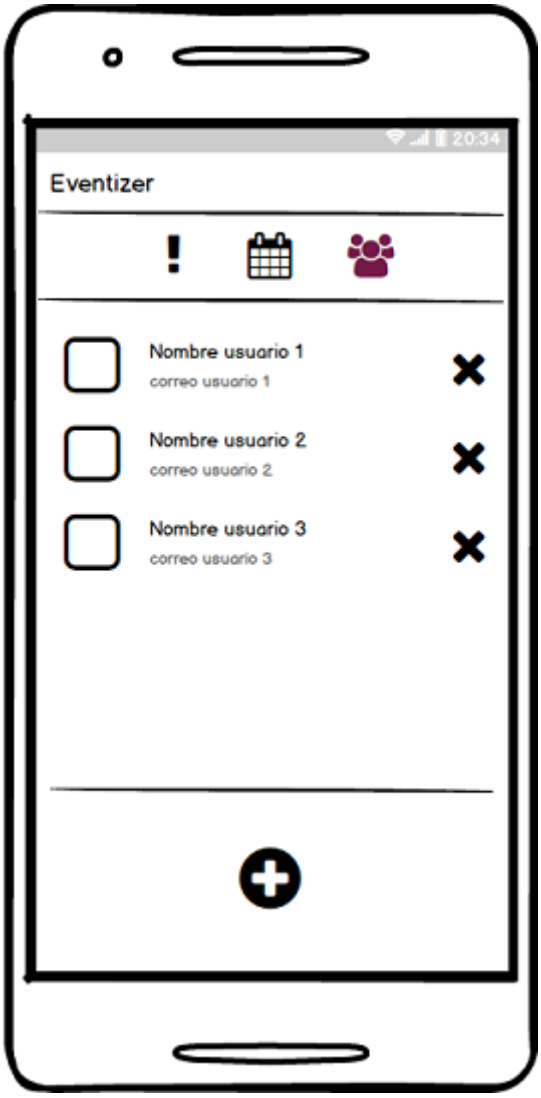

Ilustración 12: Pantalla listado contactos

En esta pantalla el usuario tendrá su libreta de contactos de Eventizer.

6. Pantalla detalle evento.

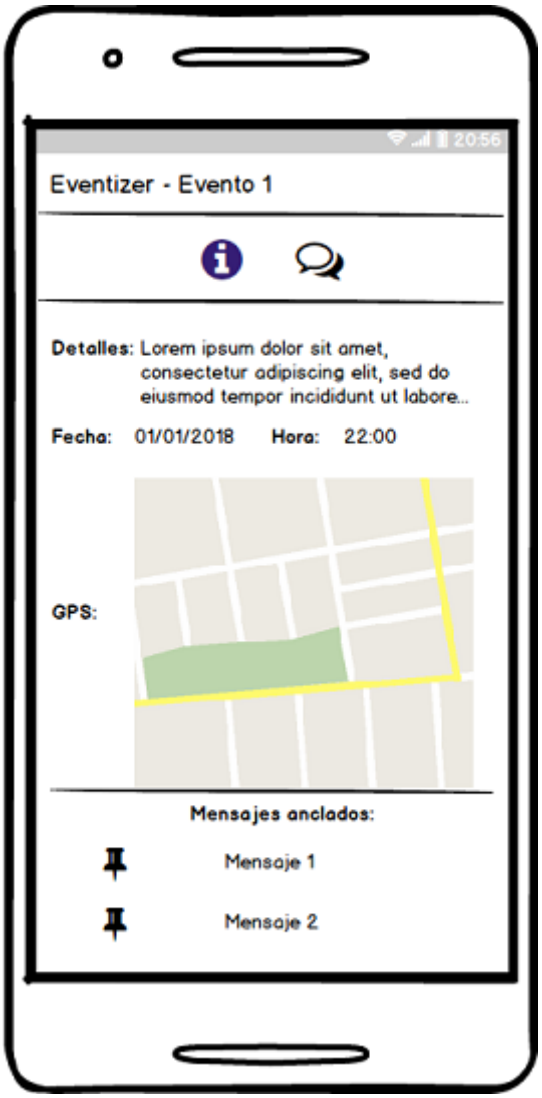

Ilustración 13: Pantalla detalle evento

En esta pantalla el usuario podrá ver la ficha completa de un evento: detalles, fecha, hora, coordenadas GPS y mensajes anclados.

7. Pantalla de chat.

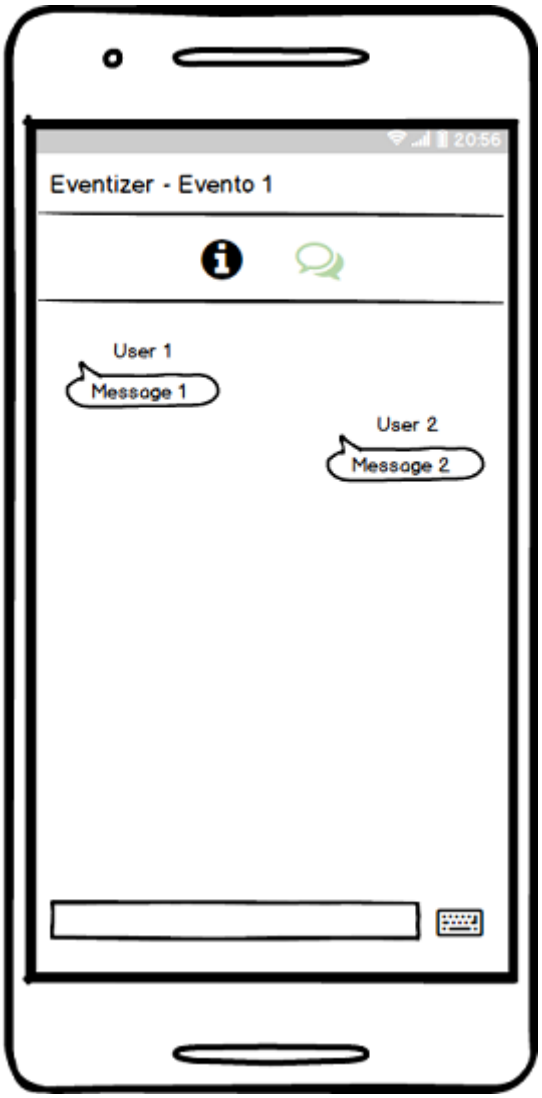

Ilustración 14: Pantalla chat

En esta pantalla el usuario podrá enviar mensajes a los usuarios que pertenezcan a un evento en cuestión.

3.4 Evaluación de los prototipos.

Por último, es necesario definir el método de evaluación de los prototipos definidos anteriormente.

Para ello se facilita la aplicación en su fase final y una breve encuesta a varios conocidos con dispositivos Android.

Se les insta a realizar las siguientes interacciones con la aplicación:

- Registrarse en la aplicación.
- Añadir contacto.
- Crear evento.
- Invitar contacto.
- Escribir mensajes en el chat asociado al evento.

Se pide la realización de la siguiente encuesta:

1- ¿Has podido realizar todas las acciones propuestas de manera satisfactoria? En caso negativo, indica cuáles no.

2- ¿El diseño de la interfaz gráfica favorece una navegación ágil entre las distintas pantallas/funcionalidades presentes en la aplicación?
(Puntúa de 1 a 5 la afirmación anterior, donde 1 es muy en desacuerdo y 5 es muy de acuerdo).

3- ¿El diseño resulta agradable a la vista?
(Puntúa de 1 a 5 la afirmación anterior, donde 1 es muy en desacuerdo y 5 es muy de acuerdo).

4 Implementación.

4.1 Entorno de desarrollo.

Eventizer es una app nativa para Android, escrita en Kotlin. De entre todos los IDES para desarrollar proyectos de Android que hay disponibles, (Eclipse, Netbeans, Android Studio…) se elige este último ya que es el IDE oficial establecido por Google. Se puede descargar de manera gratuita y consultar su documentación oficial en [9].

Otro aspecto a destacar es la elección de la versión mínima de API de Android para el proyecto. La API 14 (Ice Cream Sandwich) es la seleccionada ya que el asistente de creación de proyectos de Android Studio indica que casi la totalidad de dispositivos Android de la actualidad, utilizan esta versión o una superior.

4.2 Aspectos relevantes.

4.2.1 Gestión del registro y autenticación de los usuarios.

Firebase proporciona un sistema de autenticación simple y seguro. Los usuarios pueden autenticarse utilizando credenciales distintas (correo electrónico contraseña/redes sociales, Google, Twitter, Facebook...).

Para implementar esta funcionalidad fue consultada la siguiente página de la documentación oficial de Firebase [10].

4.2.2 Almacenamiento de datos.

De entre las dos alternativas de almacenamiento propuestas por el ecosistema Firebase: Cloud Firestone y Realtime Database, se elige esta última ya que la primera se encuentra en estado beta.

Realtime Database es una base de datos NoSql en la nube, que permite almacenar y sincronizar datos en tiempo real, permitiendo a los distintos clientes la suscripción a cambios en diferentes datos almacenados. Para implementar la base de datos se ha consultado el siguiente link de la documentación oficial [11].

La estructura de la BD de Eventizer es la siguiente:

```
//En este nodo se almacenan las listas de contactos de los distintos usuarios
Contactos
    idUsuario1
        //Lista de contactos del usuario1
        idContacto2 //A su vez los ids de contacto son ids de usuario
        idContacto3
    idUsuario2
        idContacto1
        idContacto3

//En este nodo se almacenan los datos de los distintos eventos
Eventos
    idEvento
        idUsuarioAdmin
        titulo
        descripcion
        fecha
        hora
        latitud
        longitud
        listaMiembros
            idUsuario1
            idUsuario2
            idUsuario3

//En este nodo se almacenan los datos personales de los distintos usuarios
Usuarios
    idUsuario1
        email
        nombreUsuario
        listaEventos
            idEvento1
            idEvento3

//En este nodo se almacenan los mensajes asociados a los distintos eventos
Mensajes
    idEvento1
        idMensaje1
            emisor
            cuerpoMensaje
            timestamp
            resaltado //Este campo indica si el mensaje en cuestión fue resaltado por un adminastrador

//En este nodo se almacenan las invitaciones a eventos que tiene pendientes de gestionar
//los distintos usuarios(aceptar o descartar)
InvitacionesEventos
    idUsuario1
        idEvento4
        idEvento5
```

Ilustración 15 Estructura de la BD

Nótese la desnormalización de datos producida por la relación bidireccional entre los eventos y los usuarios. Es interesante poder acceder a la lista de miembros de un evento cuando consultamos dicho evento, así cómo obtener la lista de los eventos en los que participa un usuario determinado.

4.2.3 Uso de APIS de geolocalización de Google.

Para facilitar a los usuarios la elección de los lugares asociados a los distintos eventos, se optó por utilizar la API de Google Places dentro de Eventizer. Esta API proporciona lugares de interés cercanos a la ubicación actual del dispositivo; sitios como cafeterías, restaurantes, parques...
También proporciona la misma barra de búsqueda que utiliza Google Maps por si queremos seleccionar lugares que no esté cerca.

4.2.4 Validación de los distintos formularios de la aplicación.

Para validar los campos de los distintos formularios de la aplicación, (Registro, Creación de un nuevo evento…), se toma la decisión de utilizar la librería Awesome Validation que facilita bastante esta labor [14].

4.3 Estructura del proyecto.

A continuación, se puede observar la división de los distintos paquetes del proyecto en Android Studio en consonancia con los distintos bloques de funcionalidad de la aplicación. A mayores se encuentra el Singleton FirebaseUtil, que representa la clase dónde se centraliza toda la interacción con Firebase.

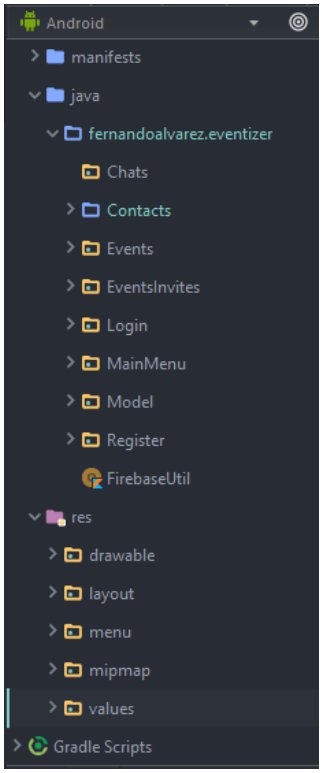

Ilustración 16 Estructura de paquetes de Eventizer

En todos los paquetes reflejados en la imagen anterior, se representan las responsabilidades del patrón de arquitectura MVP de la siguiente manera:

- Los distintos “fragments” y “activities” constituyen las vistas de la aplicación.
  Tienen métodos para actualizar la interfaz gráfica y para escuchar las interacciones de los usuarios.
- Cada vista tiene un presentador, que es el encargado de contactar con el singleton FirebaseUtil, recibir los datos de Firebase y mapearlos a clases de modelo, posteriormente, llama a los métodos de la vista para que se actualice la interfaz de usuario.
- Nuestro modelo estaría representado por las clases contenidas en el paquete Model y por nuestra BD de Firebase.

4.4 Principales problemas encontrados.

Durante la fase de implementación surgen 2 problemas principales:

- Diferencias a la hora de recuperar información en Firebase respecto a las bases de datos convencionales: A diferencia de las bases de datos convencionales que utilizan SQL y proporcionan poderosas consultas para realizar agrupamientos de datos en una sola consulta, en las NoSql, se tienen que realizar varias consultas para recuperar datos de nodos diferentes. Por esta limitación a menudo es aconsejable realizar desnormalizaciones de datos para poder realizar consultas más cómodas y rápidas como la explicitada en el punto 4.2.2.
- Curva de aprendizaje de Kotlin: Aunque para un programador con experiencia en Java, Kotlin no es excesivamente complicado y su curva de aprendizaje resulta suave, es muy recomendable leer la documentación oficial para entender las diferencias y nuevas características que aporta Kotlin con respecto a Java [15].

5. Pruebas

En este capítulo se detallará el plan de pruebas que se ha seguido para comprobar la correcta implementación de la aplicación.

Se ha comprobado el flujo de la aplicación con la ayuda del depurador integrado en Android Studio y el estado de la BD de Firebase, antes y después de ejecutar las pruebas que requieren acceso a la base de datos.

Prueba 1	**Registro: Comprobación de campos vacío.**
Precondiciones:	No estar registrado en el sistema.
Secuencia de acciones:	1. El usuario abre la aplicación y se le presenta la pantalla de login. 2. Accede al formulario de registro pulsando el botón de “¿Todavía no tienes cuenta?” 3. El usuario no cubre en su totalidad los campos del formulario de registro y presiona el botón de registrarse.
Postcondiciones:	La aplicación le muestra al usuario mensajes de error en cada uno de los campos sin cubrir, instándole a cubrirlos. El registro no se produce.

Prueba 2	**Registro: Las contraseñas no coinciden.**
Precondiciones:	No estar registrado en el sistema.
Secuencia de acciones:	1. El usuario abre la aplicación y se le presenta la pantalla de login. 2. Accede al formulario de registro pulsando el botón de “¿Todavía no tienes cuenta?” 3. El usuario no introduce el mismo valor en los campos Contraseña y Repetir Contraseña y presiona el botón de registrarse.
Postcondiciones:	La aplicación le muestra un mensaje al usuario indicando que las dos contraseñas no coinciden. El registro no se produce.

Prueba 3	**Registro: Datos correctos.**
Precondiciones:	No estar registrado en el sistema.
Secuencia de acciones:	1. El usuario abre la aplicación y se le presenta la pantalla de login. 2. Accede al formulario de registro pulsando el botón de “¿Todavía no tienes cuenta?” 3. El usuario cubre de forma correcta todos los campos del formulario de registro y pulsa el botón de registrarse.
Postcondiciones:	Se crea un nuevo registro en el módulo de autenticación de Firebase y, a mayores, en el nodo “Usuarios” de la base de datos reflejando su nombre de usuario, su contraseña y una lista vacía de eventos a los que pertenece. La aplicación redirige al usuario a la pantalla de login.

Prueba 4	**Login: Comprobación de campos vacíos.**
Precondiciones:	Tener una cuenta válida en la aplicación.
Secuencia de acciones:	1. El usuario abre la aplicación y se le presenta la pantalla de login. 2. El usuario pulsa el botón de login sin introducir ninguno de los campos requeridos (usuario y contraseña).
Postcondiciones:	La aplicación le muestra al usuario mensajes de error en cada uno de los campos sin cubrir, instándole a cubrirlos. El usuario no se autentica en la aplicación.

Prueba 5	**Login: Credenciales erróneas.**
Precondiciones:	Tener una cuenta válida en la aplicación.
Secuencia de acciones:	1. El usuario abre la aplicación y se le presenta la pantalla de login. 2. El usuario introduce un email y una contraseña. Uno de los dos es incorrecto. 3. El usuario pulsa el botón de login.
Postcondiciones:	La aplicación le muestra un mensaje al usuario indicando que las credenciales son erróneas. El usuario no se autentica en la aplicación.

Prueba 6	**Login: Credenciales correctas.**
Precondiciones:	Tener una cuenta válida en la aplicación.
Secuencia de acciones:	1. El usuario abre la aplicación y se le presenta la pantalla de login. 2. El usuario introduce un email y una contraseña correctas.
Postcondiciones:	El usuario se loguea en la aplicación con éxito. La aplicación lo redirige al menú principal.

Prueba 7	**Logout.**
Precondiciones:	El usuario está autenticado en la aplicación.
Secuencia de acciones:	1. El usuario selecciona la opción de “logout” en el menú principal.
Postcondiciones:	El usuario pierde su sesión. La aplicación lo redirige al formulario de login.

Prueba 8	**Listar invitaciones a eventos.**
Precondiciones:	El usuario está autenticado en la aplicación y tiene alguna invitación a un evento.
Secuencia de acciones:	1. El usuario accede a la lista de invitaciones a evento (primera opción del menú principal).
Postcondiciones:	La aplicación le muestra al usuario una lista de sus invitaciones a eventos; la lista de las invitaciones, que tiene cada usuario, está recogida en el nodo “InvitacionesEventos” en la BD.

Prueba 9	**Aceptar invitación a evento.**
Precondiciones:	El usuario está autenticado en la aplicación y tiene alguna invitación a un evento.
Secuencia de acciones:	1. El usuario accede a la lista de invitaciones a evento (primera opción del menú principal). 2. La aplicación le muestra al usuario una lista de sus invitaciones a eventos. 3. El usuario pulsa el botón de aceptar de algún evento de la lista.
Postcondiciones:	El evento seleccionado desaparece de la lista de invitaciones y se elimina de la BD del nodo “invitacionesEventos”. En el nodo “Usuarios”, en la entrada del usuario autenticado, aparecerá el evento aceptado en la entrada “listaEventos”. En el nodo “Eventos”, se añadirá el usuario a la entrada “listaMiembros” del evento en cuestión.

Prueba 10	**Rechazar invitación a evento.**
Precondiciones:	El usuario está autenticado en la aplicación y tiene alguna invitación a un evento.
Secuencia de acciones:	1. El usuario accede a la lista de invitaciones a evento (primera opción del menú principal). 2. La aplicación le muestra al usuario una lista de sus invitaciones a eventos. 3. El usuario pulsa el botón de aceptar invitación de algún evento de la lista.
Postcondiciones:	El evento seleccionado desaparece de la interfaz gráfica de la lista de notificaciones y se elimina de la BD del nodo “InvitacionesEventos”. En el nodo “Usuarios”, en la entrada del usuario autenticado, aparecerá el evento aceptado en la entrada “listaEventos”. En el nodo Eventos se añadirá el usuario a la entrada “listaMiembros” del evento en cuestión.

Prueba 11	**Listar eventos.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal).
Postcondiciones:	La aplicación le muestra al usuario una lista de los eventos a los que pertenece; esta lista está localizada en el campo “listaEventos” de cada usuario del nodo “Usuarios” de la BD.

Prueba 12	**Eliminar evento.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario elige un evento de la lista que desea eliminar.
Postcondiciones:	El evento seleccionado desaparece de la interfaz gráfica de la lista de eventos, de la BD del nodo “Eventos” y de la entrada “listaEventos” de cada usuario. Se elimina la entrada correspondiente al evento eliminado del nodo "Mensajes".

Prueba 13	**Crear evento: datos incompletos.**
Precondiciones:	El usuario está autenticado en la aplicación.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario pulsa el botón de añadir un nuevo evento. 3. El usuario cubre sólo alguno de los campos del formulario de creación del evento y pulsa el botón de crear evento.
Postcondiciones:	La aplicación le muestra al usuario mensajes de error en cada uno de los campos sin cubrir, instándole a cubrirlos. El evento no se crea.

Prueba 14	**Crear evento: datos completos.**
Precondiciones:	El usuario está autenticado en la aplicación.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario pulsa el botón de añadir un nuevo evento. 3. El usuario cubre todos los campos del formulario de creación del evento y pulsa el botón de crear evento.
Postcondiciones:	El evento se crea en el nodo “Eventos” de la BD. El usuario se añade al campo “listaMiembros” de este recién creado evento. Se añade el identificador del evento al campo “listaEventos” del usuario autenticado.

Prueba 15	**Consultar Ficha de evento.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario accede a la ficha de un evento al pulsar dicho evento de la lista.
Postcondiciones:	La aplicación le muestra al usuario todos los detalles del evento almacenados en el nodo “Eventos” de la BD: Título, descripción, fecha y hora y mapa con el marcador de la localización del evento.

Prueba 16	**Exportar evento a Google Calendar.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario accede a la ficha de un evento al pulsar un elemento de la lista. 3. El usuario pulsa el botón de exportación a Google Calendar.
Postcondiciones:	Los datos del evento se exportan satisfactoriamente al calendario de Google Calendar.

Prueba 17	**Abrir localización de evento en Google Maps.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario accede a la ficha de un evento al pulsar un evento de la lista. 3. El usuario pulsa el botón de abrir Google Maps encuadrado en el mapa donde se encuentra el pin de localización del evento.
Postcondiciones:	Se abre Google Maps con los datos de destino del evento seleccionado.

Prueba 18	**Chatear.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario accede a la ficha de un evento al pulsar un evento de la lista. 3. El usuario accede a la pantalla de chat del evento seleccionado. 4. El usuario introduce un mensaje en el cuadro de texto y pulsa el botón de enviar mensaje.
Postcondiciones:	El mensaje queda registrado en el nodo “Mensajes” de la BD, más concretamente, colgando del identificador del evento seleccionado.

Prueba 19	**Eliminar mensaje de un chat.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento del que es administrador.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario accede a la ficha de un evento del que es administrador, al pulsar un evento de la lista. 3. El usuario accede a la pantalla de chat del evento seleccionado. 4. El usuario selecciona un mensaje y pulsa la opción de eliminar.
Postcondiciones:	El mensaje es eliminado del nodo “Mensajes” de la BD.

Prueba 20	**Resaltar mensaje de un chat.**
Precondiciones:	El usuario está autenticado en la aplicación y pertenece a algún evento del que es administrador.
Secuencia de acciones:	1. El usuario accede a la lista de eventos a los que pertenece (segunda opción del menú principal). 2. El usuario accede a la ficha de un evento del que es administrador, al pulsar un evento de la lista. 3. El usuario accede a la pantalla de chat del evento seleccionado. 4. El usuario selecciona un mensaje y pulsa la opción de resaltar.
Postcondiciones:	El mensaje es eliminado del nodo “Mensajes” de la BD.

Prueba 21	**Añadir nuevo contacto: Email incorrecto.**
Precondiciones:	El usuario está autenticado en la aplicación.
Secuencia de acciones:	1. El usuario accede a su lista de contactos (tercera opción del menú principal). 2. El usuario pulsa el botón de añadir un nuevo contacto. 3. El usuario introduce un email que no está registrado en la aplicación.
Postcondiciones:	La aplicación le muestra al usuario un mensaje indicando que el correo electrónico introducido no pertenece a ningún usuario de Eventizer.

Prueba 22	**Añadir nuevo contacto: Email correcto.**
Precondiciones:	El usuario está autenticado en la aplicación.
Secuencia de acciones:	1. El usuario accede a su lista de contactos (tercera opción del menú principal). 2. El usuario pulsa el botón de añadir un nuevo contacto. 3. El usuario introduce un email de un usuario que está registrado en la aplicación.
Postcondiciones:	El nuevo contacto aparece en la lista de contactos de la interfaz gráfica y se añade al nodo “Contactos” de la BD, más concretamente, a la entrada del usuario autenticado.

Prueba 23	**Listar contactos.**
Precondiciones:	El usuario está autenticado en la aplicación y tiene algún contacto en su lista de contactos.
Secuencia de acciones:	1. El usuario accede a su lista de contactos (tercera opción del menú principal).
Postcondiciones:	La aplicación le muestra al usuario su lista de contactos; esta lista está almacenada en el nodo “Contactos” de la BD, más concretamente, en la entrada del usuario autenticado.

Prueba 24	**Eliminar contacto.**
Precondiciones:	El usuario está autenticado en la aplicación y tiene algún contacto en su lista de contactos.
Secuencia de acciones:	1. El usuario accede a su lista de contactos (tercera opción del menú principal). 2. El usuario pulsa el botón de eliminar un contacto.
Postcondiciones:	El contacto es eliminado de la lista de contactos de la interfaz gráfica y del nodo “Contactos” de la BD, más concretamente, en la entrada del usuario autenticado.

6. Conclusiones.

6.1 Grado de cumplimiento de los objetivos propuestos.

Tras repasar las distintas fases de este TFM, reflejadas en esta memoria, y analizar el producto entregado en forma de APP se puede concluir que Eventizer es una aplicación que permite la creación de eventos grupales con un chat asociado para facilitar la comunicación entre los distintos integrantes con alguna funcionalidad extra interesante, como la geolocalización del evento y la posibilidad de exportar los datos del evento a Google Calendar.

No obstante, por falta de tiempo, no se han podido implementar algunas funcionalidades planteadas en la fase de análisis:

- Login social: Consiste en poder autenticarse en Eventizer usando cuentas de Google, Facebook, Twitter…, sin tener que pasar por el registro de un nuevo usuario en la aplicación.

- Resaltado de mensajes importantes: Consiste en capacitar a los creadores/administradores de los eventos para poder marcar mensajes como “importantes”, dándole algún tipo de visibilidad extra en la pantalla de chat; por ejemplo, añadiéndoles un icono representativo y fijándolos siempre en las primeras posiciones de la lista de mensajes de las pantallas de chat de los distintos eventos.

- La posibilidad de actualizar los datos de los eventos una vez creados y la posibilidad de poder eliminarlos una vez que ya no estén vigentes.

6.2 Grado de seguimiento de la planificación.

Considero que el seguimiento de la planificación temporal, así como el nivel de los entregables presentados en las distintas PECS fue bastante mediocre, influido en parte por mi situación durante el segundo cuatrimestre al estar matriculado de forma paralela en otro master cuyas fechas de entrega de prácticas/PECS se solaparon.
A continuación, repasaré los distintos hitos del proyecto que coinciden con las diferentes PECS:

Para la PEC1: Se cumplieron los plazos y se planteó el plan de trabajo de todo recogido en el primer punto de esta memoria.

Para la PEC2: Se cumplieron los plazos realizando la parte de la memoria correspondiente a las fases análisis y diseño de la aplicación, pero dejando atrás algún punto importante como los escenarios del diseño centrado en el usuario.

En la PEC3: Me centré en mejorar la memoria, corrigiendo todos los detalles comentados por el consultor en las correcciones de las dos primeras PECS, desatendiendo la implementación de la aplicación que era la actividad principal de esta tercera PEC.

De las 106 horas destinadas a la implementación que estaban planificadas para este periodo, sólo pude dedicarle unas 50 horas, no pudiendo presentar una aplicación medianamente funcional para la fecha de entrega de esta PEC. Por lo tanto, se acumularon estas 56 horas de déficit a otras 25 adicionales, no contempladas en la planificación inicial, en período de la entrega final.

6.3 Posibles mejoras de la aplicación.

Aparte de las funcionalidades no implementadas comentadas en el punto 6.1, considero que la implementación de las siguientes mejoras convertiría a Eventizer en un producto mucho más completo:

- Implementar notificaciones para que los usuarios de la aplicación puedan enterarse con más facilidad de nuevas invitaciones o de nuevos mensajes en sus eventos.

- Mejorar el chat, permitiendo envío de imágenes, vídeos y archivos.

- Adaptar la aplicación para las tablets, posiblemente reorganizando algunas ventanas para aprovechar mejor las pantallas de este tipo de dispositivos, pudiendo mostrar un volumen de información mayor que en las pantallas de los móviles.

- Traducir la aplicación a distintos idiomas para que pueda ser usada por usuarios de distintos países.

- Implementar tests unitarios y de integración para asegurar la calidad de la implementación de la app.

6.4 Opinión personal del trabajo realizado.

Creo que haberme enfrentado a la planificación y realización de un proyecto Android completo, aunque de pequeña envergadura como la aplicación propuesta en este TFM, resultó muy enriquecedor de cara a afianzar y ampliar mis competencias en la programación de aplicaciones móviles en el ecosistema Android.

La mayoría de las PECS, presentadas durante las asignaturas de Android del máster, estaban constituidas en su mayoría por pocos elementos (una o dos “activities” junto con algunos “fragments”), estando muy focalizadas en asentar algún concepto o practicar con alguna herramienta/librería en concreto.

Al enfrentarme a la construcción de Eventizer, tuve que tener muy en cuenta nuevos aspectos como la implementación de una arquitectura de aplicación en

Android, manejar el ciclo de vida complejo de varios “fragments” activos en paralelo, estructurar una base de datos “NoSql”, recuperar datos de manera asíncrona…

Otra de las grandes dificultades, que creo que merece la pena mencionar, fue la realización de la planificación temporal del proyecto sin tener experiencia previa en la realización de aplicaciones completas en Android, sumado a la utilización de nuevos elementos que siempre necesitan de un período de familiarización como, por ejemplo, Kotlin.

Kotlin resultó un gran descubrimiento para mí como lenguaje de programación para el ecosistema Android. Pienso que facilita bastante la programación respecto a Java. Dentro de este lenguaje de programación destacaría 4 características por encima del resto:

- Creación automática de los “getters”, “setters” y el constructor de una clase al etiquetarla como “data class”. Esto resulta muy útil cuando queremos definir “POJOS” o “DTOS” en nuestras aplicaciones [16].

- “View Binding”: Kotlin nos permite utilizar directamente los elementos definidos en los “layouts” de forma directa sin tener que utilizar “findViewById()” [17].

- Funciones lambda y paso de firmas de función o de las propias funciones como parámetros de métodos. Esta caracterísitica es muy útil a la hora de establecer los distintos “listeners” de los elementos gráficos de la aplicación ayudando a que nuestro código sea más conciso y vistoso [18].

- Variables de inicialización tardía: Muchas veces llenamos nuestro código Java de variables privadas en nuestras clases inicializadas a “null”. Kotlin nos permite definir variables de un tipo determinado que están a la espera de que las inicialicemos con un valor determinado. Son muy útiles cuando estamos esperando por un evento o dato para darle el valor adecuado a dicha variable [19].

7. Bibliografía.

1. Looping en la Play Store. Última visita el 05/06/2018.
https://play.google.com/store/apps/details?id=familynet.de.src&hl=es

2. Link TimeTree en la Play Store. Última visita el 05/06/2018.
https://play.google.com/store/apps/details?id=works.jubilee.timetree&hl=es

3. Desarrollo en cascada. Última visita el 05/06/2018.
https://es.wikipedia.org/wiki/Desarrollo_en_cascada

4. Desarrollo por incrementos. Última visita el 05/06/2018
https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente

5. Definición caso de uso. Última visita el 05/06/2018.
https://es.wikipedia.org/wiki/Caso_de_uso

6. Arquitectura MVC. Última visita el 05/06/2018.
https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6

7. Arquitectura MVP. Última visita el 05/06/2018.
https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-8a6faaae14a5

8. Link Firebase. Última visita el 05/06/2018.
https://firebase.google.com/?hl=es-419

9. Link Android Studio. Última visita el 05/06/2018.
https://developer.android.com/studio/?hl=es-419

10. Página documentación Firebase Auth. Última visita el 05/06/2018.
https://firebase.google.com/docs/auth/?hl=es-419

11. Página documentación Firebase Realtime Database. Última visita el 05/06/2018.
https://firebase.google.com/docs/database/?hl=es-419

12. Página documentación Google Places. Última visita el 05/06/2018.
https://developers.google.com/places/android-api/start?hl=es-419

13. Página documentación Google Maps. Última visita el 05/06/2018.
https://developers.google.com/maps/documentation/android-sdk/intro?hl=es-419

14. Link Awesome Validation. Última visita el 05/06/2018.
https://github.com/thyrlian/AwesomeValidation

15. Link documentación Kotlin. Última visita el 05/06/2018.
https://kotlinlang.org/docs/reference/

16. Link “data classes” en Kotlin. Última visita el 05/06/2018.
https://kotlinlang.org/docs/reference/data-classes.html

17. “View binding” en Kotlin. Última visita el 05/06/2018.
https://kotlinlang.org/docs/tutorials/android-plugin.html

18. Funciones “lambda” en Kotlin. Última visita el 05/06/2018.
https://kotlinlang.org/docs/reference/lambdas.html

19. Variables de inicialización tardía en Kotlin. Última visita el 05/06/2018.
https://kotlinlang.org/docs/reference/properties.html

8.Anexos.

8.1 Anexo A: Instrucciones de compilación.

Abrir el proyecto con Android Studio y ejecutar la opción de “make project”.
Esta opción se encuentra en la pestaña “Build” -> “Make Project” o en el siguiente icono rodeado de blanco.

8.2 Anexo B: Manual de usuario.

A continuación, se presentará una guía apoyada con imágenes, para facilitar a los usuarios la navegación por las distintas pantallas de la aplicación.

Login y Registro

Cuando un usuario inicia la aplicación, se le muestra la pantalla de login.

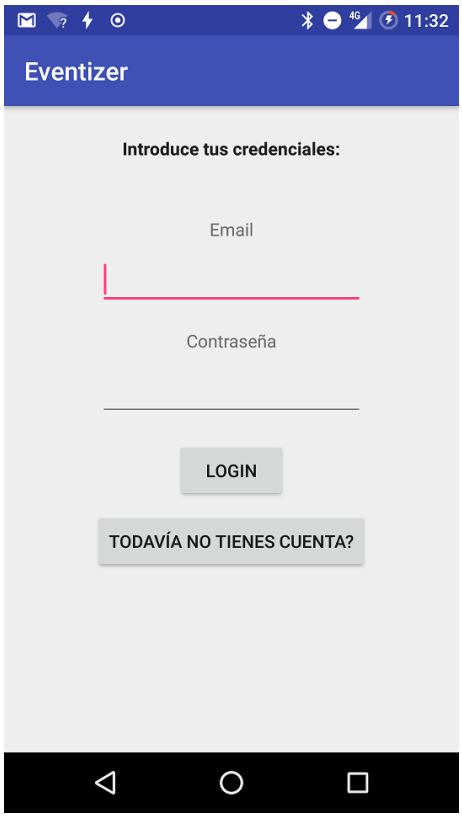

Ilustración 17: Manual - Pantalla Login

Para acceder al formulario de registro en la aplicación, el usuario tendrá que pulsar el botón “Todavía no tienes cuenta”.

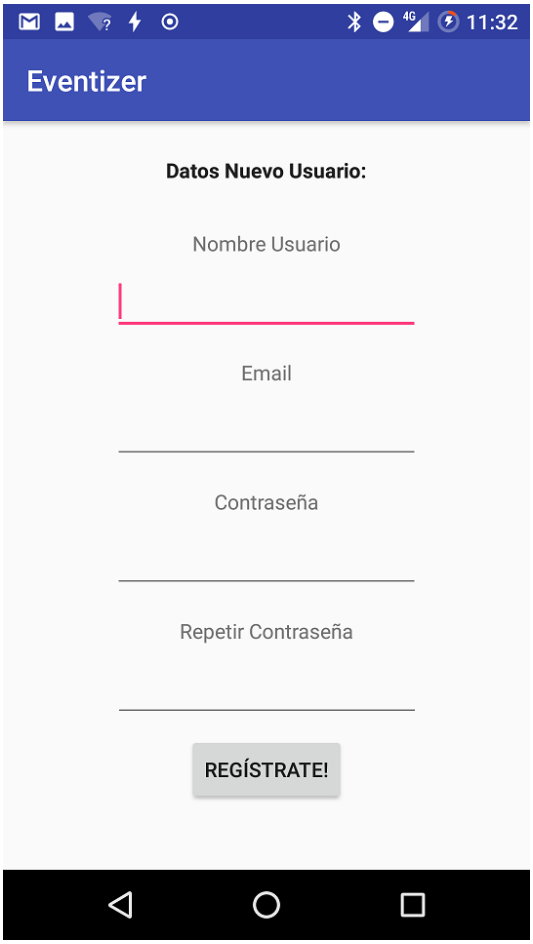

Ilustración 18: Manual - Pantalla Registro

Menú Principal

Una vez que el usuario introduce unas credenciales correctas en la pantalla de login, la aplicación lo redirige al menú principal. Este menú consta de 3 pestañas:

1- “Invitaciones a eventos”: En esta pestaña aparecerá las invitaciones para poder unirnos a eventos creados por otros usuarios. Cuando le llega a un usuario una invitación tiene dos opciones: descartarla pulsando sobre la el botón rojo o aceptarla pulsando sobre el botón verde. Si la descarta esta desaparecerá de su lista de invitaciones. Si la acepta, el evento en cuestión aparecerá en su lista de eventos (segunda opción del menú principal).

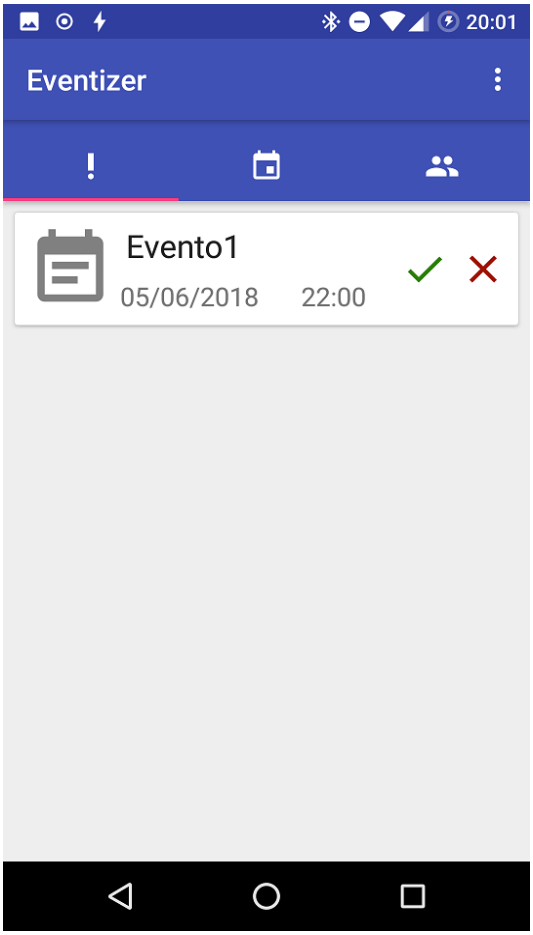

Ilustración 19: Manual - Pantalla Invitaciones Eventos

2- “Lista de eventos”: En esta pestaña se muestra la lista de los eventos de los que forma parte el usuario. El usuario podrá acceder al detalle de un evento, pulsando sobre el mismo. El usuario podrá pulsar el botón morado de la esquina inferior derecha para crear un nuevo evento.

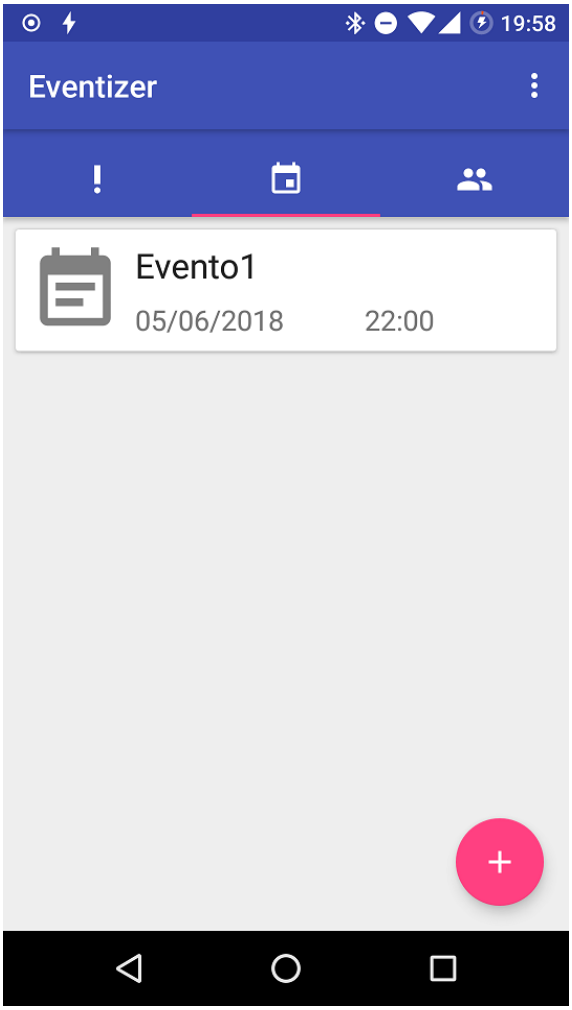

Ilustración 20: Manual - Pantalla Lista Eventos

3- “Lista de Contactos”: En esta pantalla se muestra la lista de los contactos del usuario. Pulsando el botón morado de de la esquina inferior derecha podrá añadir un nuevo contacto si conoce el email de otro usuario de Eventizer. El usuario podrá borrar a un contacto pusando sobre el icono de la papelera.

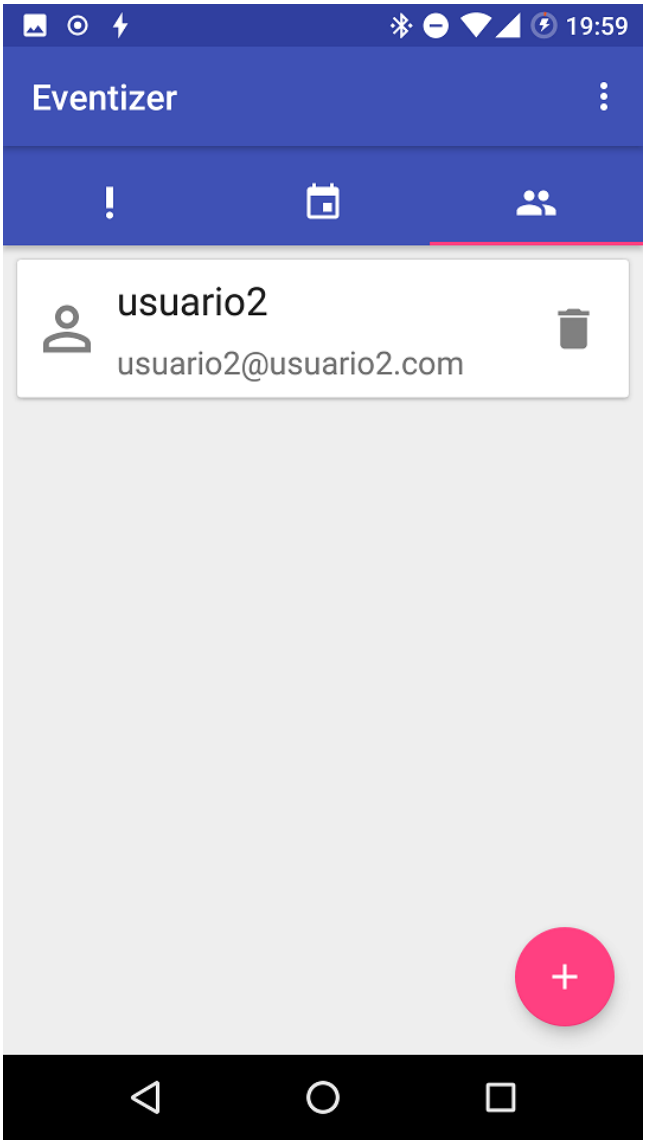

Ilustración 21: Manual - Pantalla Lista Contactos

Pantalla detalle de evento y chat

Cuando un usuario selecciona un evento de su lista, accederá a una pantalla con dos pestañas. La primera se corresponde con la pantalla que muestra los detalles del evento seleccionado, así como las opciones de exportación a Google Calendar, la posibilidad de invitar contactos, si el usuario es administrador/creador del evento, o la posibilidad de abrir Google Maps si pulsa sobre el marcador rojo en el mapa.

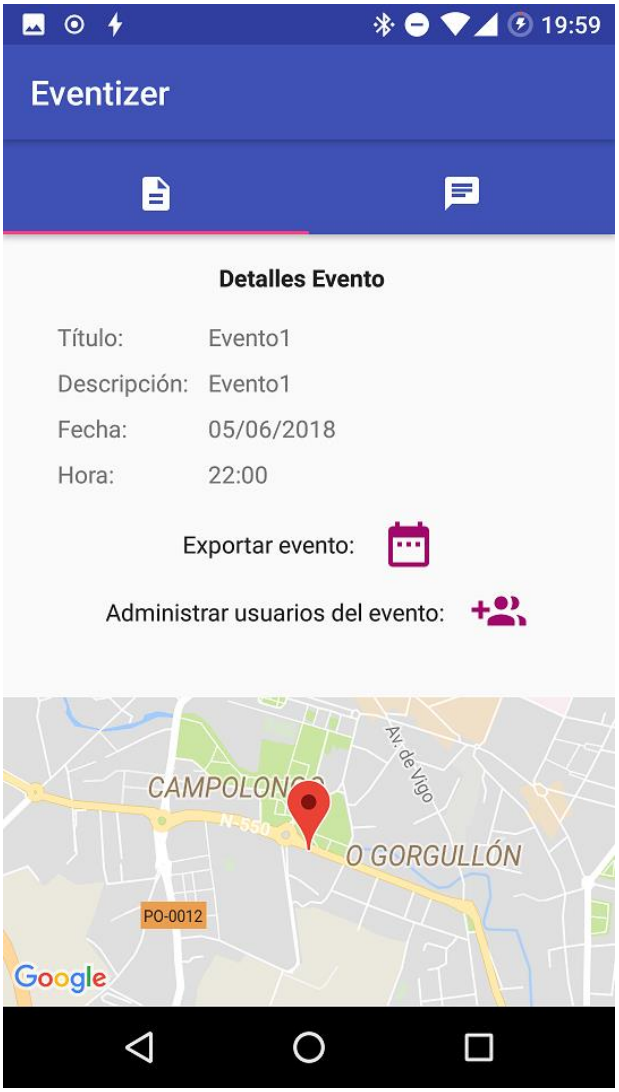

Ilustración 22: Pantalla Detalle Evento

La segunda pestaña se corresponde con la pantalla de chat asociada al evento en cuestión. El administrador/creador del evento podrá eliminar los mensajes que no considere relevantes utilizando el icono de la papelera.

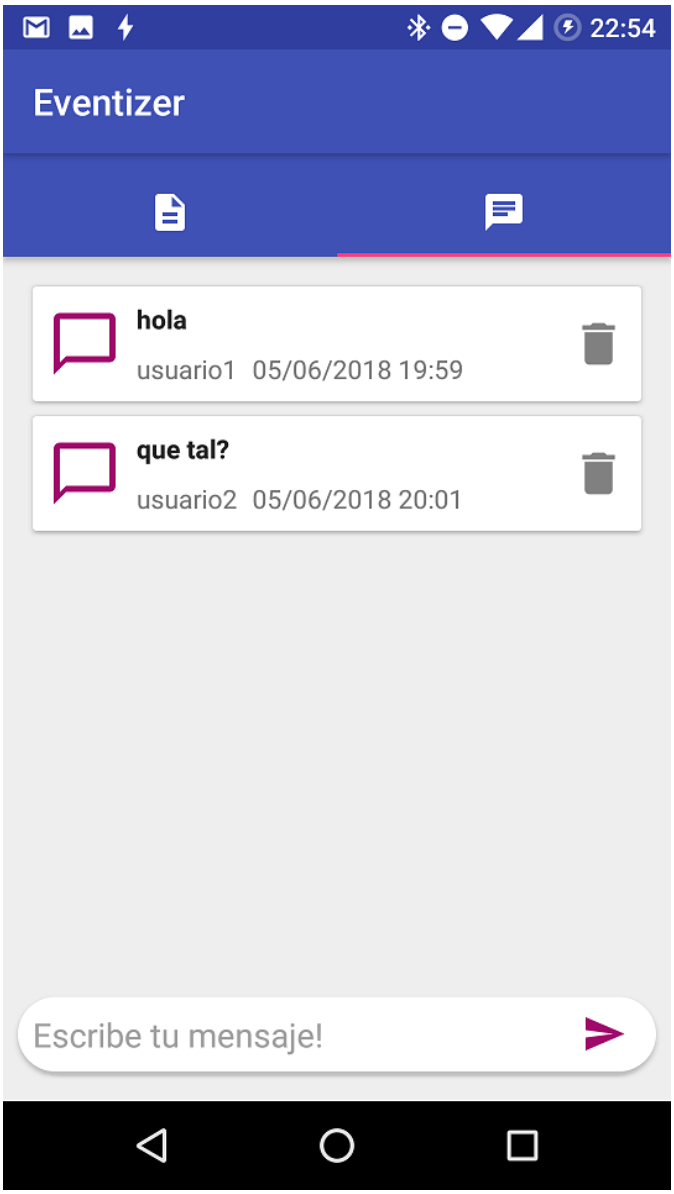

Ilustración 23: Manual - Pantalla Chat

8.3 Anexo C: Credenciales de prueba de la aplicación.

Para poder probar la aplicación se adjuntan dos cuentas de usuario:

Nombre de usuario: usuario1
Correo: usuario1@usuario1.com
Contraseña: usuario1

Nombre de usuario: usuario2
Correo: usuario2@usuario2.com
Contraseña: usuario2