Proyecto fin de carrera. Escuela de Ingeniería Informática.

Universidad de Las Palmas de Gran Canaria.

Hotel Info-Social Service – Hiss & Acclaim

Borja Rafael Castillo Hernández

Jorge Mor Varela

Las Palmas de Gran Canaria

Abril de 2015

Proyecto fin de carrera de la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria presentado por los alumnos:

Borja Rafael Castillo Hernández

Jorge Mor Varela

Título del proyecto: *Hotel Info-Social Service – Hiss & Acclaim*

Tutores: *Alexis Quesada Arencibia y Agustín Sánchez Medina*

Dedicatoria

A mis padres, mi hermano y mis amigos Jared y Jorge

Borja Rafael Castillo Hernández

A mi familia, mis amigos Jared y Borja y mi compañera de viaje

Jorge Mor Varela

Agradecimientos

Este proyecto representa una etapa muy importante en mi vida. Una etapa en la que me han acompañado muchas personas importantes y que, sin ellas, no hubiera sido capaz de lograrlo.

En primer lugar a mis tutores, *Alexis y Agustín* por ayudarme siempre que lo he necesitado y guiarme durante el proyecto. También, a todos los grandes amigos que he hecho durante la carrera: *Jared, Jorge, Jonás, Rubén y Raúl,* que a día de hoy siguen estando ahí. Asimismo, al resto de amigos que conocía anteriormente, tanto del colegio como de la infancia.

En segundo lugar, y no por ello menos importante, a mi familia, a mi hermano *Dan* por apoyarme desde niño. A mi madre, *Susana*, por la fuerza que tiene y me trasmite día a día para seguir adelante. Y a mi padre, *Rafael*, que allá donde esté agradezco su esfuerzo y cariño que me dio durante tantos años.

Por último, agradecer también a todos los profesores que me impartieron clase durante mi formación y a todas aquellas personas que me han apoyado en algún momento de su vida.

Borja Rafael Castillo Hernández

Tras la entrega y presentación de este proyecto se esconden muchas horas de dedicación que en múltiples ocasiones han provocado que desatendiera a familiares o amigos. Es por ello que hago una primera mención dirigida a ellos, a esos que han padecido las consecuencias de este proyecto, agradeciéndoles profundamente su comprensión, y en especial a ti que eres la que más las ha sufrido.

Por otro lado, no puedo dejar de dar las gracias a los profesores y personas que han influido en mi formación a lo largo de todos estos años, y que han ayudado con ello a que llegue hasta aquí. Concretamente debo agradecérselo a los profesores que me animaron a venir a la universidad cuando yo no lo tenía muy claro.

Finalmente, plasmo en estas hojas un agradecimiento especial a mis padres, que sin duda han sido siempre los mejores y más importantes consejeros y me han ayudado, en gran medida, a poder ser como soy.

Gracias a todos.

Jorge Mor Varela

Índice

Índice de imágenes

1 Introducción

Los hoteles tienen la obligación de darle al cliente un servicio muy completo si desean fidelizarlo. Así, deben hacer lo posible para mejorar el confort y lograr que tengan una sensación de bienestar.

Dentro de los servicios que ofrece un complejo hotelero, consideramos que, actualmente, la comunicación con los clientes no es todo lo buena que debería ser. Así, suelen disponer de una recepción con personal limitado para la misma en la que se forman generalmente colas, un sistema telefónico obsoleto y un tablón de novedades.

Por esto, se propone desarrollar una aplicación que permitiría aumentar la calidad del servicio de comunicaciones dentro de complejos hoteleros, de forma que el cliente se pueda poner en contacto con el hotel para presentarle sus necesidades, dudas o quejas y, a su vez, con otros clientes del hotel, mejorando la interacción social y la experiencia del cliente.

En este documento se presentan todas y cada una de las fases que hemos llevado a cabo durante el proceso de elaboración del proyecto fin de carrera, donde se verán reflejados todos los aspectos relevantes del mismo.

1.1 Organización de este documento

En esta memoria se presenta, a lo largo de nueve capítulos y seis anexos, el desarrollo de una aplicación móvil basada en la tecnología Android, la cual pretende ser una solución a la situación presentada en el epígrafe anterior. El trabajo se ha estructurado de la siguiente forma:

- Estado del arte. En este capítulo se estudia la diversidad de aplicaciones de carácter hotelero y social existentes en la actualidad.
- Objetivos. Se exponen los objetivos de este proyecto.
- Metodología. Se especifican las técnicas utilizadas para el desarrollo de este proyecto desde la perspectiva del desarrollo de software.

- Recursos necesarios. Apartado donde se indican los recursos utilizados, tanto hardware como software.
- Plan de trabajo y temporización. Se detalla la división del proyecto en tipos de trabajo, así como su planificación estimada.
- Análisis. Estudio en profundidad de la funcionalidad deseada, expresada en términos de captura de requisitos y casos de uso.
- Diseño. Estudio en profundidad de la arquitectura de la solución planteada, en términos de entidades de diseño, tales como subsistemas, clases e interfaces.
- Resultados y conclusiones. Exposición de los resultados obtenidos así como de las conclusiones extraídas.
- Anexo I. Manuales de usuario. Guía de uso de las aplicaciones desarrolladas.
- Anexo II - Detalles sobre la implementación. Se incluye adjunto un extracto de varios fragmentos significativos de código fuente extraídos de la fase de implementación.
- Anexo III - Trabajo futuro. Se detalla la hoja de ruta que puede seguirse desde el momento actual para seguir desarrollando este proyecto.
- Anexo IV - Tablas de especificación de casos de uso desarrollados. Se incluyen las tablas de especificación de los sesenta casos de uso desarrollados.
- Anexo V - Índice de imágenes. Se incluye en este apartado un índice de las imágenes que se han añadido al documento.
- Anexo VI - Bibliografía. Último anexo de este documento donde pueden consultarse las referencias utilizadas.

Por último, se adjunta un CD que incluye el código fuente desarrollado en este proyecto, siguiendo la estructura de carpetas de la Imagen 1 “Estructura del CD adjunto”:

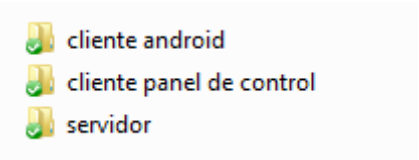


Imagen 1: Estructura del CD adjunto

2 Estado del arte

A continuación se expondrán datos y explicaciones que servirán para refinar más que es lo que debemos ofrecer para diferenciarnos de nuestros posibles competidores. Nos centraremos en analizar qué funciones no debemos olvidar incluir en la aplicación, porque son comunes en aplicaciones similares y, también, en descubrir los competidores que hay actualmente en el mercado que estén desarrollando aplicaciones similares o que puedan haber abordado en algún momento el desarrollo de las funciones que piensa ofrecer la aplicación.

2.1 Estadísticas

En este apartado se aportan algunas estadísticas con el fin de explicar cuál es el contexto actual sobre el uso de móviles y aplicaciones, y la tendencia que está apareciendo al llevar a cabo el proceso de reserva hotelera, razones por las cuales se desarrolla *Hiss & Acclaim* para un entorno móvil en lugar de para otro tipo de entorno.

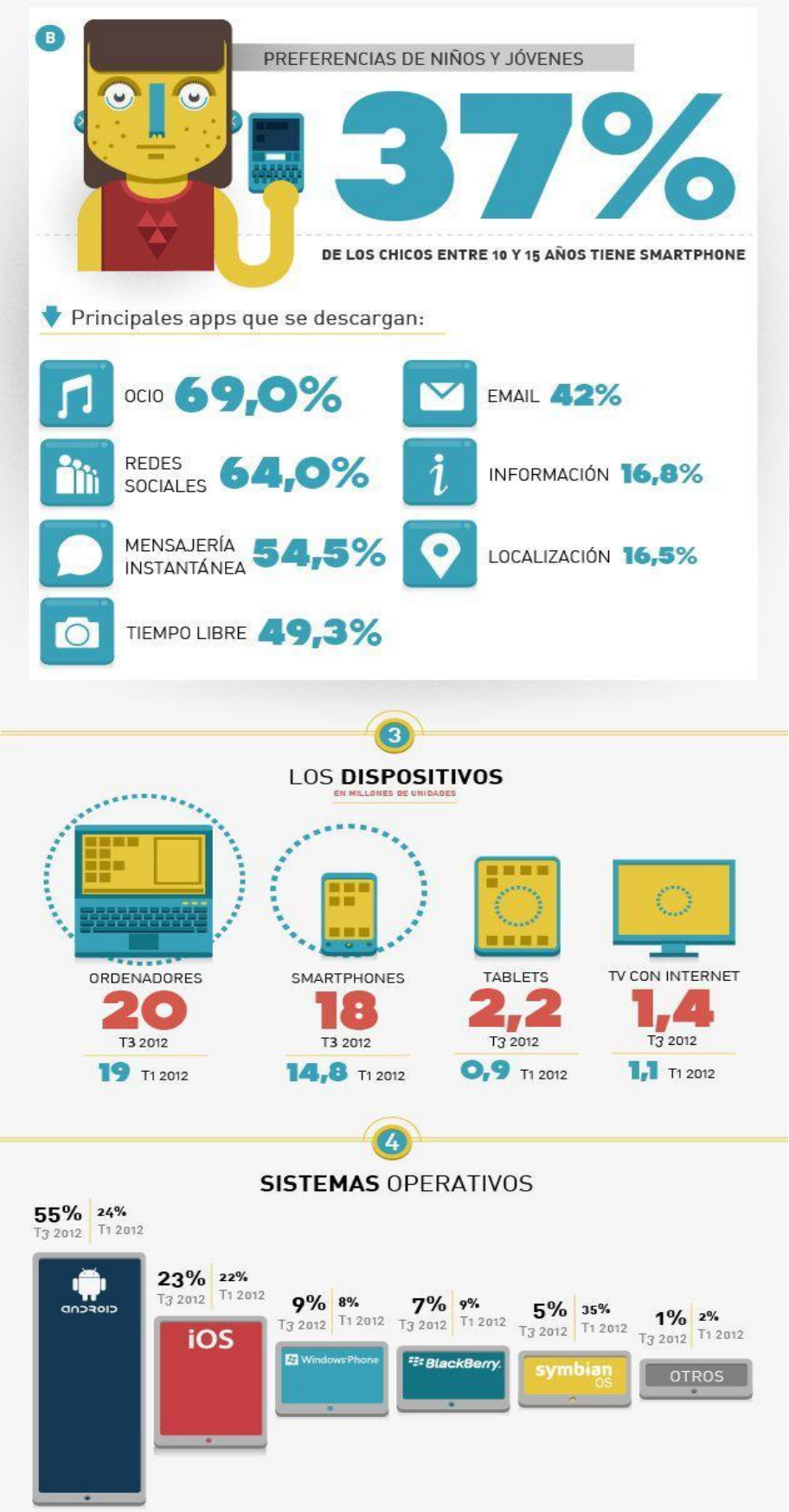
2.1.1 Estadísticas de 2012 en España

- Incremento del 15% de las reservas hoteleras a través del móvil en España.
- El 63% de los usuarios españoles de móviles poseen teléfonos inteligentes.
- El 30% de españoles han hecho reservas a través de móviles.
- Deseo de comercialización de habitaciones de hoteles en último instante.
- 20% de las visitas a web hoteleras se harán en 2013 a través de móviles.

Fuente: http://www.hosteltur.com/127850_20-visitas-web-hoteleras-se-haran-moviles-2013.html

2.1.2 Uso de aplicaciones móviles en España

Adjuntamos el estudio realizado por *The App Date* en septiembre de 2012, sobre el uso de las aplicaciones móviles en España, como se puede ver en el siguiente como se puede ver en la imagen 2.


B
PREFERENCIAS DE NIÑOS Y JÓVENES
37%
DE LOS CHICOS ENTRE 10 Y 15 AÑOS TIENE SMARTPHONE
Principales apps que se descargan:
OCIO 69,0%
EMAIL 42%
REDES SOCIALES 64,0%
INFORMACIÓN 16,8%
MENSAJERÍA INSTANTÁNEA 54,5%
LOCALIZACIÓN 16,5%
TIEMPO LIBRE 49,3%
3
LOS DISPOSITIVOS
EN MILLONES DE UNIDADES
ORDENADORES
20
T3 2012
19 T1 2012
SMARTPHONES
18
T3 2012
14,8 T1 2012
TABLETS
2,2
T3 2012
0,9 T1 2012
TV CON INTERNET
1,4
T3 2012
1,1 T1 2012
4
SISTEMAS OPERATIVOS
55% T3 2012 | 24% T1 2012
ANDROID
23% T3 2012 | 22% T1 2012
iOS
9% T3 2012 | 8% T1 2012
Windows Phone
7% T3 2012 | 9% T1 2012
BlackBerry
5% T3 2012 | 35% T1 2012
symbian OS
1% T3 2012 | 2% T1 2012
OTROS

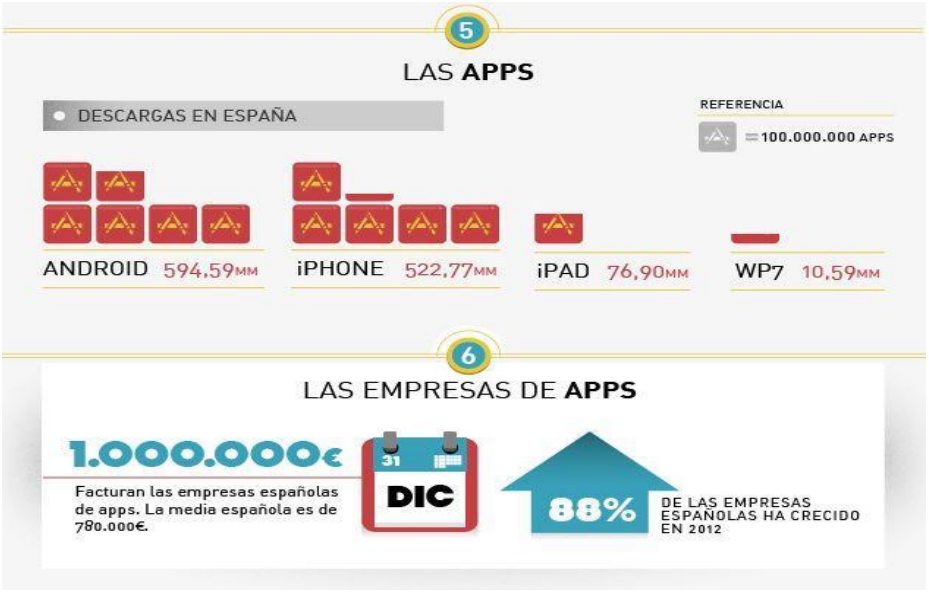


Imagen 2: Encuesta de datos de uso móvil

Como se puede apreciar en el estudio, los datos recogidos demuestran que el uso del móvil y las aplicaciones móviles está aumentando a pasos agigantados, siendo Android el sistema predominante.

2.2 Complementos y aplicaciones para integrar en Facebook

Analizaremos brevemente algunas de las aplicaciones del mercado desarrolladas anteriormente que se integran con Facebook y están relacionadas con el proyecto en desarrollo.

Estas aplicaciones se caracterizan por estar integradas en el propio *Facebook* del hotel, cosa que aunque proporciona la ventaja de evitar tener que mantener una aplicación nueva de la propia empresa, tiene el inconveniente de aportar una menor flexibilidad y menos posibilidades. Ello se debe a que son solamente complementos y todo en definitiva queda limitado a las posibilidades que *Facebook* proporcione en cada momento.

2.2.1 The booking button

Es una aplicación para *Facebook* que permite tener al hotel un sistema de reserva de habitaciones.

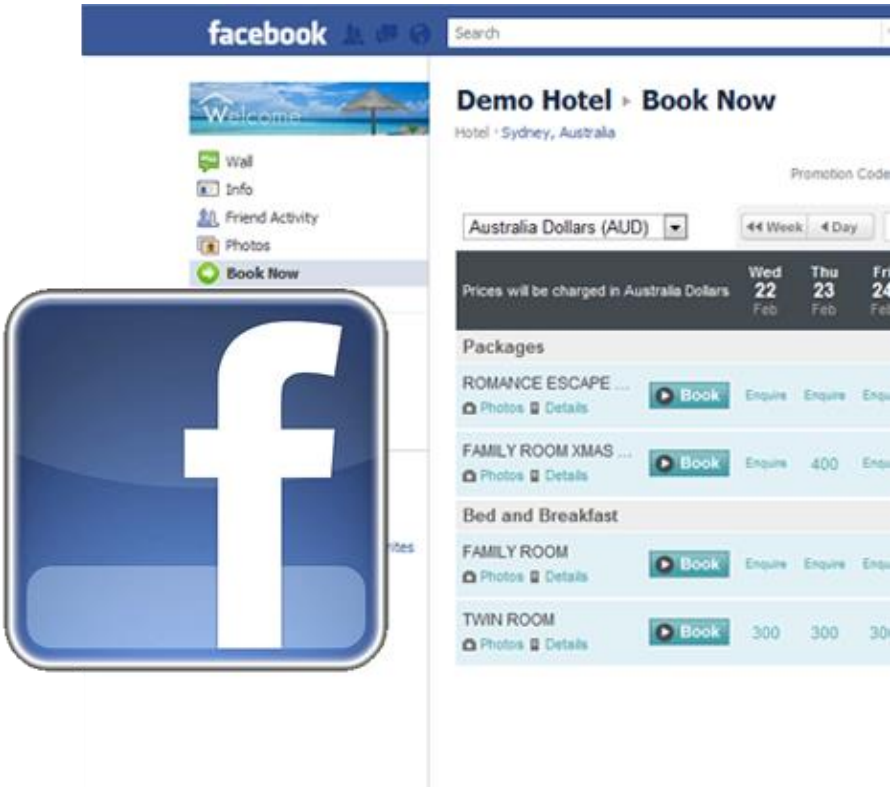


Imagen 3: Aplicación *The booking button*

2.2.2 Hotels Tab

Permite poner información variada del hotel en pestañas dentro del *Facebook* del hotel.

Imagen 4: Hotels Tab, aplicación de Facebook

2.2.3 Bookassist

Es un asistente para realizar reservas en el hotel que además contiene otros apartados con todo tipo de información y ofertas del hotel.

Imagen 5: Bookassist, aplicación de Facebook

2.3 Aplicaciones móviles exclusivas de un hotel o cadena

A continuación analizaremos el mercado en cuanto a las aplicaciones de hoteles desarrolladas con anterioridad.

Como podremos ver, la gran mayoría de aplicaciones de hoteles existentes en el mercado son meramente informativas, añadiendo algunas de ellas la posibilidad de reservar habitaciones como único elemento interactivo. Algunas tienen una interfaz amigable y bonita y por lo general contienen información básica sobre el propio hotel como el tipo de habitaciones de las que disponen, menú del restaurante, y los alrededores (tiendas o lugares importantes cercanos, mapas, espectáculos y actividades, etc.).

Nuestra aplicación busca ofrecer la mayoría de posibilidades informativas que contienen aplicaciones como las que vamos ver a continuación, además de otras muchas posibilidades centradas en la interactividad cliente-cliente y cliente-hotel. Siendo este último aspecto, una de las mayores carencias de las aplicaciones que se pueden encontrar actualmente, como veremos a continuación. De cada aplicación se destacarán sus puntos más importantes.

2.3.1 Wynn Las Vegas and Encore App

Permite hacer reservas de habitaciones y restaurantes además de tener mucha información general.

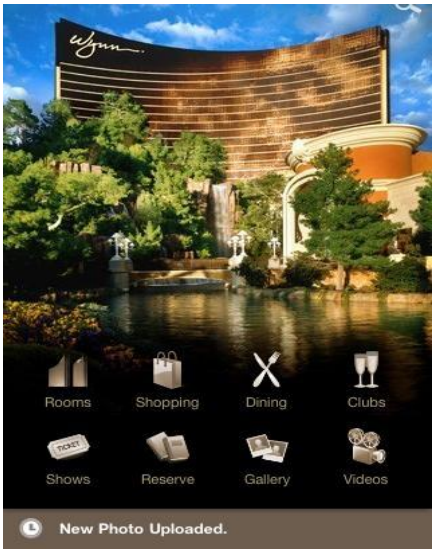


Imagen 6: Wynn Las Vegas, aplicación específica.

2.3.2 ARIA

Integrado con *Facebook* y *Twitter*, además de tener un modo offline.

Imagen 7: Aria, aplicación específica

2.3.3 Caesars Palace Las Vegas

Permite hacer reservas de habitaciones y restaurantes además de tener mucha información general.

Imagen 8: Caesars Palace Las Vegas

2.3.4 New York-New York

Integrado con *Twitter* y con modo offline. Permite realizar reservas.

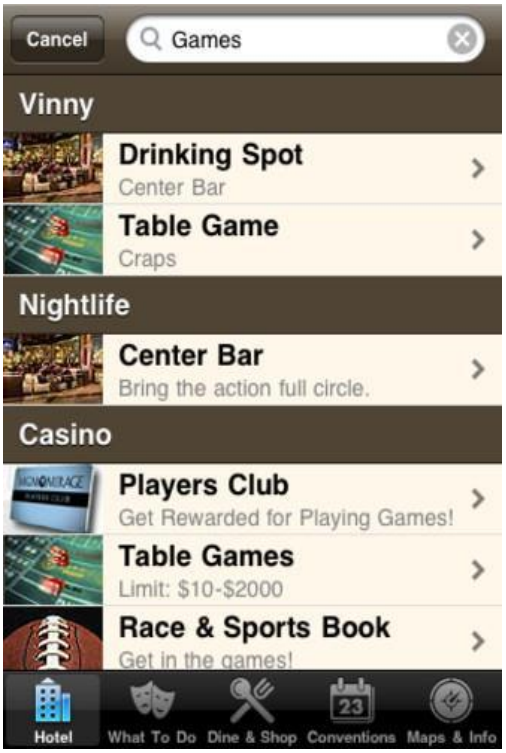


Imagen 9: New York - New York. Aplicación específica

2.3.5 MGM Grand

Integrado con *Twitter* y con modo offline. Permite realizar reservas.

Imagen 10: MGM Grand. Aplicación específica

2.3.6 Mandalay Bay

Integrado con *Twitter* y con modo offline. Permite realizar reservas.

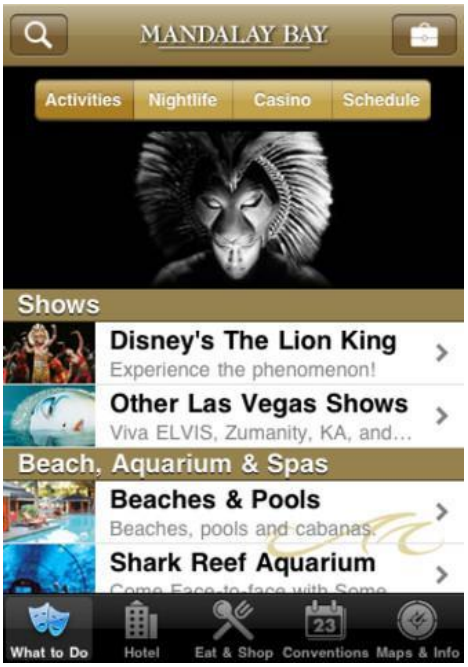


Imagen 11: Mandalay Bay

2.3.7 Luxor Las Vegas

Permite hacer reservas de habitaciones.

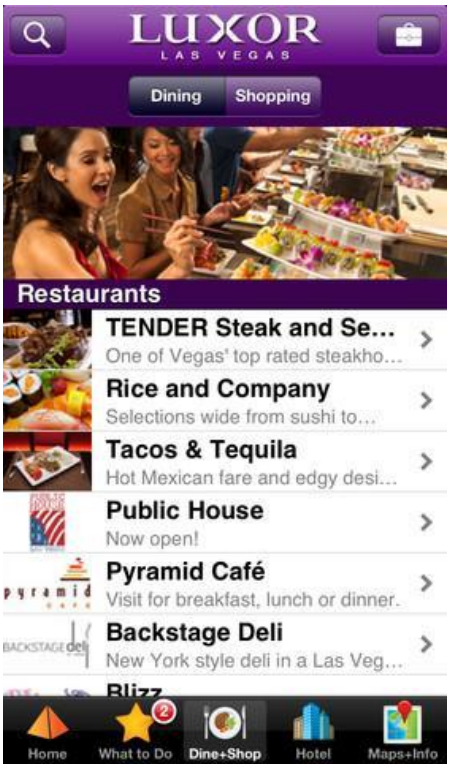


Imagen 12: Luxor Las Vegas, aplicación específica

2.3.8 Accorhotels

Aplicación que permite buscar y reservar habitaciones en hoteles de todo el mundo mediante el operador de hoteles a nivel mundial *Accor*. No ofrece muchas más posibilidades y se limita a ofrecer hoteles de *Accor,* con lo que no hay demasiada variedad en cada país.

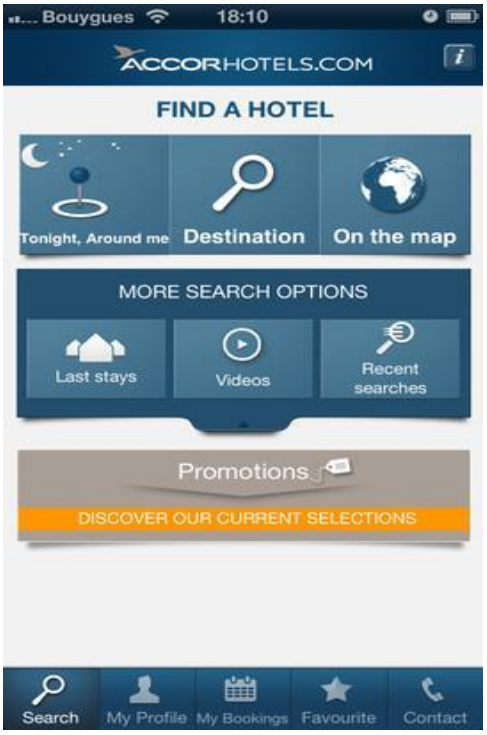


Imagen 13: AccorHotels. Aplicación específica

2.4 Gestores configurables por cada hotel para crear su propia app

Estudiaremos gestores que permiten mediante configuraciones realizar un app personalizada propia del hotel. Como veremos por lo general tienen grandes limitaciones, aunque pueden ser útiles para pequeños hoteles.

2.4.1 mGuest

La aplicación *mGuest* ofrece a hoteles la capacidad de crear conexiones más profundas y significativas con sus clientes. Ofrece un servicio de "*e-concierge*" (conserje online) funcional en línea e incluso facilita las reservas y los servicios en la habitación.

Las características adicionales de *mGuest* incluyen la integración de medios sociales con los blogs, hacer el *check-in* en el hotel, compras en grupo y módulos tales como eventos y bodas. Elegantemente útil, esta plataforma móvil permite a los hoteles personalizar rápidamente, administrar y actualizar fácilmente sus aplicaciones móviles. Inicialmente estuvo disponible para *iPhone*, *iPad* y *Android*, pero ya no existe.

Imagen 14: MGuest, aplicación específica

2.5 Aplicaciones de uso no exclusivo por parte de un hotel o cadena

Aquí se muestran algunas aplicaciones que no son de un hotel o cadena hotelera en concreto, sino que permiten interactuar u obtener información de distintos hoteles y cadenas hoteleras. Estas serán las aplicaciones más completas que hemos podido analizar pero su punto negativo es que en general no permiten una interacción social entre clientes, ni una personalización propia concreta por parte de los hoteles.

2.5.1 Bellhop

Es una aplicación de reserva de habitaciones para *Mac OS*. Hay infinidad de aplicaciones similares a esta que, básicamente, lo que ofrecen son datos de relevancia del hotel así como opiniones de anteriores clientes y la posibilidad de reservar habitaciones.

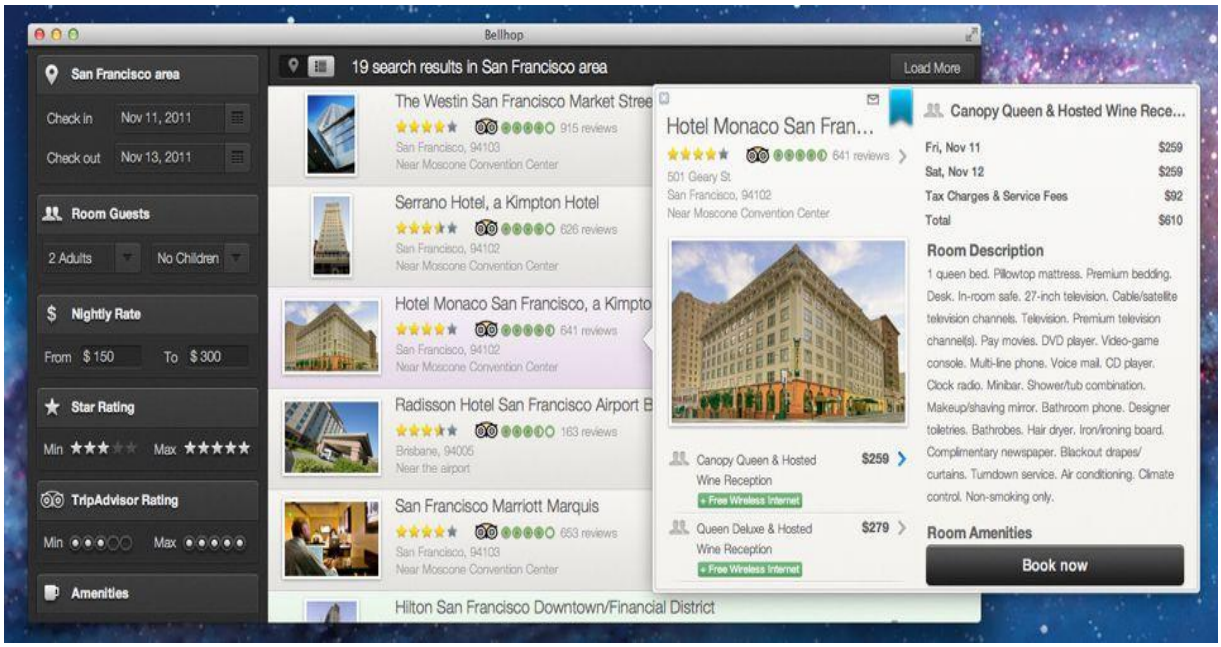


Imagen 15: Bellhop, aplicación específica

2.5.2 LastRoom

Aplicación móvil para realizar reservas de habitaciones a última hora en hoteles de una zona específica.

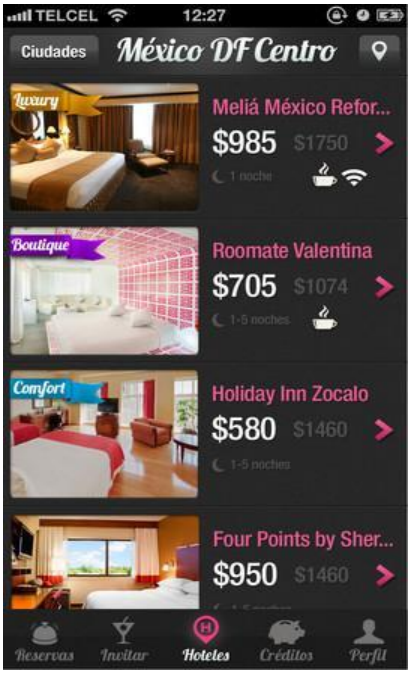


Imagen 16: LastRoom, aplicación específica

2.5.3 Lobby Friend

Lobby Friend es una aplicación que conecta temporalmente a gente que está hospedándose en el mismo hotel o en la misma ciudad. Es la aplicación más parecida a nuestro proyecto, ya que tiene el carácter social que nosotros queremos dar al proyecto, aunque le falta el carácter interactivo con el propio hotel.

Cuando un usuario llega a un nuevo hotel o una nueva ciudad podrá ingresar en *Lobby Friend* y pertenecerá a la red durante su tiempo de estancia en dicho hotel o ciudad. Cuando abandona el hotel o ciudad dejará de tener acceso a la red. Está dirigido mayoritariamente a personas que han encontrado trabajo en otra ciudad. *Lobby Friend* es una aplicación disponible tanto para *iOS* como *Android*.

Imagen 17: LobbyFriend

A continuación, en la imagen 18, resumimos las características de las aplicaciones estudiadas y las comparamos con Hiss & Acclaim. Las características que ofrecen están en verde y las que no, en rojo.

Aplicación	Información general	Reserva de habitaciones	Modo offline	Social	Comunicación con el hotel
The booking button					
Hotels tab					
Book assist					
Wynn Las Vegas					
ARIA					
Caesars Palace					
New York – New York					
MGM Grand					
Mandalay Bay					
Luxor Las Vegas					
Accor Hotels					
mGuest					
Bellhop					
LastRoom					
LobbyFriend					
Hiss & Acclaim					

Imagen 18: Resumen de las aplicaciones estudiadas

Leyenda: Ofrece esta característica - No ofrece esta característica

3 Objetivos

El objetivo principal del proyecto *Hiss & Acclaim* es, como anteriormente hemos explicado, brindar a los clientes del hotel y al propio hotel, una herramienta que permita la comunicación entre clientes de un establecimiento hotelero y de estos con el propio hotel.

En esta comunicación será muy importante la opinión de los clientes, que podrán constantemente expresar en la aplicación móvil lo que les gusta y lo que no les gusta de su estancia en el hotel.

Con esto se pretende que el hotel pueda resolver en tiempo real cualquier mala experiencia que sufra el cliente aumentando así su satisfacción y la calidad del servicio ofrecido. Esto servirá previsiblemente para conseguir un aumento de clientes que acudan al hotel recurrentemente porque valoran los buenos servicios del mismo y su diferenciación.

Además, la herramienta hará de intermediaria entre el cliente y el hotel de manera que el cliente pueda informarse y gestionar todos los servicios disponibles en el hotel, así como las opciones para su habitación, pagos, etc.

En resumen, por puntos y en orden de prioridad, los objetivos directos de *Hiss & Acclaim* serán:

- La interacción y comunicación social de unos clientes con otros y con el hotel.
- El acceso directo por parte del cliente a la gestión de sus datos con el hotel.
- Mostrar al cliente toda la información del hotel, sus servicios y actividades.
- La evaluación y opinión sobre todos los servicios y características del hotel.

Además, como objetivos indirectos cabe destacar:

- Obtener el título de la carrera Ingeniería Informática.
- Aprender a programar para Android.

- Desarrollar un proyecto propio de cierta envergadura con posible orientación comercial.

4 Metodología

Para implementar este proyecto hemos escogido una metodología de desarrollo incremental.

El desarrollo incremental es un proceso de desarrollo de software que fue creado en respuesta a las debilidades del modelo tradicional en cascada.

La idea principal de este mejoramiento iterativo es desarrollar un sistema de programas de manera incremental, permitiéndole al desarrollador sacar ventaja de lo que se ha aprendido a lo largo del desarrollo anterior, incrementando versiones entregables del sistema. El aprendizaje viene de dos vertientes: el desarrollo del sistema y su uso (mientras sea posible).

El procedimiento clave para comenzar con este tipo de metodología es una implementación simple de los requerimientos del sistema, e iterativamente mejorar la secuencia evolutiva de versiones hasta que el sistema completo esté implementado. En cada iteración, se realizan cambios en el diseño y se agregan nuevas funcionalidades y capacidades al sistema.

El proceso consta de dos etapas, etapa de inicialización y etapa de iteración.

En la etapa de inicialización se crea una versión básica del sistema, cuyo objetivo es tener un producto con el que el usuario pueda interactuar y retroalimentar el proceso. Debe contener los aspectos claves del problema y proveer una solución lo suficientemente simple para ser entendida e implementada fácilmente.

Para la etapa de iteración necesitaremos una lista de control de proyecto, la cual contiene un historial de todas las tareas que hay que hacer. Incluye elementos como nuevas funcionalidades para ser implementadas, áreas de rediseño, etc. Debe ser revisada periódicamente.

En la etapa de iteración se involucra el rediseño e implementación de una tarea de la lista de control de proyecto y el análisis de la versión más reciente del sistema. La meta del diseño e implementación de cualquier iteración es ser simple, directa y

modular para poder soportar el rediseño de la etapa o una tarea añadida a la lista de control de proyecto. El código puede representar la mayor fuente de documentación del sistema.

El análisis de una iteración se basa en dos aspectos, la retroalimentación del usuario y el análisis de las funcionalidades disponibles del programa. Involucra el análisis de la estructura, modularidad, usabilidad, confiabilidad, eficiencia y eficacia. La lista de control del proyecto se modifica según los resultados de este análisis, tal y como se puede observar en la imagen 18.

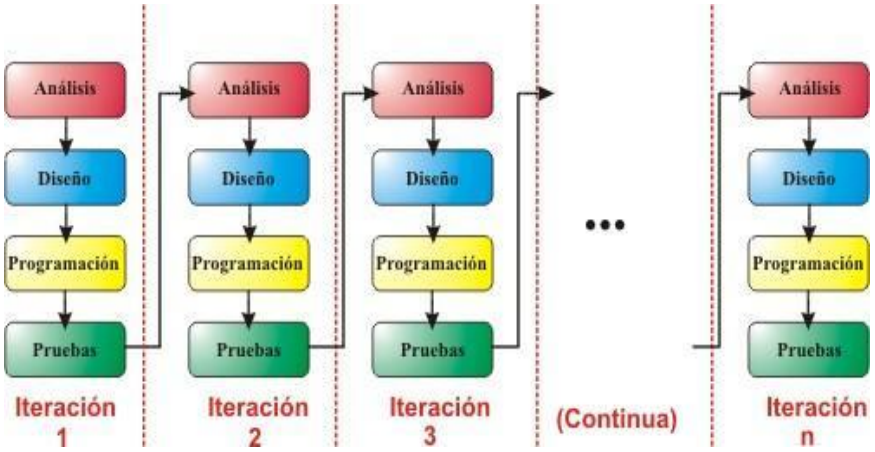


Imagen 19: Modelo de desarrollo incremental

Las guías primarias que guían la implementación y el análisis incluyen:

- Cualquier dificultad en el diseño, codificación y prueba de una modificación debería apuntar a la necesidad de rediseñar o recodificar.
- Las modificaciones deben ajustarse fácilmente a los módulos fáciles de encontrar y a los aislados. Si no es así, entonces se requiere algún grado de rediseño.
- Las modificaciones a las tablas deben ser especialmente fáciles de realizar. Si dicha modificación no ocurre rápidamente, se debe aplicar algo de rediseño.
- Las modificaciones deben ser más fáciles de hacer conforme avanzan las iteraciones. Si no es así, hay un problema primordial usualmente encontrado en un diseño débil o en la proliferación excesiva de parches al sistema.
- Los parches normalmente deben permanecer solo por una o dos iteraciones. Se hacen necesarios para evitar el rediseño durante una fase de implementación.

- La implementación existente debe ser analizada frecuentemente para determinar qué tal se ajusta a las metas del proyecto.
- Las facilidades para analizar el programa deben ser utilizadas cada vez para ayudar en el análisis de implementaciones parciales.
- La opinión del usuario debe ser solicitada y analizada para indicar deficiencias en la implementación referida por él.

5 Recursos necesarios

5.1 Recursos hardware

- Servidor de alta disponibilidad, con un sistema operativo de la familia GNU/Linux instalado y con la siguiente configuración software:
    - Servidor Web Apache, versión 2.0.
    - Intérprete PHP versión 5.
    - Sistema de gestión de bases de datos *Mysql* versión 5.
- Dos equipos con sistema operativo Windows 7.
- Tablet Android con la versión *Ice Cream Sandwich* 4.0.4.

5.2 Recursos software

- Entorno de desarrollo Android Studio desarrollado por Google.
- Emulador de Android llamado *GenyMotion*.
- Cliente FTP *Filezilla*.
- *KITTY*, cliente SSH.
- *XAMPP,* como servidor de pruebas en local.
- Librerías aplicación Android:
    - *v7 appcompat library:* Librería de soporte de Android relacionada con la interfaz y barra de menú superior.
    - *Apache HttpClient:* Librería usada para la gestión de las transferencias de datos entre cliente y servidor.
    - *Universal Image Loader*: Librería para cargar imágenes desde su URL.
    - *PhotoView:* Librería que permite hacer zoom sobre una imagen.
    - *Parse:* Librería que sirve para conectar al servicio de Parse y realizar gestiones relacionadas con las notificaciones push.
    - *Google Play Services:* librería para implementar un mapa de google con la ubicación del hotel.
- Librerías panel web:
    - *Bootstrap:* Librería para desarrollo rápido de interfaces web.
    - *jQuery:* Librería de JScript útil para implementar peticiones AJAX.

- *Dynatable:* Librería para apoyar en la generación de tablas dinámicas a raíz de datos JSON.

- Librerías Servidor:
  - *MCrypt:* Librería necesaria que se utiliza para desencriptar las peticiones enviadas por el cliente de la aplicación Android.
  - *fileUpload:* Librería utilizada para subir imágenes al servidor desde el cliente Android al servidor.
  - *processUpload:* Librería para permitir la subida de imágenes desde el panel web al servidor.
- Sistema de control de versiones Git con un repositorio en *Github*.

5.2.1 Herramientas empleadas para la documentación de este proyecto

- Microsoft Word 2010.
- Microsoft Excel 2010.

5.3 Estudio de las tecnologías y las herramientas

5.3.1 Tecnologías

¿Por qué Android y no iOS?

Hemos analizado estadísticas que muestra el mercado de la telefonía para decidir entre programar para Android o iOS. En uno de los últimos informes de la consultora IDC para el primer trimestre de 2013, podemos asegurar que Android gusta en Europa, como se puede ver en las imágenes 19 y 20, obtenidas de "http://www.xatakandroid.com/mercado/android-comanda-con-mano-de-hierro-el-mercado-europeo".

Top Western European Mobile Phone Vendors, Total Shipments and Market Share, 1Q13 Smartphones and Feature Phones (Units in Millions)

Vendor	1Q13 Unit Shipments	1Q13 Market Share	1Q12 Unit Shipments	1Q12 Market Share	1Q13/1Q12 Change
1. Samsung	19.9	46%	18.1	40%	10%
2. Apple	6.2	14%	7.0	15%	-11%
3. Nokia	6.1	14%	8.7	19%	-30%
4. Sony	3.2	7%	1.9	4%	64%
5. LG	2.5	6%	0.9	2%	178%
6. Others	5.7	13%	8.9	20%	-36%
Total	**43.6**	**100%**	**45.5**	**100%**	**4%**

Imagen 20: Datos principales vendedores

En el informe se destaca la fuerza con la que sigue creciendo el sistema operativo de Google, que solamente con los teléfonos inteligentes de la empresa Samsung, tiene un 46% del mercado, a lo que si le sumamos los otros dispositivos de otras empresas podemos observar el dominio de este sistema operativo.

El crecimiento de Android es innegable, pues en el cuarto trimestre de 2012 contaba con un 55% de share, aunque realmente donde se comprueba el dominio es en los rivales, pues iOS y Apple bajan de un 25% al 20% en el mismo período.

Sin embargo, parece que el mercado empieza a saturarse, y la crisis europea tampoco es que ayude en este sentido a hacer crecer las ventas, pues desde IDC se asegura que ha sido el trimestre con menor crecimiento desde 2004. En total se han vendido 31.6 millones de teléfonos, un 12% más que en el primer trimestre de 2012.

También puede influir en la caída de las ventas el final de las subvenciones en países como el nuestro, uno de los que mayores registros de venta de teléfonos inteligentes presentaba, y donde la mayoría compraban sus teléfonos a través de las operadoras, firmando largas permanencias a cambio de un precio menor por el dispositivo.

Top Western European Mobile Phone Vendors, Shipments and Market Share, 1Q13 Smartphones (Units in Millions)

Vendor	1Q13 Unit Shipments	1Q13 Market Share	1Q12 Unit Shipments	1Q12 Market Share	1Q13/1Q12 Change
1. Samsung	14.3	45%	10.9	39%	31%
2. Apple	6.2	20%	7.0	25%	-11%
3. Sony	3.2	10%	1.6	6%	100%
4. LG	2.4	8%	0.5	2%	380%
5. Nokia	1.6	5%	2.3	8%	-30%
6. Others	3.9	12%	5.9	20%	-34%
Total	**31.6**	**100%**	**28.2**	**100%**	**12%**

Imagen 21: Datos sobre crecimiento de las empresas

En el caso de los fabricantes, Samsung continúa a su ritmo, creciendo hasta el 45% de cuota de mercado; mientras que su principal punto de mira, Apple, decrece un 11% hasta el 20% de cuota, manteniendo el segundo puesto.

Por todo esto vamos a diseñar la aplicación para sistemas Android debido al mayor volumen de uso de este sistema, que además sigue aumentando.

¿Por qué Java y no otro lenguaje de programación?

Quizás para alguien inmerso en la programación para Android la pregunta no tenga mucho sentido ya que Java es el lenguaje nativo para programar sobre el sistema operativo Android y por eso tiene todas las de ganar. No obstante, existen muchas herramientas y lenguajes distintos que permiten programar para Android sin saber Java. Hemos analizado estas alternativas debido a que en algunos casos pueden tener una curva de aprendizaje menor o dar facilidades en ciertos aspectos y para ciertos nuevos desarrolladores.

Algunas de las herramientas alternativas que hemos revisado son las que pasamos a detallar a continuación explicando sus puntos fuertes.

Basic4Android

Basic4Android es una plataforma de programación para aplicaciones Android cuyo lenguaje base de programación es VisualBasic, el eterno rival de Java. VisualBasic es un lenguaje que está orientado a aquellas personas que empiezan en el mundo de la

programación de una manera más gráfica y no tan abstracta. No es exactamente VisualBasic pero su sintaxis es la misma, lo cual tiene sus mismas ventajas así como algunos de sus inconvenientes.

Esta herramienta tiene a su favor que la curva de aprendizaje es prácticamente nula si ya sabes VisualBasic, y que da facilidades a las personas que no tienen muchos conocimientos en programación.

Sin embargo tiene en contra las limitaciones propias de VisualBasic como son las dificultades de gestión de memoria dinámica o la falta de ciertos mensajes de advertencia en el compilador. También va en su contra que es una herramienta de pago.

Mono para Android

Mono permite programar para Android en C# y .NET. Al igual que en el caso de Basic4Android su punto fuerte es que si ya sabes C# o .NET la curva de aprendizaje es prácticamente nula. Sin embargo la licencia de uso es muy costosa con lo que incluso en el caso de saber C# o .NET puede seguir sin parecernos una buena opción, a no ser que necesitemos programar una aplicación con urgencia o pensemos reutilizar código escrito en uno de esos lenguajes.

APP Inventor

Esta es una plataforma de desarrollo orientada a personas que no han tenido contacto con la programación, ni quieren tenerlo. Permite programar gráficamente mediante el uso de bloques lógicos con funciones definidas. Obviamente es la opción que debe escoger alguien que no haya programado nunca, pero para desarrollos serios que necesiten complejidad no es una opción oportuna ya que no da demasiada libertad.

LiveCode

Esta es una opción interesante. Es una plataforma de programación orientada a eventos que permite exportar un mismo código a distintos entornos y dispositivos. Se programa en base a los eventos que se lanzan mediante cada instrumento de la interfaz gráfica.

Claramente su punto fuerte es que permite exportar un mismo código para ser usado sobre distintas plataformas, por otro lado esto conlleva ciertas limitaciones al

programar y pérdida de rendimiento durante la ejecución. Además también tiene en su contra que es una opción de pago.

5.3.2 Herramientas

¿Por qué Android Studio y no eclipse u otro entorno?

Primeramente habíamos tomado la decisión de usar Eclipse frente a cualquier otro entorno, ya que es un entorno de programación con el que estamos muy familiarizados porque hemos desarrollado multitud de aplicaciones sobre él. Para nuestro gusto eclipse es un entorno que permite un desarrollo cómodo incluso para desarrollar los programas más complejos y no nos planteamos ninguna otra opción. Sin embargo, descubrimos Android Studio.

Android Studio es un entorno para programar en Java para Android que ha desarrollado Google, lo cual ya pone de manifiesto que puede ser una de las mejores opciones para desarrollar en Android. Tras analizar la herramienta hemos podido comprobar que tiene ciertas ventajas frente a Eclipse, que se irán incrementando a medida que la herramienta vaya actualizándose. Algunas de estas ventajas son, entre otras, la posibilidad de probar cómo se vería nuestra aplicación en diferentes resoluciones automáticamente, desde el mismo IDE sin necesidad de compilar, además de la posibilidad de probarla en múltiples dispositivos virtuales.

Sin duda actualmente es una de las mejores opciones para programar para Android y también consideramos que tiene mucho potencial, con lo cual tenderá a mejorar con cada actualización.

6 Plan de trabajo y temporización

Atendiendo a nuestra experiencia en otros proyectos hemos realizado las siguientes estimaciones:

Etapa	Horas
Análisis	300
Diseño	400
Implementación	1000
Validación y manuales	150
Presentación del proyecto	25
TOTAL DE HORAS PLANIFICADAS	1875

Imagen 22: Estimación de horas de trabajo

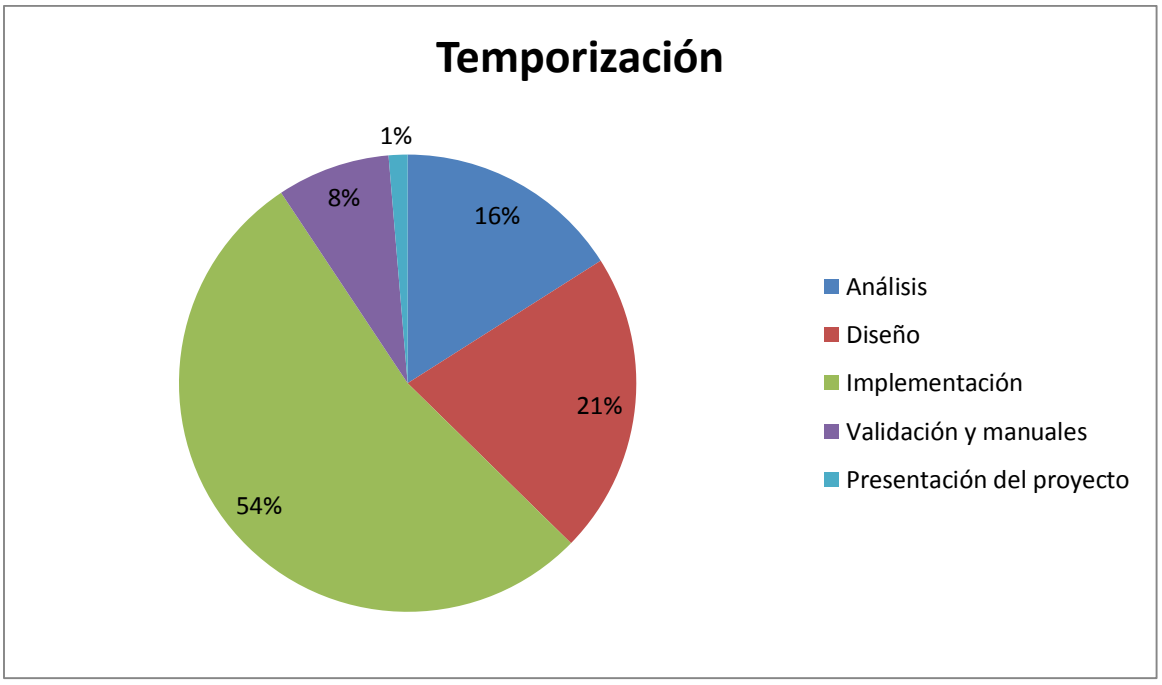


Imagen 23: Diagrama de temporización del trabajo

7 Análisis

7.1 Determinación del Alcance del Sistema

El software a realizar va a ser una aplicación para el sistema operativo Android que estará disponible gratuitamente en la Play Store e instalará el cliente del hotel en su dispositivo móvil.

Esta aplicación tendrá como objetivo dar al cliente información del hotel, así como permitir al cliente comunicarse con el hotel y con otros clientes. También permitirá que el cliente vea las actividades y servicios a través de la aplicación, pudiendo valorar positiva o negativamente los mismos, con lo que se da información vital a los administradores del complejo hotelero.

7.1.1 Requisitos

Los requisitos primarios del sistema son:

- Aplicación para Android.
- Panel de control web para administradores.
- Comunicación entre clientes.
- Posibilitar la comunicación con el servicio de habitaciones.
- Posibilitar la gestión de datos personales del cliente.
- Integración de servicios hoteleros.
- Integración de actividades del hotel.
- Modificación de los datos personales del usuario.
- Categorización de comentarios.

Los requisitos secundarios del sistema son:

- Comentarios sobre servicios y actividades (tanto positivos como negativos).
- Valoraciones sobre servicios, actividades y comentarios (tanto positivas como negativas).
- Ver el perfil de datos de otros usuarios y sus comentarios.

7.1.3 Identificación del Entorno Tecnológico

El entorno tecnológico en el que vamos a desarrollar nuestra aplicación será en el sistema operativo de software libre Android, que será el sistema que estará instalado en los dispositivos móviles de los usuarios.

La principal característica necesaria es que el dispositivo móvil tenga al menos el sistema operativo Android Ice Cream Sandwich, especificamente desde la versión 4.0.3 en adelante.

También se podrá acceder mediante un ordenador con cualquier sistema operativo y navegador a un panel de administración web realizado en HTML, CSS y JavaScript desde el cual los administradores podrán gestionar los aspectos importantes de la aplicación, ya que la comodidad de un panel web es superior a la de un dispositivo móvil.

Además se utilizará un servidor con Linux y una base de datos MySQL.

7.1.4 Identificación de los Usuarios Participantes y Finales

- Jefes de Proyecto (Tutores)
    - Alexis Quesada Arencibia
    - Agustín Jesús Sánchez Medina

- Analistas y Desarrolladores
    - Jorge Mor Varela
    - Borja Castillo Hernández

- Cliente
    - Empresa hotelera

- Usuarios Finales
    - Clientes del hotel
    - Administradores

7.2 Establecimiento de requisitos

Se establecerán los requisitos del proyecto

7.2.1 Obtención de Requisitos

7.2.1.1 Catálogo de requisitos

A continuación hemos desarrollado un catálogo de requisitos según su tipo:

Funcionales

- Gestión de los datos propios por parte del cliente.
- Integración de la información de servicios hoteleros como son restauración, aparcacoches y otros.
- Integración de la información de actividades como son excursiones, fiestas y otros.
- Comunicación con otros clientes y expresión de opiniones.
- Comunicación con el servicio de habitaciones.
- Visualización de datos básicos del hotel.
- Visualización de un mapa con la ubicación del hotel.
- Recepción de notificaciones sobre nuevos mensajes recibidos.
- Gestión de datos del hotel y usuarios por parte del administrador del hotel.

Rendimiento

- Programación para dispositivo móvil Android Ice Cream Sandwich 4.0.3 en adelante.

Seguridad

- Cifrado AES

Implantación

- Interfaz amigable.
- Volcado o adaptación de la base de datos anterior.

Disponibilidad del sistema

El sistema estará disponible las 24 horas durante la estancia del cliente a excepción de las muy poco probables interrupciones imprevistas. Las opciones y servicios disponibles en la aplicación podrán estar limitados en función de las necesidades del hotel.

7.2.2 Modelo de casos de uso

En los siguientes apartados analizaremos los casos de uso en profundidad y representaremos en varios diagramas de casos de uso el conjunto de casos necesarios para llevar a cabo los requisitos del proyecto. Éstos los mostraremos según los siguientes actores:

- Usuario
- Administrador

Además, agruparemos los casos de uso por módulos. Estos módulos serán los siguientes:

- General
- Social
- Actividades
- Servicios
- Restauración
- Datos cliente

7.2.2.1 Casos de uso y descripción

A continuación realizamos un resumen explicativo de las acciones que podrá llevar a cabo cada actor en cada categoría de la aplicación. Los casos de usos concretos a desarrollar aparecerán subrayados en la descripción de acciones posibles de cada actor. Además se adjunta un diagrama resumen de cada módulo para cada actor.

Categoría general

- Administrador

El administrador será el encargado de añadir, y eliminar las imágenes de la galería del hotel. También podrá realizar las funciones de usuario descritas a continuación.

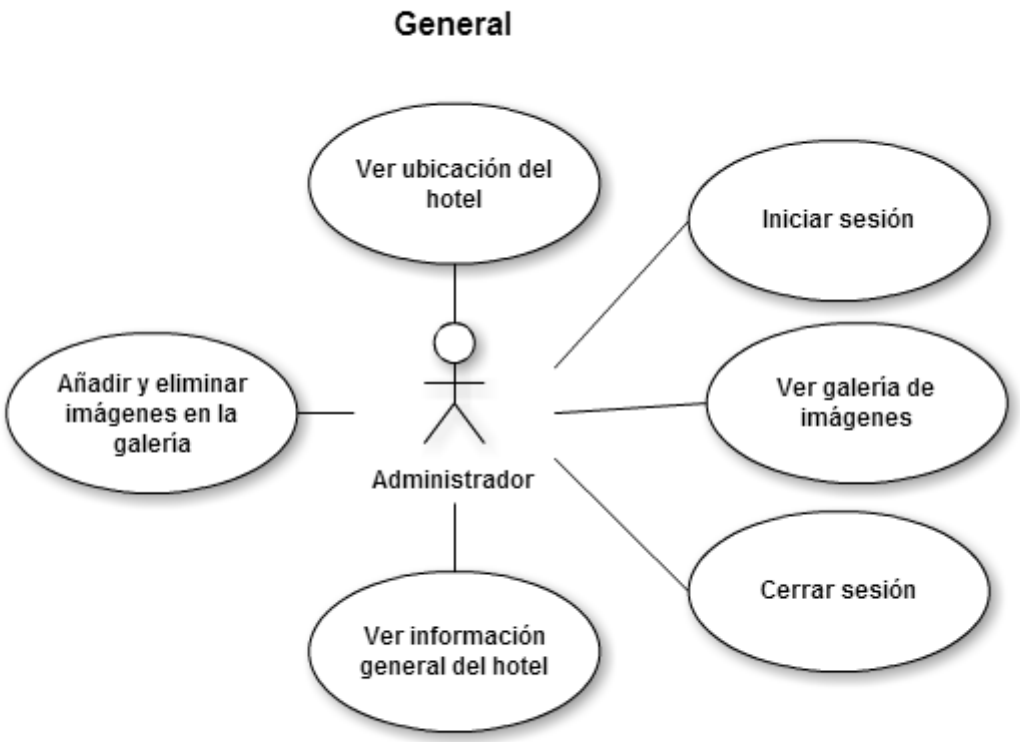


Imagen 24: Diagrama de casos de uso de Administrador generales

- Usuario

El usuario podrá, en primer lugar, iniciar sesión. Además, podrá cerrar su sesión, ver la galería de imágenes, la información general del hotel, la ubicación del mismo, los datos y opciones de su habitación y modificar estos últimos a través de una pestaña disponible en el software en el menú principal.

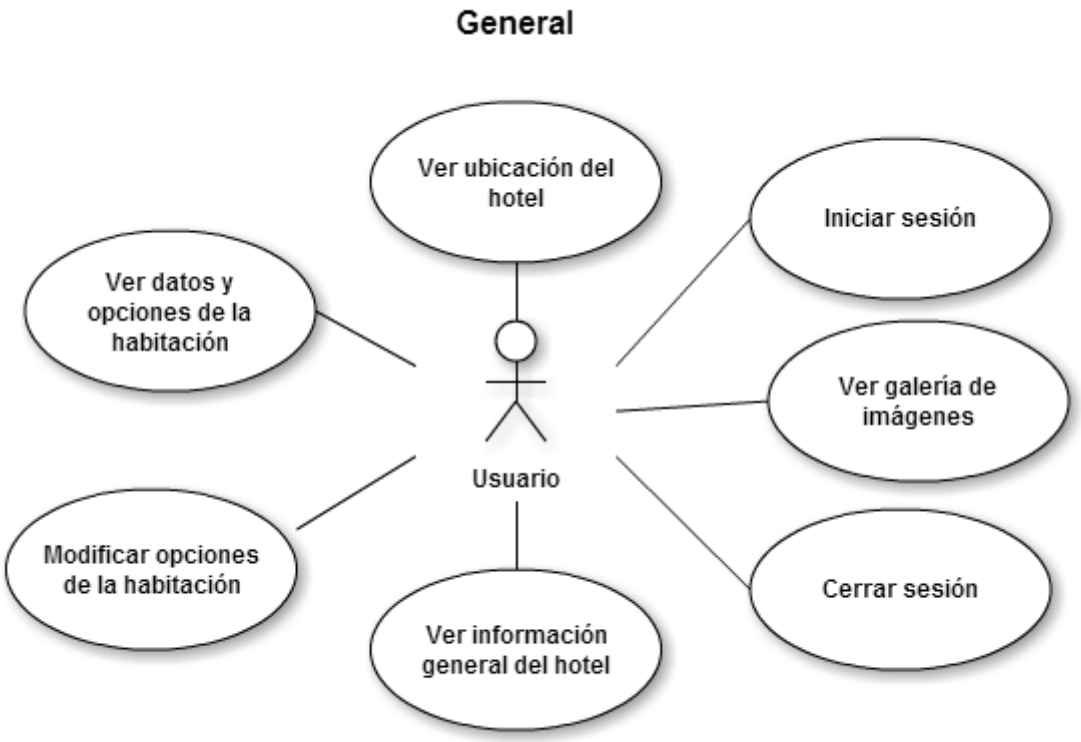


Imagen 25: Diagrama de casos de uso de Usuario generales

Categoría Actividades

- Administrador

El administrador será el encargado de añadir, editar y eliminar las actividades del hotel. También podrá ver los comentarios sobre una actividad concreta.

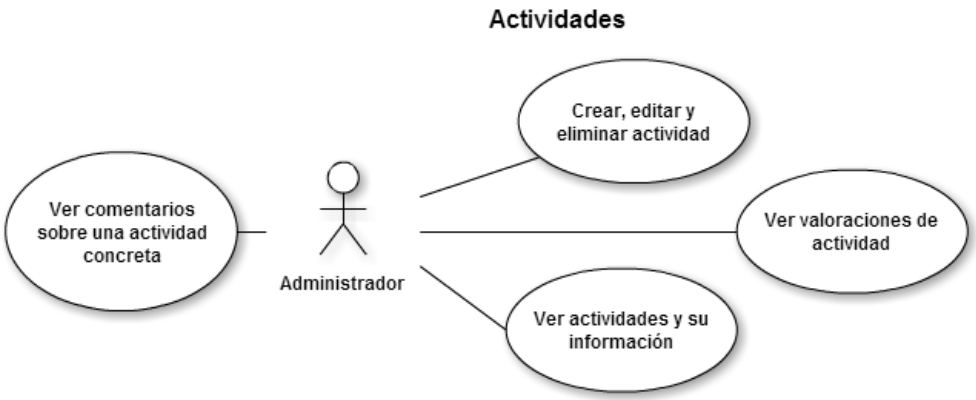


Imagen 26: Diagrama de casos de uso de Administrador para actividades

- Usuario

El usuario podrá ver las actividades disponibles en el hotel a través de una pestaña disponible en el software en el menú principal. Picando en cada actividad tendrá acceso a la información detallada sobre la actividad y los comentarios de otros usuarios sobre la misma. Por último, puede valorar las actividades.

Imagen 27: Diagrama de casos de uso de Usuario para actividades

Categoría Restauración

- Administrador

Podrá crear, editar o eliminar una carta. Además será el encargado de seleccionar la carta que está activa en un momento determinado.

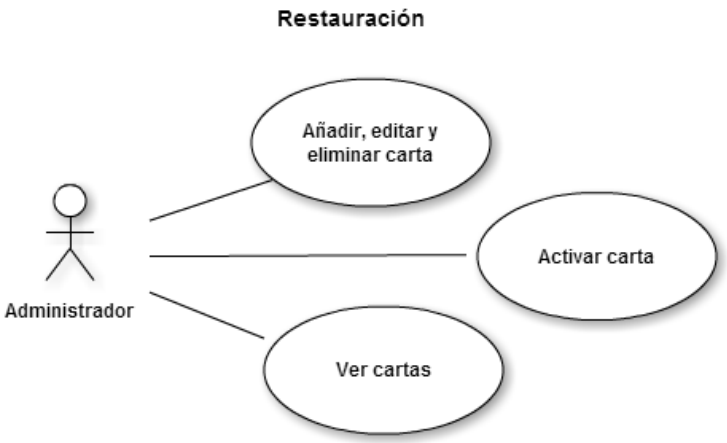


Imagen 28: Diagrama de casos de uso de Administrador para restauración

- Usuario

  Podrá ver la carta del restaurante.

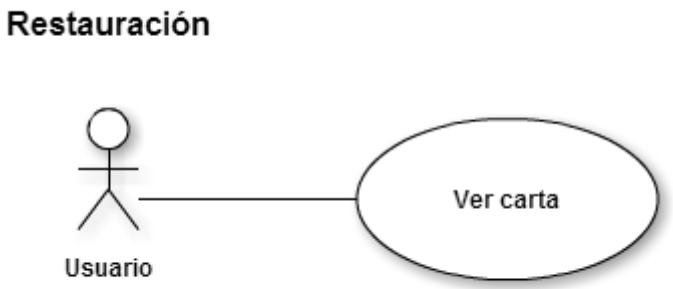


Imagen 29: Diagrama de casos de uso de Usuario para restauración

Categoría Servicios

- Administrador

El administrador será el encargado de añadir, editar y eliminar los servicios del hotel. También podrá ver los comentarios sobre un servicio concreto.

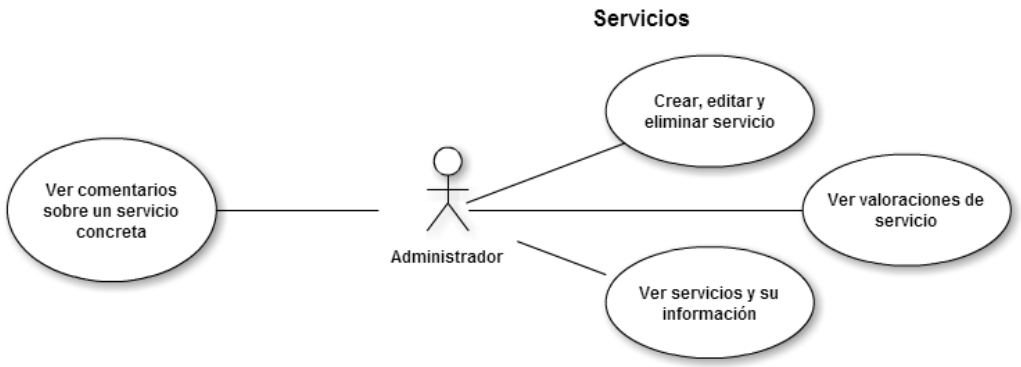


Imagen 30: Diagrama de casos de uso de Administrador para servicios

- Usuario

El usuario podrá ver los servicios disponibles en el hotel a través de una pestaña disponible en el software en el menú principal. Picando en cada servicio tendrá acceso a la información detallada y los comentarios de otros usuarios sobre el mismo. Por último, puede valorar los servicios.

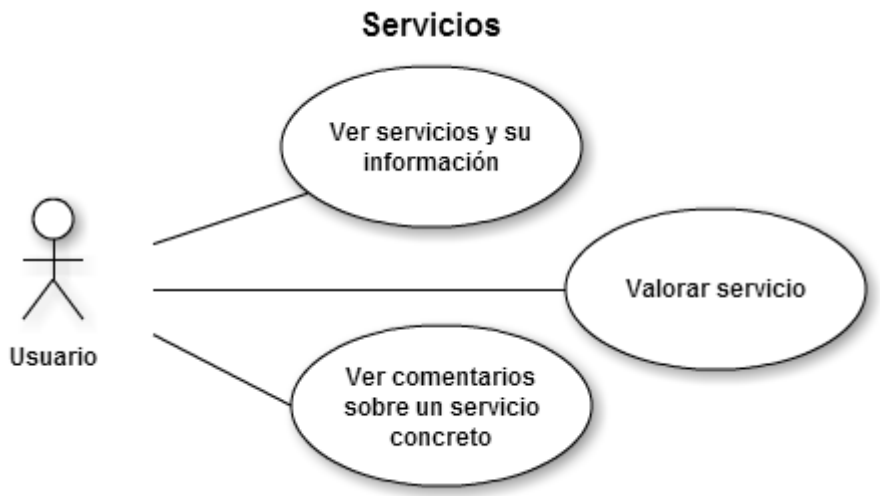


Imagen 31: Diagrama de casos de uso de Usuario para servicios

Categoría Datos Cliente

- Administrador

Será el encargado de crear, ver, editar y eliminar los datos personales del cliente. Entre ellos está la foto de perfil.

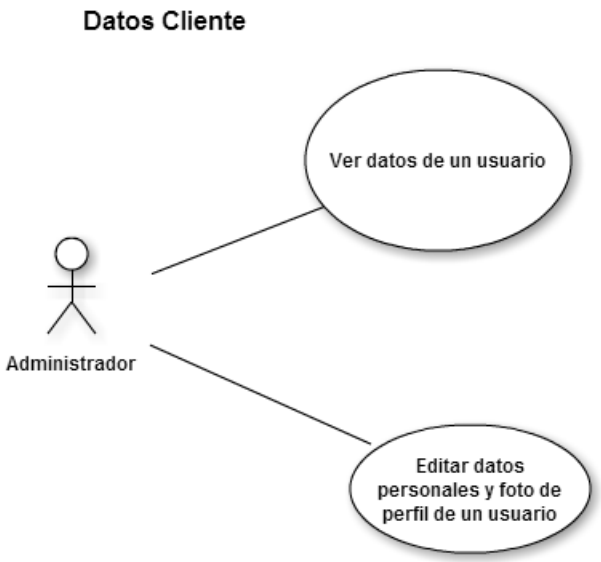


Imagen 32: Diagrama de casos de uso de Administrador para datos del cliente

- Usuario

El usuario podrá en todo momento consultar sus datos personales y editarlos.

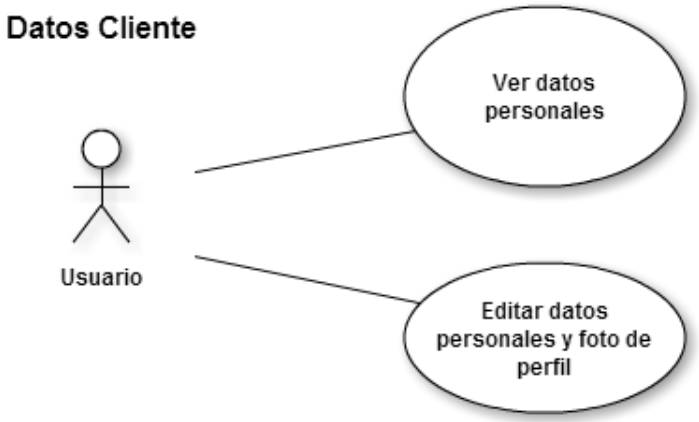


Imagen 33: Diagrama de casos de uso de Usuario para datos del cliente

Categoría Social

- Administrador

Atenderá las demandas privadas del cliente mediante la atención del canal de chat privado cliente-servicio de habitaciones. Además, se le asignará una cuenta de cliente con el perfil del hotel para que pueda comentar y funcionar en la red social como otro usuario más defendiendo la imagen del hotel ante las críticas negativas que puedan surgir.

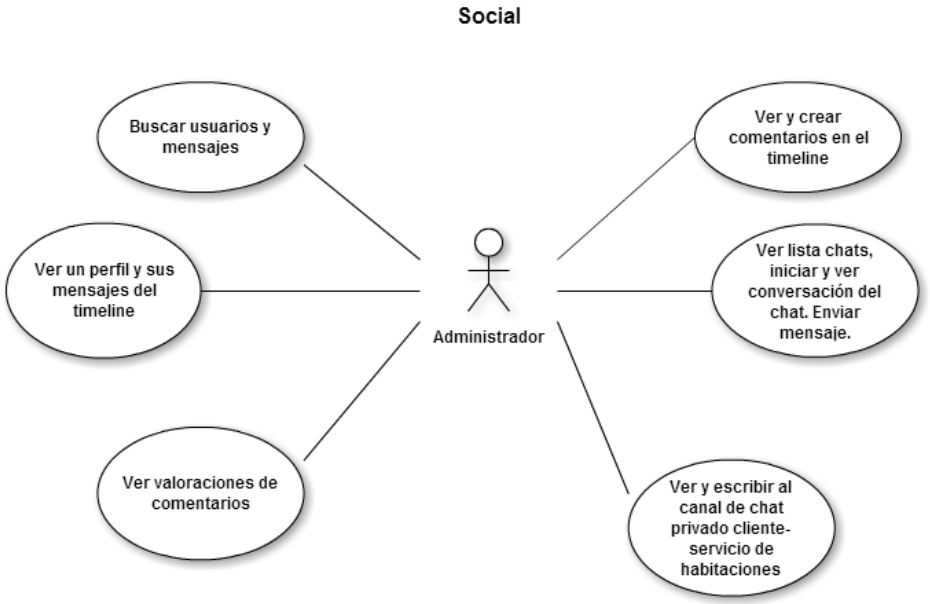


Imagen 34: Diagrama de casos de uso de Administrador social

- Usuario

Podrá configurar su perfil y también buscar, los perfiles de los demás usuario. También podrá modificar su estado. Podrá crear comentarios en el historial de mensajes (también conocido como *timeline*) con sus correspondientes opciones (comentario positivo o negativo, categoría, etc.). Podrá iniciar y continuar conversaciones en el chat/área de mensajes. Podrá también escribir al canal de chat privado cliente-servicio de habitaciones. Por último, también tiene la capacidad de valorar los mensajes del timeline de otros usuarios.

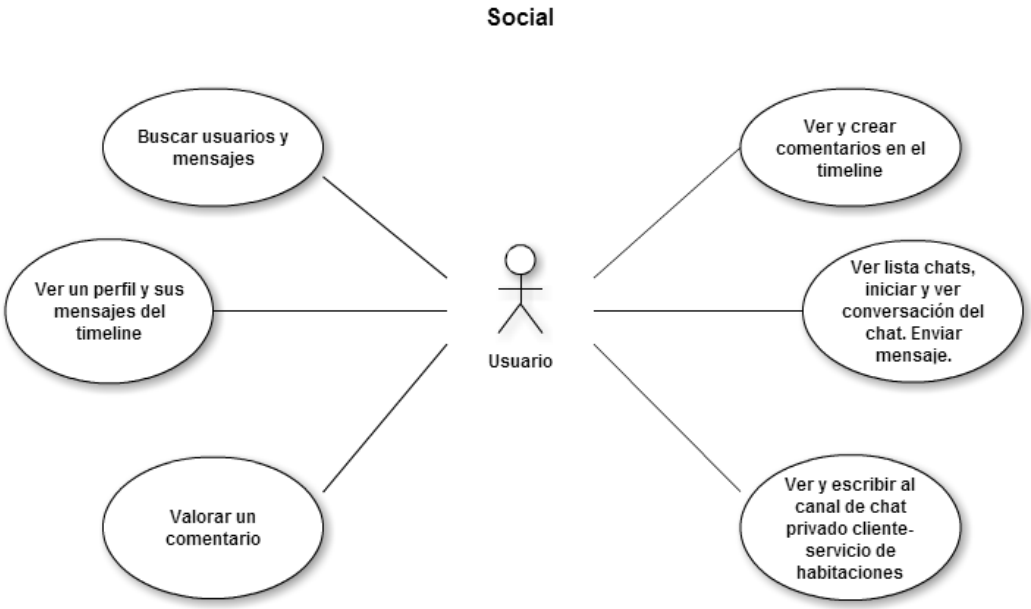


Imagen 35: Diagrama de casos de uso de Usuario social

7.2.3 Especificación de Casos de Uso

La especificación de los casos de uso ha sido añadida como anexo al final del documento. Específicamente se encuentra en el anexo IV.

7.3 Prototipo de interfaz de usuario en Android

A continuación expondremos los prototipos principales de la interfaz.

7.3.1 Timeline

Imagen 36: Prototipo TimeLine

7.3.2 Añadir publicación

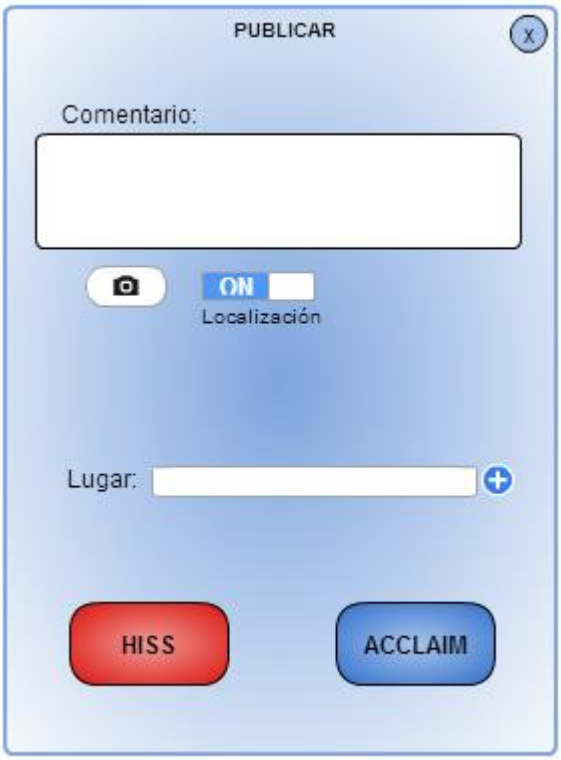


Imagen 37: Prototipo añadir publicación

7.3.3 Lista de chats

Imagen 38: Prototipo de la lista de chats

7.3.4 Vista interna de chat

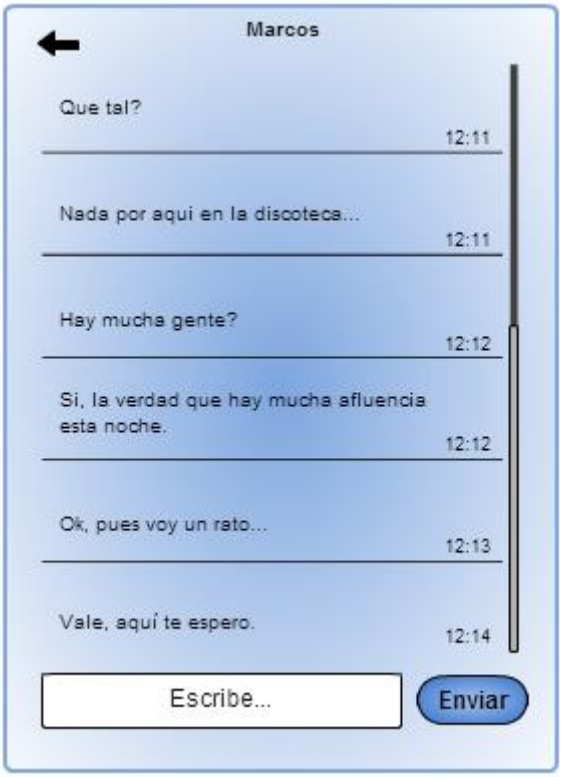


Imagen 39: Prototipo de vista interna de chat

7.3.5 Vista Hotel

Imagen 40: Prototipo de vista hotel

7.3.6 Lista de servicios y lista de actividades

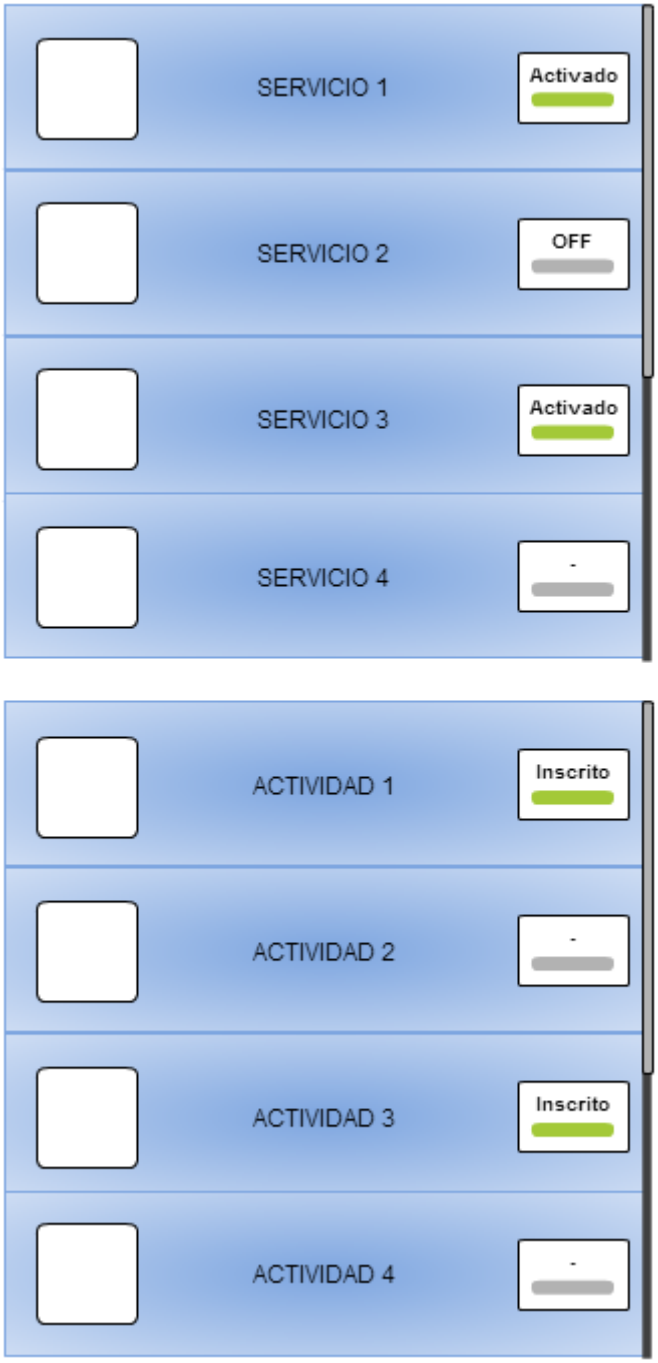


Imagen 41: Vista de servicios y actividades

7.3.7 Vista de ajustes

Imagen 42: Prototipo de vista de ajustes

7.3.8 Vista de edición de perfil

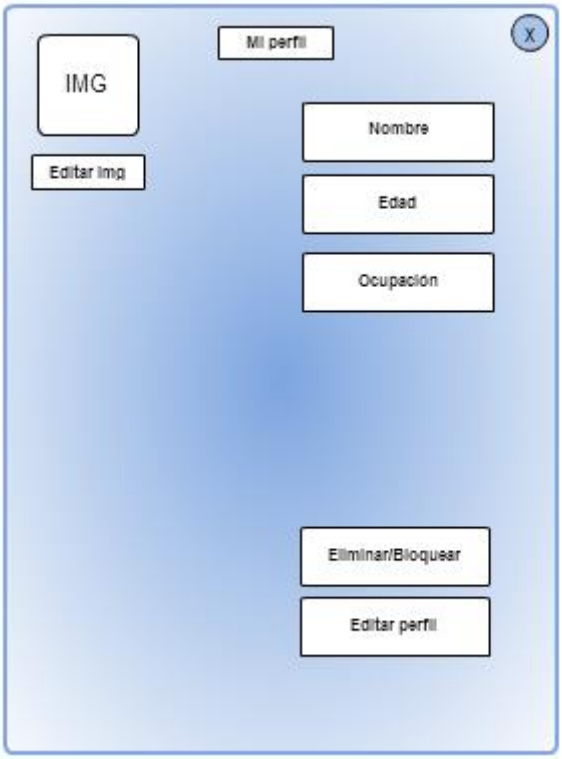


Imagen 43: Prototipo de vista de edición de perfil

8 Diseño

En este apartado mostraremos el modelado y la estructura del sistema desarrollado para poder realizar la implementación del proyecto.

8.1 Diseño de la arquitectura

Para la realización del proyecto se han implementado varios subsistemas. A continuación determinaremos cuales son y mostraremos su organización e interconexión.

8.1.1 Identificación de nodos y conexiones

Después de identificar, en los casos de uso, cómo interactúan los actores con el sistema, podemos identificar los nodos y conexiones necesarias para realizar la implementación. En el siguiente diagrama de despliegue podemos visualizarlo.

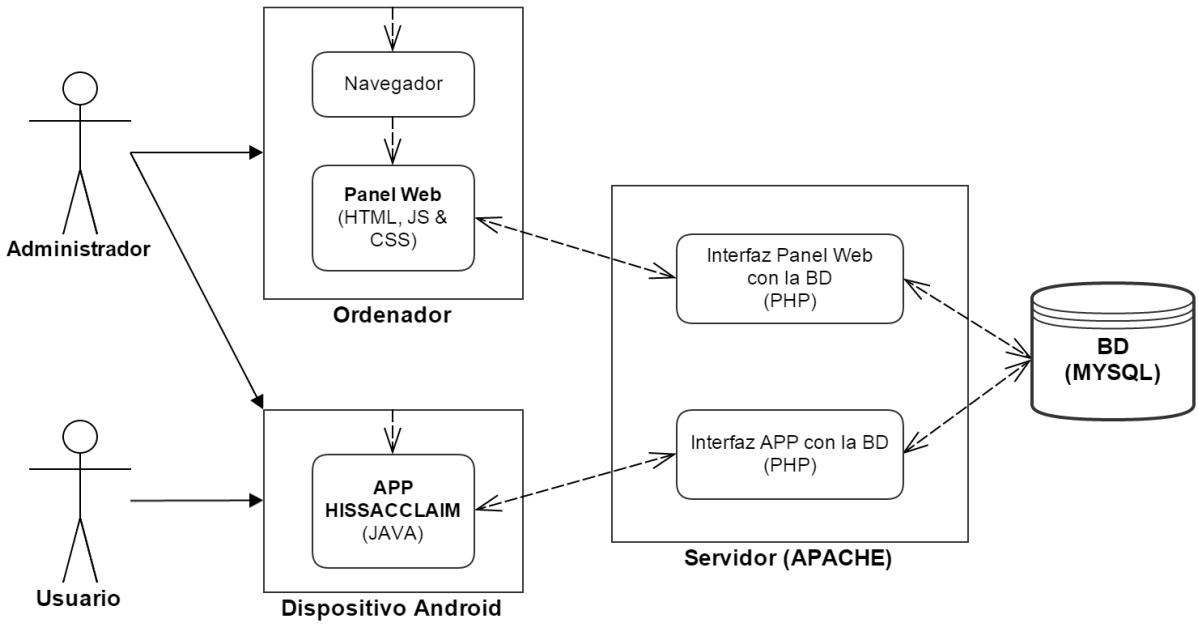


Imagen 44: Diagrama de despliegue

Como podemos observar, la arquitectura del sistema implementa el modelo Cliente-Servidor. En tiempo de ejecución existen los siguientes nodos, componentes y enlaces:

Identificamos los nodos:

- Dispositivo Android
- Ordenador (PC o Mac) con navegador

- Servidor Apache
- Base de datos MySQL

Identificamos los componentes:

- Panel Web
- APP HISSACCLAIM
- Interfaz APP con la base de datos
- Interfaz Panel Web con la base de datos

8.1.2 Identificación de subsistemas

Como se ha detallado en el capítulo 8.1.1 de este mismo documento, la arquitectura se corresponde con el modelo Cliente-Servidor y aparecen los siguientes subsistemas:

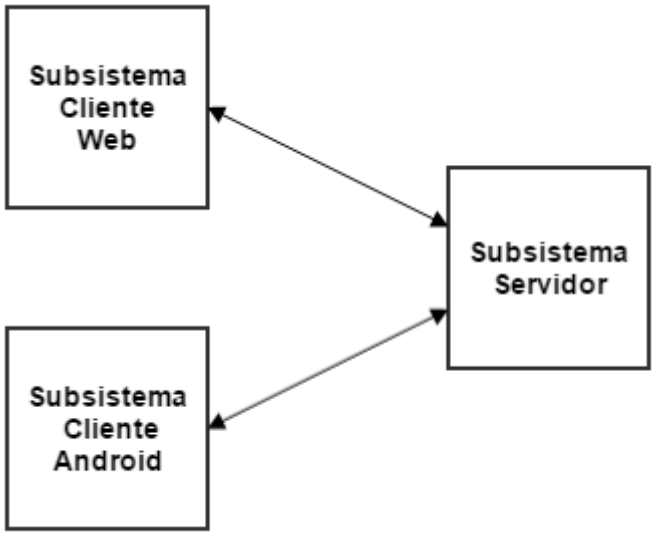


Imagen 45: Diagrama de subsistemas

Como vemos en la ilustración anterior tenemos:

- **Subsistema cliente web:** Será un panel de control programado usando HTML, JAVASCRIPT Y CSS.

- **Subsistema cliente Android:** Será una aplicación nativa de Android programada en JAVA.
- **Subsistema servidor:** Será programado usando PHP. Contará con dos interfaces de comunicación con los subsistemas clientes y nos permitirá acceder a la base de datos.

En los siguientes apartados profundizaremos en mayor medida sobre el diseño de cada subsistema.

8.2 Diseño del subsistema cliente Android

El subsistema cliente Android será el encargado de realizar las peticiones oportunas al subsistema servidor y presentará la información a los usuarios en sus dispositivos Android.

Será programado en nativo para Android usando el lenguaje JAVA y deberá ofrecer las funcionalidades que se analizaron con anterioridad, dentro del sistema que se ha expuesto.

Para ello, necesitará hacer uso de una interfaz de comunicación para obtener y enviar datos desde el servidor o hacia el mismo, respectivamente. En este sentido, se desarrollará una clase que se encargue de enviar las peticiones al servidor y recibir los datos de respuesta.

A continuación mostramos una imagen en la cual exponemos cómo será la estructura del subsistema y las clases que contendrá:

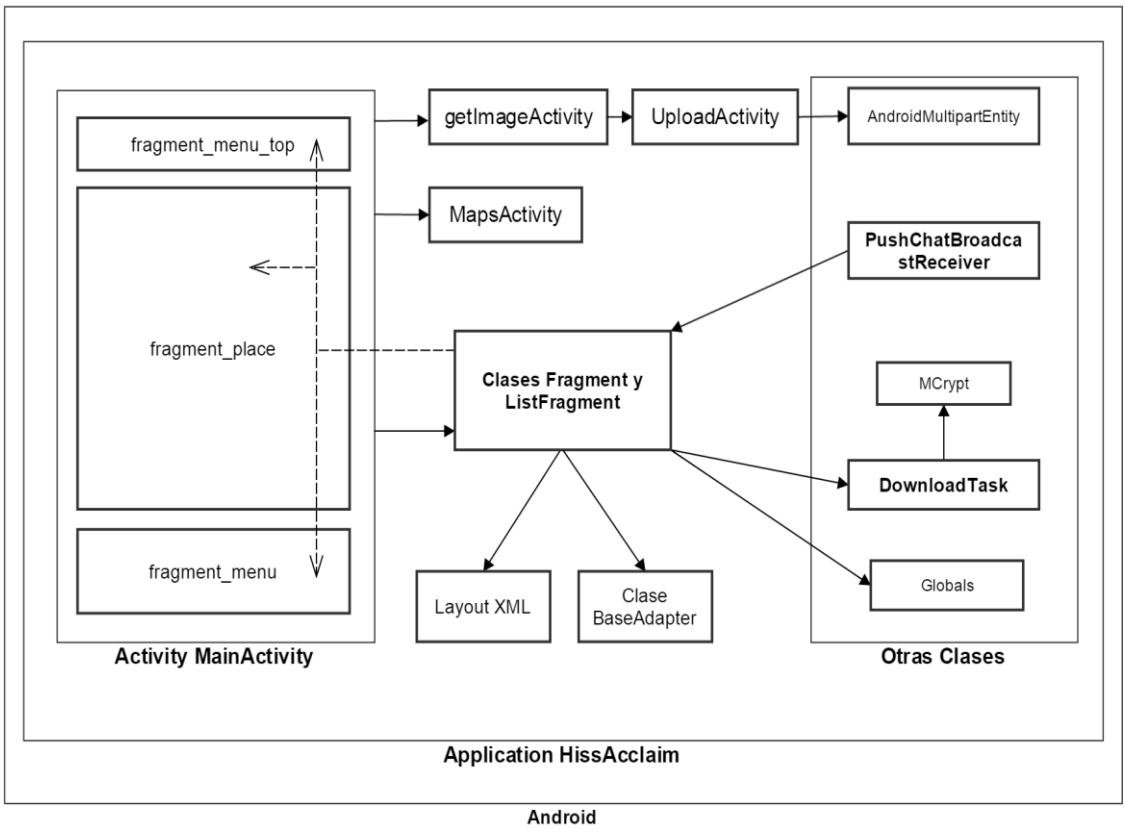


Imagen 46: Estructura del subsistema cliente Android

En la ilustración se puede observar que la aplicación contendrá distintos tipos de clases y también cómo será el uso de unas por parte de otras. Las flechas punteadas simbolizan que los "fragments" se cargarán esas zonas de la interfaz ("menu_top", "place" y "menú") y el resto de flechas simboliza el uso o la activación de una clase por parte de otra.

En los siguientes sub-apartados explicaremos las clases más importantes que se implementarán en el subsistema cliente.

Es importante señalar que el subsistema se implementará de forma que mantenga almacenados siempre localmente los últimos datos que se recibieron del servidor (sólo en el apartado Hotel de la aplicación). De esta manera, en caso de que se pierda la conexión a internet, los usuarios podrán seguir accediendo a estos últimos datos.

8.2.1 Clase Application

Es la clase principal de cualquier aplicación en Android y sirve para mantener el estado general de la aplicación, incluso cuando no está activa. Clases de la aplicación que extenderán de esta clase básica de Android:

- Clase HissAcclaim: Extiende de la clase Application explicada anteriormente. Permitirá recibir las notificaciones push de la aplicación así como consultar el estado y el contexto de la Activity MainActivity. Vemos su detalle en la siguiente imagen:

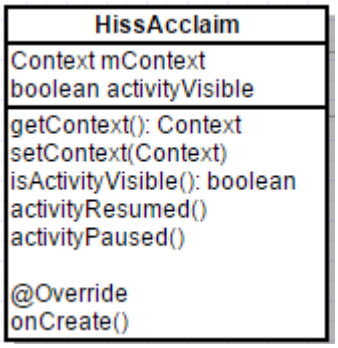


Imagen 47: Clase HissAcclaim

8.2.2 Clase Activity

Esta clase es la que permite mostrar elementos en la pantalla e interactuar con ellos en nuestra aplicación pudiendo cargar elementos gráficos desde un xml relacionado. A continuación podemos ver las clases que extenderán de Activity:

- **Clase MainActivity:** Será el activity principal de la aplicación y el padre del resto de activities que se nombran en este apartado. Lo usaremos para capturar y gestionar muchas de las acciones que se pueden realizar en las distintas vistas y para mostrar prácticamente todas las vistas de nuestra aplicación, que estarán organizadas en fragments que se cargaran en las tres posiciones dispuestas por el MainActivity. Las tres posiciones tienen los siguientes nombres:
    - ***fragment_menu_top:*** Se usará para mostrar un menú en lo alto del activity, correspondiente al apartado que estemos visualizando.
    - ***fragment_place:*** Se usará para mostrar el contenido de cada apartado de la aplicación cargando distintos fragments.
    - ***Fragment_menu:*** Se usará para mostrar el menú principal de la aplicación.

Imagen 48: Clase MainActivity

- **Clase MapsActivity:** Será el activity encargado de mostrar el mapa de Google Maps con la ubicación del hotel (librería).
- **Clase getImageActivity:** Será el activity encargado de permitir realizar fotos cuando sea necesario (librería).
- **Clase UploadActivity:** Será el activity encargado de la subida al servidor de las fotos realizadas mediante el getImageActivity (librería).

8.2.3 Clase Fragment

Es una clase que funciona de manera similar a la clase Activity. La diferencia es que un Fragment normalmente es un fragmento de la interfaz de un Activity.

La aplicación tendrá diversos fragments que servirán para mostrar cada vista de la aplicación en las posiciones principales definidas por el MainActivity.

En las clases que implementaremos (que extiendan de la clase Fragment) solamente se actuará sobrescribiendo el método onCreateView() para cargar con ellas vistas. De esta forma, podemos ver la siguiente ilustración de la clase:

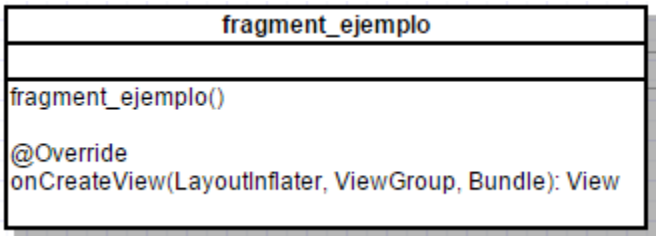


Imagen 49: Clase ejemplo Fragment

Debido al alto número de clases definidas que extienden de la clase Fragment, se englobarán en bloques explicativos aquellas que guardan grandes similitudes en su manera de funcionamiento. El nombre de cada clase se identifica con la vista sobre la que actúa.

A continuación expondremos las clases que extenderán de la clase Fragment:

- **Clase login:** Es la clase que carga la vista del login y la prepara para obtener los datos en el método onCreateView().

- **Fragments básicos de carga de interfaz:** Dentro de esta categoría incluimos todos los fragments de interfaz que son básicos. Simplemente, mediante el método onCreateView(), cargan desde un fichero xml el layout correspondiente que se quiere mostrar. Algunos cargan alguna imagen en tiempo de ejecución.

  Dentro de esta categoría se encuentran las siguientes clases:

  - activity_base
  - profile_base
  - chat_messages
  - chat_message_send
  - carta
  - hiss_menu
  - hotel_info
  - hotel_place
  - image_gallery
  - menú_top_ajustes
  - menú_top_chat_messages
  - menú_top_chats
  - menú_top_hotel
  - menú_top_profile
  - menú_top_timeline
  - profile_base
  - service_base
  - timeline_search

- **Fragments de carga de interfaz que implementan alguna funcionalidad:** Éstos son fragments que desempeñan la función de cargar un layout xml mediante el método onCreateView(), realizar una petición a la base de datos mediante la clase DownloadTask e introducir los datos en el layout cargado. Algunos

implementan listener para poder realizar acciones cuando se disparan eventos, por ejemplo al pulsar sobre un botón.

Dentro de esta categoría se encuentran las siguientes clases:

- activity
- edit_profile
- profile
- room
- service
- timeline_add_post

- Clase Gallery: Funciona de manera parecida a los ListFragments que veremos en el siguiente punto. Descarga una lista de elementos de la base de datos, carga un layout e introduce los elementos mediante un adaptador similar al que se puede contemplar en la ilustración del apartado "Clase ListFragment". Es un fragment y no un ListFragment porque se visualizan los elementos en una cuadrícula (grid).

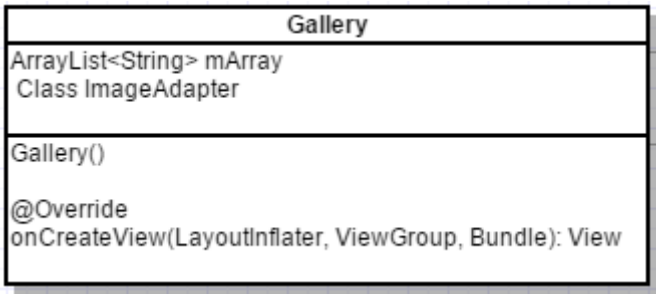


Imagen 50: Clase Gallery

8.2.4 Clase ListFragment

Esta clase funciona de igual manera que un fragment pero permite tratar listas de elementos con mayor facilidad. En general, son vistas con mayor complejidad.

En la siguiente ilustración mostramos la estructura genérica de todas las clases de este tipo implementadas:

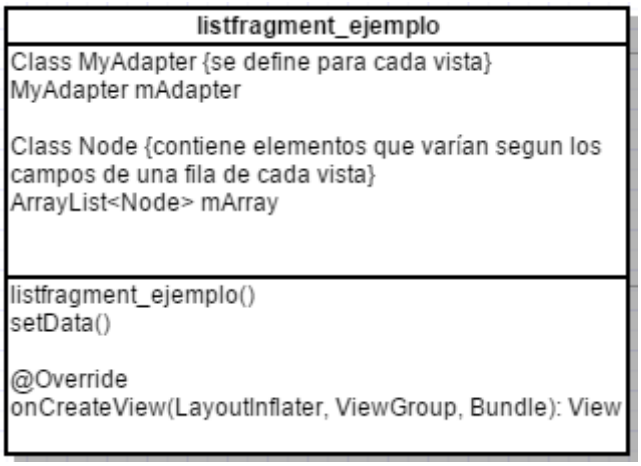


Imagen 51: Clase ejemplo ListFragment

Todas las clases que se nombran a continuación son vistas en las que en el método setData() se descarga una lista de elementos, cuyos datos se introducen en nodos.

Después, se define un adaptador de la clase BaseAdapter en el que el método getView() se encarga de recuperar los datos de los nodos y los introduce en el layout, rellenando con ello una fila de la lista que se ve en el momento actual.

Todas las clases que se nombran a continuación siguen esos pasos, aunque algunas incluyen muchas funciones de control y cambios dinámicos sobre las vistas que son complejos de implementar. Estas funciones sirven, por ejemplo, para controlar el sistema de valoraciones o el acceso a otros fragments.

- chats
- hotel
- chat_messages_list
- search_people
- services
- activities
- settings
- activity_timeline

- profile_timeline
- search_timeline
- service_timeline
- timeline

8.2.5 Clase BaseAdapter

Esta clase permite mostrar y controlar listas de elementos. Se usan clases que extienden de ella en todos los ListFragments definidos con anterioridad y todas tienen métodos similares.

El método sobre el que se trabaja principalmente es el getView(). Permite construir cada fila de la lista, insertando los datos obtenidos por el listfragment. A continuación podemos ver su estructura básica:

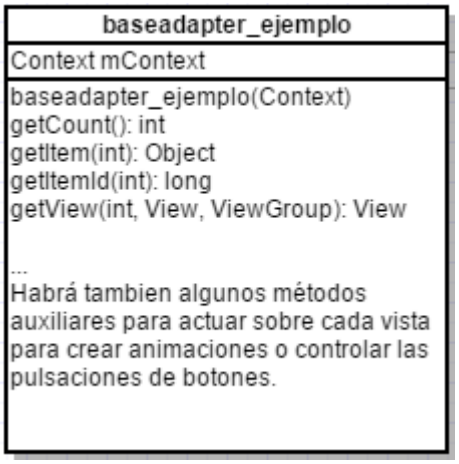


Imagen 52: Clase ejemplo BaseAdapter

Las clases que se implementarán que extiende de esta clase son:

- MyAdapter: se declara dentro de cada listfragment con algunos métodos de control de animaciones o control de vista distintos entre una vista y otra.
- ImageAdapter: se declara dentro del fragment Gallery

8.2.6 Clase AsyncTask

Permite realizar tareas en segundo plano de forma que no se bloquee el hilo principal. Extienden de esta clase las siguientes:

- **Clase DownloadTask:** Una de las clases más importantes. Permite realizar la comunicación con el servidor y obtener todos los datos que son necesarios de la base de datos. Todas las peticiones a la base de datos comienzan con una llamada a uno de los métodos declarados dentro de esta clase. La respuesta se recibe en Json mediante el método get() de esta clase. La vemos a continuación:

Imagen 53: Clase DownloadTask

8.2.7 Clase BroadcastReceiver

Permite realizar todas las gestiones relacionadas con la recepción de las notificaciones push.

- Clase PushChatBroadcastReceiver: Captura y gestiona las notificaciones push según su tipo y el estado actual de la aplicación. Recarga algunos fragments como el fragment chat_message_list, e introduce alertas visuales en el fragment hiss_menu o chats. En la siguiente ilustración vemos la composición de la clase:

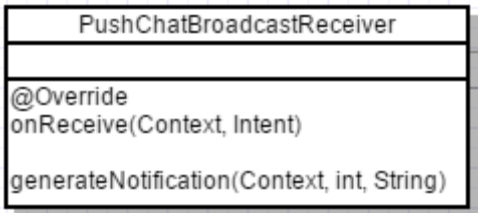


Imagen 54: Clase PushChatBroadcastReceiver

8.2.8 Otras clases

- Clase MCrypt: Esta clase se utiliza para encriptar todas las peticiones de salida al servidor (líbrería).
- Clase AndroidMultipartEntity: Extiende de MultipartEntity y lo utiliza la actividad de subida de imágenes (líbrería).
- Clase Config: Simplemente guarda unos datos de configuración que son utilizados por el activity de subida de imágenes al servidor (líbrería).

- **Clase Globals:** Se utiliza para mantener el estado de algunas variables globales. En el diagrama siguiente vemos su organización:

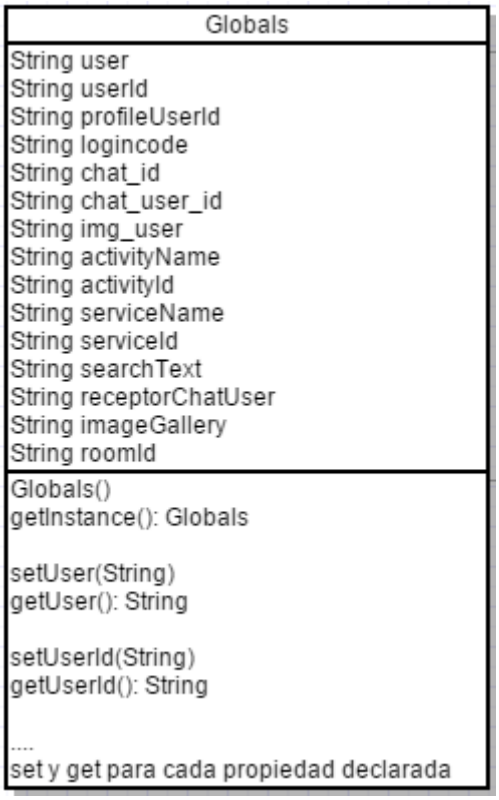


Imagen 55: Clase Globals

8.3 Diseño del Subsistema cliente Web

Este subsistema será utilizado únicamente por el administrador o administradores determinados por el hotel. Será programado con HTML, JAVASCRIPT y CSS y deberá ofrecer los elementos necesarios para realizar las operaciones descritas por los casos de uso, dentro siempre del sistema descrito con anterioridad.

Será el encargado de realizar las peticiones necesarias al subsistema servidor, presentando la información de respuesta en tablas estructuradas, con posibilidad de realizar búsquedas y realizar distintas acciones sobre los elementos.

Contará con un acceso con usuario y contraseña que dará paso a un panel de administración y con un diseño básico que permitirá añadir, editar o eliminar elementos de distinto tipo. Adicionalmente, con algunos tipos de elementos habrá operaciones disponibles de activación y desactivación.

Todo esto se podrá realizar de forma dinámica sin cambios de pantalla empleando AJAX para su implementación.

En el panel se usará la librería en JavaScript "Dynatable". Esta librería permite la generación de tablas a partir de datos en JSON, lo cual nos ayudará con las tareas de representación de información en tablas.

A continuación podemos ver gráficamente la estructura del panel:

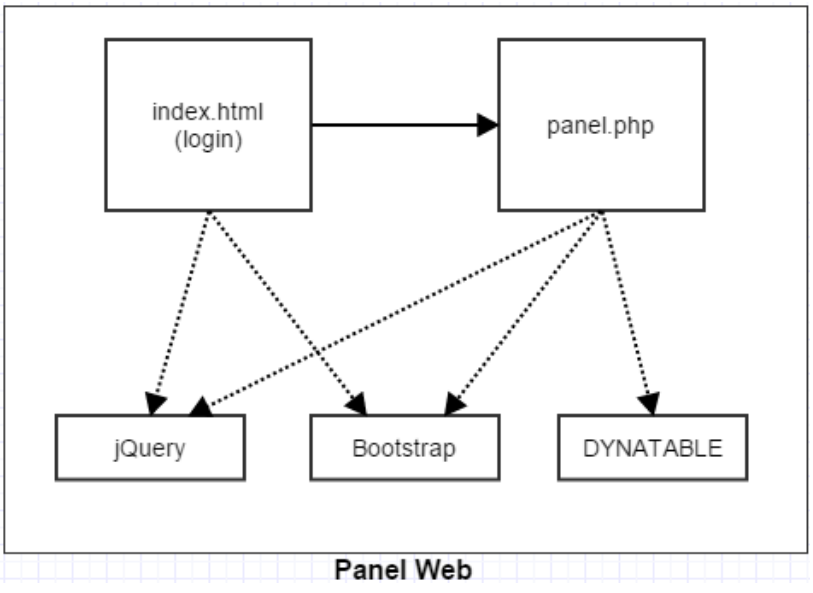


Imagen 56: Estructura del panel web

Como vemos el subsistema cliente Web se compondrá de los siguientes ficheros:

- **index.html:** Fichero que se cargará al acceder a la dirección web de acceso al panel. Contendrá la vista del login.
- **panel.php:** Este fichero contendrá la vista del panel de control. Todos los datos se cargarán dinámicamente en esta vista mediante JavaScript y las peticiones al servidor se gestionarán mediante AJAX.
- **Librería jQuery:** Esta es una librería de JavaScript ampliamente utilizada para dar interacción AJAX a las páginas web. Ese será su principal propósito.
- **Librería Bootstrap:** Usaremos la librería bootstrap (muy conocida y utilizada) para implementar el estilo CSS de las vistas.
- **Librería Dynatable:** Como se ha comentado con anterioridad se usará esta librería como apoyo para la incrustación de datos en tablas dinámicas.

8.4 Diseño del Subsistema Servidor

Este subsistema será el encargado de recibir las peticiones de los clientes, interpretarlas, realizar las operaciones oportunas en la base de datos (BD) y devolver los datos o respuestas. Deberá ser capaz de realizar estas tareas ajustándose al sistema que se ha descrito con anterioridad.

Este subsistema será programado usando lenguaje PHP y algunas librerías cuyo uso se explicará en los siguientes apartados.

Su funcionamiento básico se puede plasmar como aparece en el siguiente gráfico:

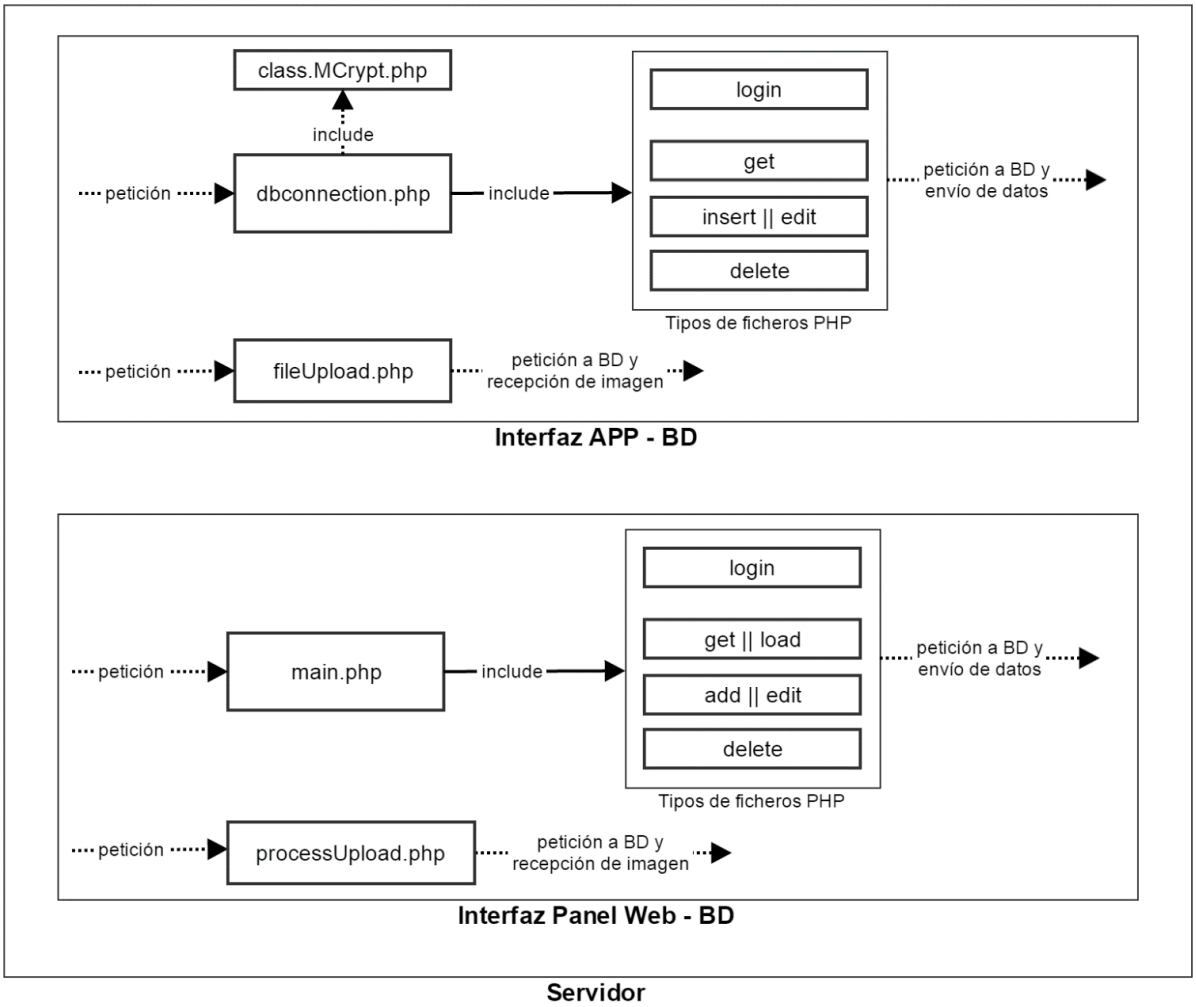


Imagen 57: Subsistema Servidor

Como vemos existirán dos interfaces con la BD, una para el cliente de la APP Android y otra para el cliente Web. A continuación detallamos como serán sus estructuras.

8.4.1 Interfaz cliente APP – BD

Esta interfaz llevará a cabo su labor cuando se reciba una petición por parte del cliente de la APP Android. Seguirá los siguientes pasos básicos:

- Dbconnection.php recibe la petición.
- Se incluye la librería MCrypt.php que se usa para desencriptar los datos recibidos, que se habrán encriptado al enviarlos desde el cliente.
- Se identifica el tipo de la petición que puede ser de cuatro tipos básicos: login, get, insert/edit o delete.

- En base al tipo de la petición y sobre que tabla se tiene que realizar, se hace el include de un fichero PHP que se encargará de realizar las operaciones apropiadas según la petición.
- El código de ese fichero incluido se encargará de realizar las comprobaciones y acciones oportunas relacionadas con la BD y también de devolver los datos en JSON al cliente APP.

Un ejemplo de los archivos que procesarán las peticiones atendiendo a su tipo es el siguiente:

- InsertPost.php
- GetPosts.php
- DeletePosts.php

Como se ve, a modo de ejemplo hemos puesto los tres ficheros correspondientes a las publicaciones, esta sería la organización seguida para el resto de casos, como chats o services.

Por último, hay que destacar que, para el caso concreto de la subida de imágenes al servidor, el cliente se comunicará con el servidor a través de la librería fileUpload.php.

8.4.2 Interfaz cliente Web – BD

Esta interfaz llevará a cabo su labor cuando se reciba una petición por parte del cliente del panel web. Todas sus peticiones están relacionadas con la administración, y seguirá los siguientes pasos básicos:

- main.php recibe la petición.
- Se identifica el tipo de la petición que puede ser de cuatro tipos básicos: login, get/load, add/edit o delete.
- En base al tipo de la petición y sobre que tabla se tiene que realizar, se hace el include de un fichero PHP que se encargará de realizar las operaciones apropiadas según la petición.

- El código de ese fichero incluido se encarga de realizar las comprobaciones y acciones oportunas relacionadas con la BD y también de devolver los datos en JSON al cliente APP.

Un ejemplo de los archivos que procesarán las peticiones atendiendo a su tipo es el siguiente:

- addActivity.php
- getActivities.php
- deleteActivity.php

Como se ve, a modo de ejemplo hemos puesto los tres ficheros correspondientes a las actividades, esta sería la organización seguida para el resto de casos, como servicios o cartas de restauración.

De nuevo hay que destacar que para el caso concreto de la subida de imágenes al servidor, el cliente se comunicará con el servidor a través de la librería processUpload.php.

8.4.3 Patrón de diseño utilizado

Para la implementación de este subsistema se utilizara sobre cada interfaz el patrón de diseño Fachada (Facade). Este patrón de diseño consiste en tener un punto de entrada que da acceso al resto de componentes del subsistema. Esto hace que sea más fácil de usar.

En la siguiente ilustración podemos ver un ejemplo gráfico de este patrón en nuestro subsistema:

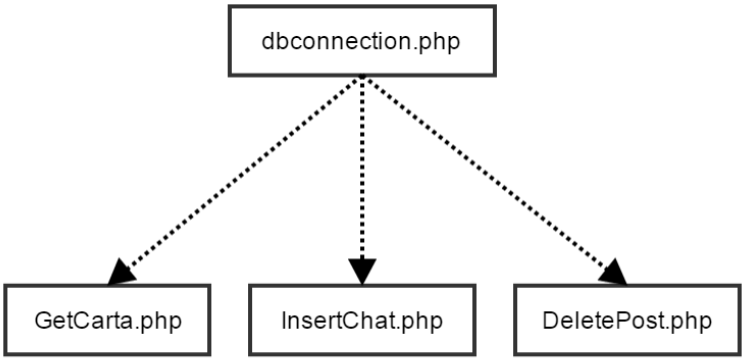


Imagen 58: Patrón de diseño utilizado

Como se ha explicado con anterioridad, en cada interfaz del sistema se tendrá un punto de acceso que se encargará de interpretar el tipo de la operación a realizar. Tras identificar el tipo de operación incluirá un fichero php específico, que realizará las operaciones oportunas, y enviará los datos oportunos al cliente.

8.5 Diseño de la base de datos

Se decide usar Mysql como sistema de base de datos (BD), debido a su amplia documentación y su disponibilidad gratuita. El diseño desarrollado deberá ajustarse a las especificaciones que necesitan los distintos subsistemas analizados con anterioridad.

A continuación mostramos dos imágenes con el diseño realizado. Las flechas indican las relaciones de precedencia que existen entre unas tablas y otras. Por ejemplo, para que existan una tabla de mensajes tiene que existir una tabla de usuarios.

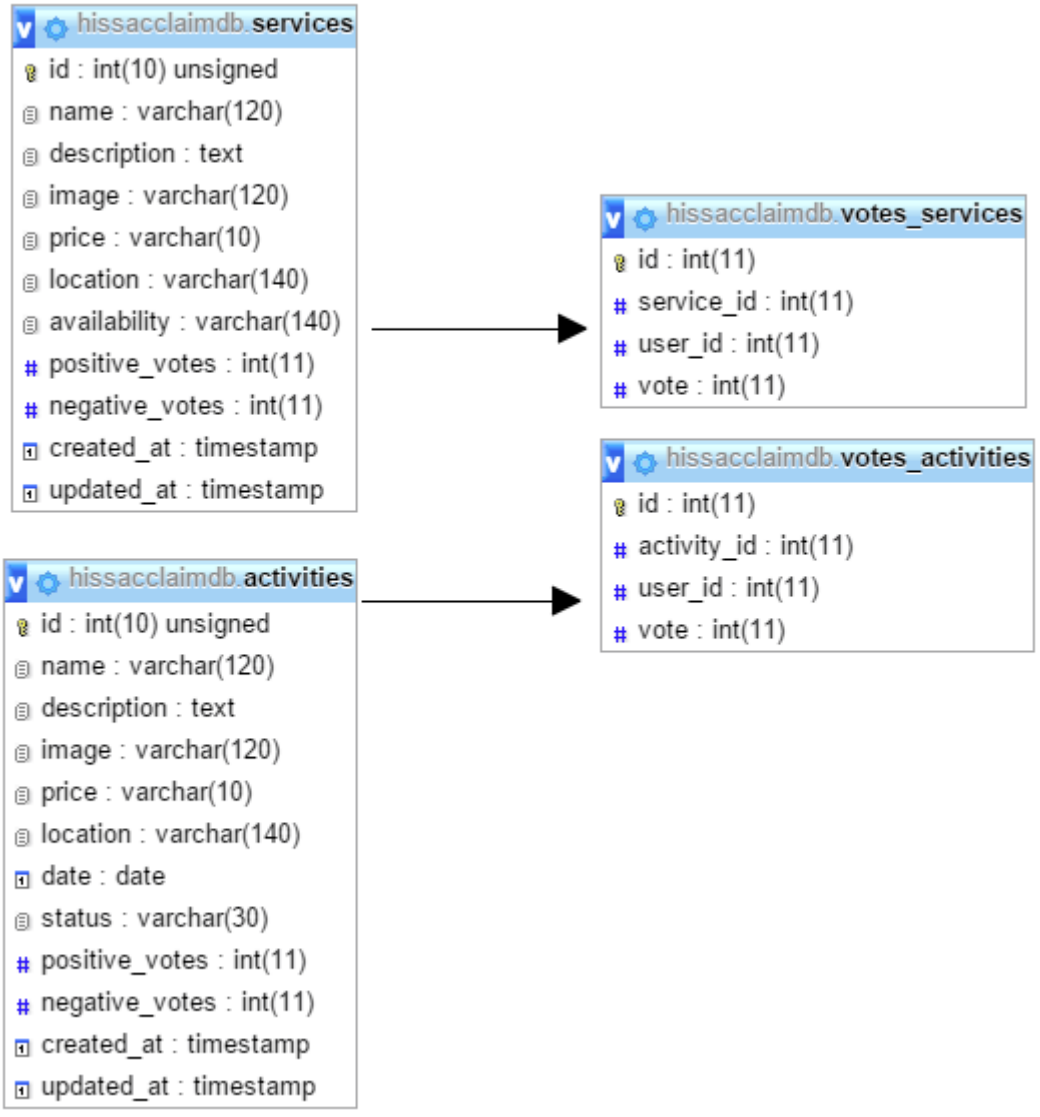


Imagen 59: Diseño BD 1

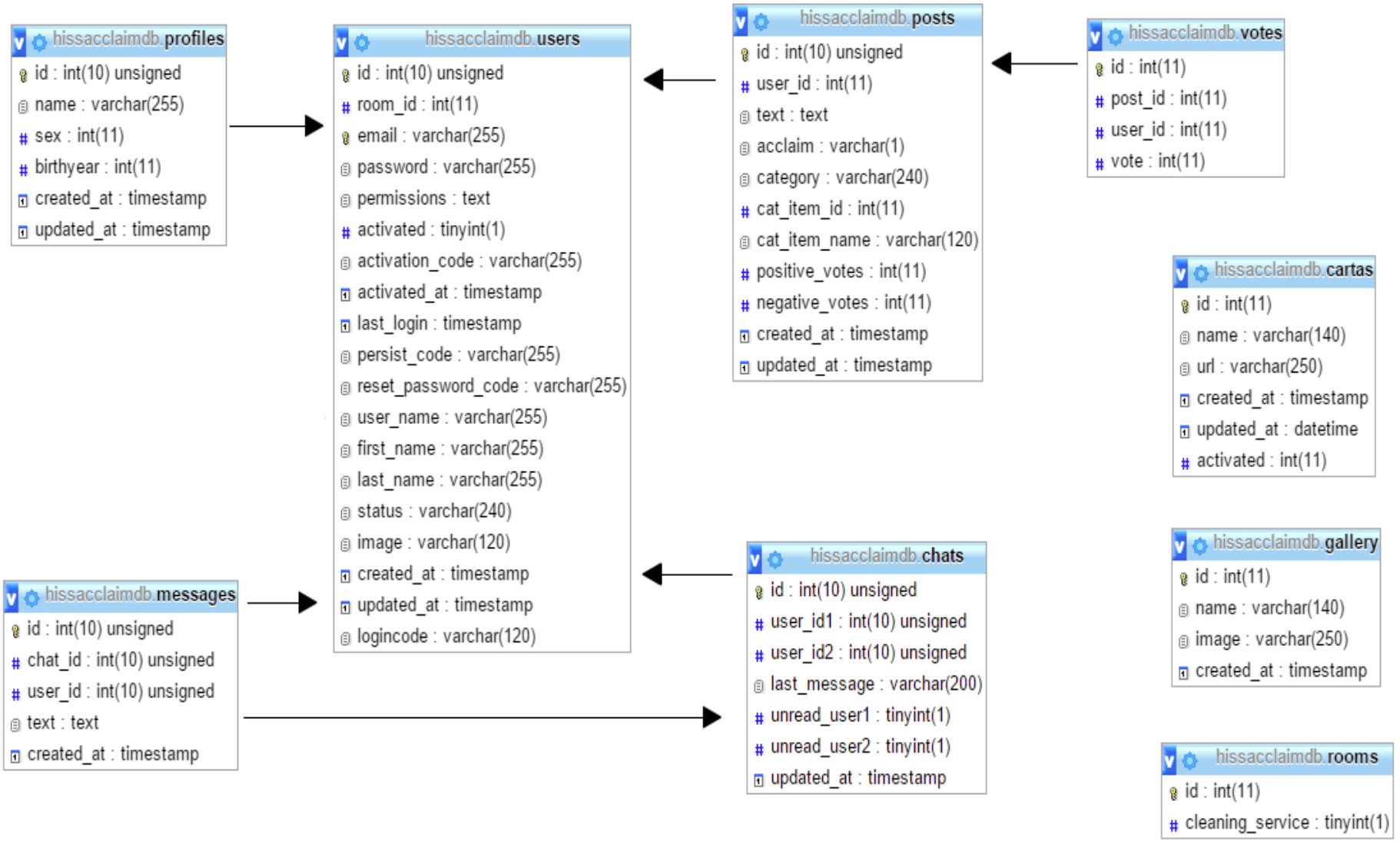


Imagen 60: Diseño BD 2

9 Detalles sobre la implementación

El resultado de esta fase es el código que se encuentra en el CD adjunto a la memoria.

No obstante, a continuación expondremos algunos de los ficheros de código que se han desarrollado y que consideramos de especial interés dentro del desarrollo realizado.

9.1 Interfaces gráficas Android

En cuanto a las interfaces, queremos destacar la complejidad que existe en Android a la hora de implementarlas de forma nativa. Al existir numerosos dispositivos con diferentes tamaños, resoluciones y densidad de píxeles, hay que tener en cuenta todos estos datos variables para poder implementar un diseño válido.

Por esto, en Android se forman varios rangos en función de la densidad de píxeles y la resolución, tal y como se muestra a continuación:

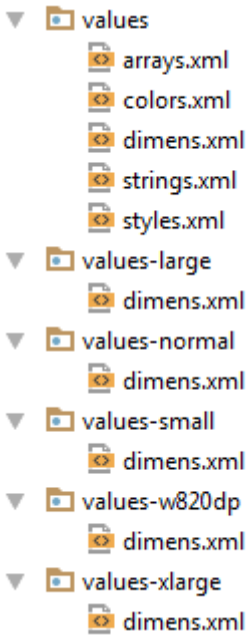


Imagen 61: Ejemplo values Hiss&Acclaim

Valores estándar, grandes, normales, pequeños, 820 píxeles de ancho y extra grandes. Cada uno, tiene un fichero "dimens.xml" donde estableceremos las dimensiones de los elementos que queramos indicar.

Esto requiere de un trabajo de muchas horas a base de prueba y error ya que, dentro de cada rango, también es complejo encontrar valores que encajen con el valor máximo y mínimo de dicho rango.

A continuación podemos ver un ejemplo de uno de los ficheros dimens.xml:

```
/*
Cada fichero de valores contiene la etiqueta "resources" (recursos)
y dentro se definen parámetros necesarios para las diferentes vistas
*/

<resources>
    <!-- Default screen margins, per the Android Design guidelines. --
>

    //Valores para la vista de login
    <!-- Valores del login -->
    <dimen type="integer" name="layout_margin_logo">20dp</dimen>
    <dimen type="integer" name="login_box_width">250dp</dimen>

    //Valores para el menú general
    <!-- Valores del menu inferior -->
    <dimen name="menu_margin_top">2dp</dimen>

    //Valores para el timeline
    <!-- Valores del timeline -->
    <dimen name="hissline_text_size">11sp</dimen>
    <dimen name="hissline_date_size">9sp</dimen>
    <item type="integer" name="pic_weight">0.24</item>
    <item type="integer" name="text_weight">0.56</item>
    <item type="integer" name="hands_weight">0.20</item>

    //Valores para la vista de chats
    <!-- Valores de chats -->
    <dimen name="chat_pic_size">80dp</dimen>
    <dimen name="chat_name_size">16sp</dimen>
    <dimen name="chat_text_size">12sp</dimen>
    <dimen name="chat_time_size">10sp</dimen>

    //Valores generales
    <!-- Valores generales -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>

    //Valores para la vista de editar perfil
    <!-- Valores editar perfil -->
    <dimen name="edit_profile_title">18sp</dimen>
    <dimen name="edit_profile_subtitle">14sp</dimen>
    <dimen name="edit_profile_pic">100dp</dimen>
    <dimen name="edit_profile_margin_block">10dp</dimen>

</resources>
```

9.2 Clase ListFragment con su adaptador

Esta clase ha sido muy utilizada a lo largo del proyecto para mostrar vistas con listas. Cada clase desarrollada que extienda de la clase ListFragment, contiene una clase nodo (que almacenará los datos de cada elemento de la lista), un array que almacenará todo el listado de nodos, un método setData() que obtiene los datos de la base de datos y los añade al nodo correspondiente y un adaptador para gestionar a través de su método getView() todos los datos del array en la vista.

Mostramos a continuación, a modo de ejemplo, una de las clases desarrolladas de este tipo, la clase timeline. package com.hissacclaim.pfc.pfchissacclaim;

```
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.app.ListFragment;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.nostra13.universalimageloader.core.ImageLoader;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;


/**
 * A simple {@link Fragment} subclass.
 */

public class timeline extends ListFragment {

    public timeline() {
        // Constructor público vacío necesario
    }
```

```
    //Propiedad para gestionar la visualización y controles de la
vista
    private MyAdapter mAdapter = null;

    //Nodo para un timeline, con todos los datos necesarios
    public class HissNode {
        public String Name;
        public String Category;
        public String Acclaim;
        public String Description;
        public String Img;
        public String myVote;
        public String Positives;
        public String Negatives;
        public String MyDate;
        public String PostId;
        public String UserId;
        public Boolean MasPosts= false;
        public Boolean ultimo= false;
    }

    //La propiedad índice sirve para controlar si hay que cargar más
posts o no
    private int indice=1;

    //Array que almacena los nodos para formar la lista
    private static ArrayList<HissNode> mArray = new
ArrayList<HissNode>();

    //Método que obtiene los datos de la base de datos y los asigna a
nodos y los encadena en mArray
    private void setData () throws JSONException {
        //Reseteamos la lista
        mArray.clear();

        JSONArray request = null;

        //Pedimos los datos al servidor
        try{
            DownloadTask dlTask = new DownloadTask();
            dlTask.getPosts(Integer.toString(indice*10));
            dlTask.execute();
            request= dlTask.get();
        }catch(Exception e){
            Toast.makeText(getActivity().getApplicationContext(),
                "Error al recuperar BD data del timeline",
Toast.LENGTH_LONG).show();
        }

        JSONObject json_data;
        if(request!=null){

            //Calculamos si hay posts antiguos o no
            if( 9 < ((indice*10) - request.length())) {
                Toast.makeText(getActivity().getApplicationContext(),
                    "No hay posts más antiguos",
Toast.LENGTH_LONG).show();
            }

            //Por cada elemento, le asignamos valores a sus
propiedades,
```

```
            //lo metemos en un nodo y encadenamos el nodo en el array
            for (int i = 0; i < request.length(); i++) {
                json_data = request.getJSONObject(i);

                HissNode mynode = new HissNode();

                mynode.Name = json_data.getString("first_name") + " "
+ json_data.getString("last_name");
                mynode.Category =
json_data.getString("cat_item_name");
                if (json_data.getString("acclaim").equals("1"))
mynode.Acclaim = "Acclaim";
                else mynode.Acclaim = "Hiss";
                mynode.Description = json_data.getString("text");
                mynode.Positives =
json_data.getString("positive_votes");
                mynode.Negatives =
json_data.getString("negative_votes");
                mynode.MyDate = json_data.getString("created_at");
                mynode.Img = json_data.getString("image");
                mynode.PostId = json_data.getString("id");
                mynode.UserId = json_data.getString("user_id");
                mynode.MasPosts = false;

                if (json_data.getString("vote") == "null")
mynode.myVote = "0";
                else mynode.myVote = json_data.getString("vote");

                mArray.add(mynode);
            }

            //preparamos un último nodo para mostrar "Cargar más si
fuera necesario"
            HissNode mynode = new HissNode();
            mynode.MasPosts = true;
            if( 0 > ( (request.length()*1) - (indice*10))) {
                mynode.ultimo=true;
            }
            mArray.add(mynode);


        }else{
            Toast.makeText(getActivity().getApplicationContext(), "No
Data", Toast.LENGTH_LONG).show();
        }

    }


    //Adaptador para controlar la gestión y visualización de la vista
    public class MyAdapter extends BaseAdapter {
        //Campos y propiedades básicos necesarios cuando se extiende
de baseadapter
        private Context myContext;
        public MyAdapter(Context c) {
            myContext=c;
        }
        public int getCount() {
            return mArray.size();
        }
        public Object getItem(int position) {
```

```
            return mArray.get(position);
        }
        public long getItemId(int position) {
            return 0;
        }

        //Método que obtiene la vista
        public View getView(final int position, View convertView,
ViewGroup parent) {
            View view = null;
            //Si ya tenemos la vista, la actualizamos, si no, creamos
una nueva
            if (convertView == null) {
                LayoutInflater inflater = (LayoutInflater)
myContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                view =
inflater.inflate(R.layout.timeline_layout,null);
            }
            else {
                view = convertView;
            }
            //Obtenemos el layout de la vista
            LinearLayout myLayout = (LinearLayout)
view.findViewById(R.id.timeline_layout);

            //En función de la versión de Android que utilicemos, le
asignamos un fondo al layout
            int sdk = android.os.Build.VERSION.SDK_INT;
            if(sdk < android.os.Build.VERSION_CODES.JELLY_BEAN) {

myLayout.setBackgroundDrawable(getResources().getDrawable(R.drawable.t
imeline_degrade));
            } else {

myLayout.setBackground(getResources().getDrawable(R.drawable.timeline_
degrade));
            }


            //Si es el elemento "Cargar más", ocultamos todos los
elementos de la vista
            if(mArray.get(position).MasPosts){

                ImageView img = (ImageView)
view.findViewById(R.id.NodeImage);
                img.setVisibility(View.GONE);
                TextView name = (TextView)
view.findViewById(R.id.NodeName);
                name.setVisibility(View.GONE);
                TextView category = (TextView)
view.findViewById(R.id.TextCategory);
                category.setVisibility(View.GONE);
                TextView acclaim = (TextView)
view.findViewById(R.id.TextAcclaim);
                acclaim.setVisibility(View.GONE);
                TextView description = (TextView)
view.findViewById(R.id.NodeDescription);
                description.setVisibility(View.GONE);
                TextView positiv = (TextView)
view.findViewById(R.id.NodePositives);
                positiv.setVisibility(View.GONE);
```

```
                TextView negativ = (TextView)
view.findViewById(R.id.NodeNegatives);
                negativ.setVisibility(View.GONE);
                TextView myVote = (TextView)
view.findViewById(R.id.myVote);
                myVote.setVisibility(View.GONE);
                TextView postID = (TextView)
view.findViewById(R.id.NodePostID);
                postID.setVisibility(View.GONE);
                TextView userID = (TextView)
view.findViewById(R.id.NodeUserID);
                userID.setVisibility(View.GONE);
                TextView mydate = (TextView)
view.findViewById(R.id.HissDate);
                mydate.setVisibility(View.GONE);
                ImageButton btnminus = (ImageButton)
view.findViewById(R.id.buttonMinus);
                btnminus.setVisibility(View.GONE);
                ImageButton btnplus = (ImageButton)
view.findViewById(R.id.buttonPlus);
                btnplus.setVisibility(View.GONE);

                //Si no es el último elemento, cargamos todos los
datos del layout y les asignamos los valores de su nodo
correspondiente
                if(!mArray.get(position).ultimo) {
                    TextView ultimo = (TextView)
view.findViewById(R.id.TextUltimo);
                    ultimo.setVisibility(View.GONE);
                    Button btn = (Button)
view.findViewById(R.id.buttonMasPosts);
                    btn.setVisibility(View.VISIBLE);
                    btn.setOnClickListener(new View.OnClickListener()
{
                        @Override
                        public void onClick(View v) {
                            indice++;

                            Fragment currentFragment =
getFragmentManager().findFragmentByTag("TIMELINE");
                            final FragmentTransaction fragTransaction
= getFragmentManager().beginTransaction();
                            fragTransaction.detach(currentFragment);
                            fragTransaction.attach(currentFragment);
                            fragTransaction.commit();

                            return;
                        }

                    });
                }else{
                    Button btn = (Button)
view.findViewById(R.id.buttonMasPosts);
                    btn.setVisibility(View.GONE);

                    TextView ultimo = (TextView)
view.findViewById(R.id.TextUltimo);
                    ultimo.setVisibility(View.VISIBLE);
                }

                return view;
```

```
            }else{
                //Si no es el último ponemos todos los elementos
visibles
                ImageView img = (ImageView)
view.findViewById(R.id.NodeImage);
                img.setVisibility(View.VISIBLE);
                TextView name = (TextView)
view.findViewById(R.id.NodeName);
                name.setVisibility(View.VISIBLE);
                TextView category = (TextView)
view.findViewById(R.id.TextCategory);
                category.setVisibility(View.VISIBLE);
                TextView acclaim = (TextView)
view.findViewById(R.id.TextAcclaim);
                acclaim.setVisibility(View.VISIBLE);
                TextView description = (TextView)
view.findViewById(R.id.NodeDescription);
                description.setVisibility(View.VISIBLE);
                TextView positiv = (TextView)
view.findViewById(R.id.NodePositives);
                positiv.setVisibility(View.VISIBLE);
                TextView negativ = (TextView)
view.findViewById(R.id.NodeNegatives);
                negativ.setVisibility(View.VISIBLE);
                TextView mydate = (TextView)
view.findViewById(R.id.HissDate);
                mydate.setVisibility(View.VISIBLE);
                ImageButton btnminus = (ImageButton)
view.findViewById(R.id.buttonMinus);
                btnminus.setVisibility(View.VISIBLE);
                ImageButton btnplus = (ImageButton)
view.findViewById(R.id.buttonPlus);
                btnplus.setVisibility(View.VISIBLE);

                Button btn = (Button)
view.findViewById(R.id.buttonMasPosts);
                btn.setVisibility(View.GONE);
                TextView ultimo = (TextView)
view.findViewById(R.id.TextUltimo);
                ultimo.setVisibility(View.GONE);
            }

            //Cargamos los datos en la vista
            ImageView img = (ImageView)
view.findViewById(R.id.NodeImage);

ImageLoader.getInstance().displayImage(mArray.get(position).Img, img);
            if(sdk < android.os.Build.VERSION_CODES.JELLY_BEAN) {

img.setBackgroundDrawable(getResources().getDrawable(R.drawable.timeli
ne_image));
            } else {

img.setBackground(getResources().getDrawable(R.drawable.timeline_image
));
            }
            TextView name = (TextView)
view.findViewById(R.id.NodeName);
            name.setText(mArray.get(position).Name);
```

```
            TextView category = (TextView)
view.findViewById(R.id.TextCategory);
            category.setText(mArray.get(position).Category);

            TextView acclaim = (TextView)
view.findViewById(R.id.TextAcclaim);
            acclaim.setText(mArray.get(position).Acclaim);

            TextView description = (TextView)
view.findViewById(R.id.NodeDescription);
            description.setText(mArray.get(position).Description);

            TextView positiv = (TextView)
view.findViewById(R.id.NodePositives);
            positiv.setText(mArray.get(position).Positives);

            TextView negativ = (TextView)
view.findViewById(R.id.NodeNegatives);
            negativ.setText(mArray.get(position).Negatives);

            TextView myVote = (TextView)
view.findViewById(R.id.myVote);
            myVote.setText(mArray.get(position).myVote);

            TextView postID = (TextView)
view.findViewById(R.id.NodePostID);
            postID.setText(mArray.get(position).PostId);

            TextView userID = (TextView)
view.findViewById(R.id.NodeUserID);
            userID.setText(mArray.get(position).UserId);

            TextView mydate = (TextView)
view.findViewById(R.id.HissDate);
            mydate.setText(mArray.get(position).MyDate);

            //Ponemos la mano del color necesario en función del voto
del usuario
            ImageButton myButton = (ImageButton)
view.findViewById(R.id.buttonPlus);
            ImageButton myButton2 = (ImageButton)
view.findViewById(R.id.buttonMinus);
            if (acclaim.getText()=="Hiss")
acclaim.setTextColor(getResources().getColor(R.color.hiss));
            if (acclaim.getText()=="Acclaim")
acclaim.setTextColor(getResources().getColor(R.color.acclaim));
            if (mArray.get(position).myVote.equals("1")) {

                if(sdk < android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton.setBackgroundDrawable(getResources().getDrawable(R.drawable.i
c_gusta_azul));

myButton2.setBackgroundDrawable(getResources().getDrawable(R.drawable.
ic_nogusta_gris));
                } else {

myButton.setBackground(getResources().getDrawable(R.drawable.ic_gusta_
azul));
```

```
myButton2.setBackground(getResources().getDrawable(R.drawable.ic_nogus
ta_gris));
                }

            }
            else {
                if (mArray.get(position).myVote.equals("-1")) {
                    if(sdk <
android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton.setBackgroundDrawable(getResources().getDrawable(R.drawable.i
c_gusta_gris));

myButton2.setBackgroundDrawable(getResources().getDrawable(R.drawable.
ic_nogusta_rojo));
                    }else{

myButton.setBackground(getResources().getDrawable(R.drawable.ic_gusta_
gris));

myButton2.setBackground(getResources().getDrawable(R.drawable.ic_nogus
ta_rojo));
                    }

                }
                else {
                    if(sdk <
android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton.setBackgroundDrawable(getResources().getDrawable(R.drawable.i
c_gusta_gris));

myButton2.setBackgroundDrawable(getResources().getDrawable(R.drawable.
ic_nogusta_gris));
                    }else{

myButton.setBackground(getResources().getDrawable(R.drawable.ic_gusta_
gris));

myButton2.setBackground(getResources().getDrawable(R.drawable.ic_nogus
ta_gris));
                    }

                }
            }
            myButton.setVisibility(View.VISIBLE);
            myButton2.setVisibility(View.VISIBLE);

            ImageButton btn = (ImageButton)
view.findViewById(R.id.buttonPlus);
            btn.setOnTouchListener(new View.OnTouchListener() {
                /*@Override
                public void onClick(View v) {
                    likePost(v, position);
                }*/
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                    int sdk = android.os.Build.VERSION.SDK_INT;
                    ImageButton btn = (ImageButton)
v.findViewById(R.id.buttonPlus);
```

```
                    final Animation anim =
AnimationUtils.loadAnimation(myContext, R.anim.scale_icon);

                    ConnectivityManager connMgr =
(ConnectivityManager)
getActivity().getSystemService(getActivity().getApplicationContext().C
ONNECTIVITY_SERVICE);
                    NetworkInfo networkInfo =
connMgr.getActiveNetworkInfo();
                    if (networkInfo == null ||
!networkInfo.isConnected()) {

Toast.makeText(getActivity().getApplicationContext(), "Not Connected",
Toast.LENGTH_LONG).show();
                        return true;
                    }

                    if (event.getAction() == MotionEvent.ACTION_DOWN)
{
                        v.startAnimation(anim);
                    }
                    else if ((event.getAction() ==
MotionEvent.ACTION_UP)&&(0<event.getX())&&(event.getX()<btn.getWidth()
)) {
                        likePost(v, position);
                    }
                    return true;
                }

            });

            ImageButton btn2 = (ImageButton)
view.findViewById(R.id.buttonMinus);
                btn2.setOnTouchListener(new View.OnTouchListener() {
                 /*@Override
                 public void onClick(View v) {
                     likePost(v, position);
                 }*/
                 @Override
                 public boolean onTouch(View v, MotionEvent event) {
                     int sdk = android.os.Build.VERSION.SDK_INT;
                     ImageButton btn2 = (ImageButton)
v.findViewById(R.id.buttonMinus);
                    final Animation anim =
AnimationUtils.loadAnimation(myContext, R.anim.scale_icon);

                    ConnectivityManager connMgr =
(ConnectivityManager)
getActivity().getSystemService(getActivity().getApplicationContext().C
ONNECTIVITY_SERVICE);
                    NetworkInfo networkInfo =
connMgr.getActiveNetworkInfo();
                    if (networkInfo == null ||
!networkInfo.isConnected()) {

Toast.makeText(getActivity().getApplicationContext(), "Not Connected",
Toast.LENGTH_LONG).show();
                        return true;
                    }
```

```
                    if (event.getAction() == MotionEvent.ACTION_DOWN)
{
                        v.startAnimation(anim);
                    }
                    else if ((event.getAction() ==
MotionEvent.ACTION_UP)&&(0<event.getX())&&(event.getX()<btn2.getWidth(
))) {
                        dislikePost(v, position);
                    }

                    return true;
                }

            });

            return view;
        }


        //Método para la llamada de valoración positiva
        public int likePost(View v, int position) {
            int sdk = android.os.Build.VERSION.SDK_INT;

            String value;
            String current = mArray.get(position).myVote;
            String postID = mArray.get(position).PostId;
            ImageButton myButton = (ImageButton)
v.findViewById(R.id.buttonPlus);
            ImageButton myButton2 = (ImageButton)
v.findViewById(R.id.buttonMinus);

            //current=peticion a la base de datos del valor de la
valoración

                if (current.equals("1")) {
                    value="0";
                    if(sdk <
android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton.setBackgroundDrawable(getResources().getDrawable(R.drawable.i
c_gusta_gris));
                    }else{

myButton.setBackground(getResources().getDrawable(R.drawable.ic_gusta_
gris));
                    }
                }else{
                    value="1";

                    if(sdk <
android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton.setBackgroundDrawable(getResources().getDrawable(R.drawable.i
c_gusta_azul));
                    }else{

myButton.setBackground(getResources().getDrawable(R.drawable.ic_gusta_
azul));
                    }
                }
```

```
            try{
                JSONArray prueba = null;

                DownloadTask dlTask = new DownloadTask();
                dlTask.insertVote(value, postID);
                dlTask.execute();
                prueba = dlTask.get();

                JSONObject json_data;
                json_data = prueba.getJSONObject(0);

                String Result = json_data.getString("response");

                if (!Result.equals("ACCEPT"))
                    Toast.makeText(v.getContext(), "Error al guardar
la valoración-- ", Toast.LENGTH_LONG).show();

            }catch(Exception e){
                Toast.makeText(v.getContext(), "Error al enviar la
valoración", Toast.LENGTH_LONG).show();
            }


            Fragment currentFragment =
getFragmentManager().findFragmentByTag("TIMELINE");
            final FragmentTransaction fragTransaction =
getFragmentManager().beginTransaction();
            fragTransaction.detach(currentFragment);
            fragTransaction.attach(currentFragment);
            fragTransaction.commit();

            return 0;
        }

        //Método para la llamada de valoración negativa
        public int dislikePost(View v, int position) {
            int sdk = android.os.Build.VERSION.SDK_INT;

            String value;
            String current = mArray.get(position).myVote;
            String postID = mArray.get(position).PostId;
            ImageButton myButton = (ImageButton)
v.findViewById(R.id.buttonPlus);
            ImageButton myButton2 = (ImageButton)
v.findViewById(R.id.buttonMinus);

            //current=peticion a la base de datos del valor de la
valoración


            if (current.equals("-1")) {
                value="0";
                if(sdk < android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton2.setBackgroundDrawable(getResources().getDrawable(R.drawable.
ic_nogusta_gris));
                }else{

myButton2.setBackground(getResources().getDrawable(R.drawable.ic_nogus
ta_gris));
```

```
                }
            }else{
                value="-1";
                if(sdk < android.os.Build.VERSION_CODES.JELLY_BEAN) {

myButton2.setBackgroundDrawable(getResources().getDrawable(R.drawable.
ic_nogusta_rojo));
                }else{

myButton2.setBackground(getResources().getDrawable(R.drawable.ic_nogus
ta_rojo));
                }
            }

            String Result= null;
            try{
                JSONArray prueba = null;

                DownloadTask dlTask = new DownloadTask();
                dlTask.insertVote(value, postID);
                dlTask.execute();
                prueba = dlTask.get();

                JSONObject json_data;
                json_data = prueba.getJSONObject(0);
                Result = json_data.getString("response");

                if (!Result.equals("ACCEPT"))
                    Toast.makeText(v.getContext(), "Error al guardar
la valoración-- ", Toast.LENGTH_LONG).show();

            }catch(Exception e){
                Toast.makeText(v.getContext(), "Error fatal al enviar
la valoración", Toast.LENGTH_LONG).show();
            }

            Fragment currentFragment =
getFragmentManager().findFragmentByTag("TIMELINE");
            final FragmentTransaction fragTransaction =
getFragmentManager().beginTransaction();
            fragTransaction.detach(currentFragment);
            fragTransaction.attach(currentFragment);
            fragTransaction.commit();

            return 0;
        }


    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        //Inflate the layout for this fragment
        try {
            setData();
        }catch(JSONException e){
            e.printStackTrace();
        }
        mAdapter = new MyAdapter(this.getActivity());
```

```
        setListAdapter(mAdapter);

        return inflater.inflate(R.layout.timeline_layout, container,
false);
    }


}
```

9.3 Clase DownloadTask

La clase DownloadTask contiene unas propiedades que se utilizarán para realizar peticiones al servidor y unos métodos que preparan la petición para que luego ésta sea lanzada. Además, se encarga de enviar los datos recibidos cuando se le solicite.

```
package com.hissacclaim.pfc.pfchissacclaim;


import android.app.Activity;
import android.content.Intent;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

/**
 * Created by Torrevieja on 24/10/2014.
 */
class DownloadTask extends AsyncTask<String, Void, JSONArray> {

//Conjunto de propiedades que se utilizan para lanzar peticiones a la
base de datos
    private String
urlWebService="http://hissacclaim.morwebs.com/webservice/";
    private String type=null;
    private String query=null;
    private String columns=null;
    private String colValues=null;
    private String table=null;
    private String condition=null;
    private String user=null;
    private String password=null;

    private String id=null;
    private String limit=null;
    //Se inicializan todas a nulo y luego serán modificadas cuando
sean necesarias

    //El método reset reinicializa los valores a nulo para cuando haya
varias peticiones, que no queden residuos de la anterior
    public int reset(){

this.urlWebService="http://hissacclaim.morwebs.com/webservice/";
        this.type=null;
```

```
        this.query=null;
        this.columns=null;
        this.colValues=null;
        this.table=null;
        this.condition=null;
        this.user=null;
        this.password=null;

        this.id=null;
        this.limit=null;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
login
    public int login(String user, String password){
        this.reset();
        this.type= "login";
        this.user= user;
        this.password= password;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
logincode
    public int loginCode(String user, String logincode){
        this.reset();
        this.type= "loginCode";
        this.user= user;
        this.password= logincode;
        return 1;
    }


//////////////////GETS/////////////////
    //Método que prepara las propiedades para lanzar la petición de
obtener usuario
    public int getActualUser(){
        this.reset();
        this.type= "getActualUser";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener el perfil de un usuario
    public int getUserProfileData(String user_id){
        this.reset();
        this.type= "getUserProfileData";
        this.table= "users";
        this.id= user_id;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
login
    public int getActivityData(String activity_id){
        this.reset();
        this.type= "getActivityData";
        this.table= "activities";
        this.id= activity_id;
        return 1;
```

```
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener datos de servicio
    public int getServiceData(String service_id){
        this.reset();
        this.type= "getServiceData";
        this.table= "services";
        this.id= service_id;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener los datos de editar perfil
    public int editMyProfile(String valores, String columnas){
        this.reset();
        this.type= "editMyProfile";
        this.columns= columnas;
        this.colValues= valores;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener posts del timeline
    public int getPosts(String indice){
        this.reset();
        this.type= "getPosts";
        this.limit= indice;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener los datos de un timeline exclusivo de perfiles
    public int getProfilePosts(String profile_user_id, String indice){
        this.reset();
        this.type= "getProfilePosts";
        this.id= profile_user_id;
        this.limit=indice;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener los datos de un timeline exclusivo de actividades
    public int getActivityPosts(String activity_id, String indice){
        this.reset();
        this.type= "getActivityPosts";
        this.id= activity_id;
        this.limit=indice;
        return 1;
    }

//Método que prepara las propiedades para lanzar la petición de
obtener los datos de un timeline exclusivo de servicios
    public int getServicePosts(String service_id, String indice){
        this.reset();
        this.type= "getServicePosts";
        this.id= service_id;
        this.limit=indice;
        return 1;
    }
```

```
    //Método que prepara las propiedades para lanzar la petición de
obtener los datos de un timeline después de una búsqueda de posts
    public int getSearchPosts(String text, String indice){
        this.reset();
        this.type= "getSearchPosts";
        this.id= text;
        this.limit=indice;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener los datos de un timeline después de una búsqueda de usuarios
    public int getSearchPeople(String text, String indice){
        this.reset();
        this.type= "getSearchPeople";
        this.id= text;
        this.limit=indice;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener el listado de chats de un usuario
    public int getChats(){
        this.reset();
        this.type= "getChats";
        this.table= "chats";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener los mensajes de un chat concreto
    public int getMessages(String chat_id, String yo){
        this.reset();
        this.type= "getMessages";
        this.table= "messages";
        this.id= chat_id;
        this.limit= yo;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener las actividades del hotel
    public int getActivities(){
        this.reset();
        this.type= "getActivities";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener las actividades del hotel que están disponibles
    public int getAvailablesActivities(){
        this.reset();
        this.type= "getAvailablesActivities";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener los servicios del hotel
    public int getServices(){
        this.reset();
        this.type= "getServices";
```

```
        return 1;
    }

        //Método que prepara las propiedades para lanzar la petición
de obtener los servicios del hotel que están disponibles
    public int getAvailablesServices(){
        this.reset();
        this.type= "getAvailablesServices";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener la carta actual del hotel
    public int getCarta(){
        this.reset();
        this.type= "getCarta";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener la galería del hotel
    public int getGallery(){
        this.reset();
        this.type= "getGallery";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
obtener las opciones de la habitación actuales
    public int getRoomOptions(String roomId){
        this.reset();
        this.type= "getRoomOptions";
        this.id= roomId;
        return 1;
    }

    //////////////////INSERTS/////////////////

    //Método que prepara las propiedades para lanzar la petición de
insertar comentarios en el timeline
    public int insertPost(String text, String acclaim, String
category, String cat_type, String cat_id){
        this.reset();
        this.type= "insertPost";
        this.table= "posts";
        this.columns= "text, acclaim, category, cat_item_name,
cat_item_id";
        this.colValues= "'"+text+"', '"+acclaim+"', '"+cat_type+"',
'"+category+"', '"+cat_id+"'";
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
insertar un chat en el listado de chats
    public int insertChat(String id){
        this.reset();
        this.type= "insertChat";
        this.table= "chats";
        this.id= id;
        return 1;
    }
```

```
    //Método que prepara las propiedades para lanzar la petición de
insertar un mensaje dentro de un chat concreto
    public int insertMessage(String text, String chat_id, String
receptorChatUser){
        this.reset();
        this.type= "insertMessage";
        this.table= "messages";
        this.columns= "text";
        this.colValues= "'"+text+"'";
        this.id= chat_id;
        this.limit= receptorChatUser; //Enviamos en limit el receptor
user
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
insertar una valoración de un post del timeline general
    public int insertVote(String vote, String post_id){
        this.reset();
        this.type= "insertVote";
        this.table= "votes";
        this.columns= "vote";
        this.colValues= vote;
        this.id= post_id;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
insertar una valoración de un post del timeline de actividades
    public int insertActivityVote(String vote, String post_id){
        this.reset();
        this.type= "insertActivityVote";
        this.table= "votes_activities";
        this.columns= "vote";
        this.colValues= vote;
        this.id= post_id;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
insertar una valoración de un post del timeline de servicios
    public int insertServiceVote(String vote, String post_id){
        this.reset();
        this.type= "insertServiceVote";
        this.table= "votes_services";
        this.columns= "vote";
        this.colValues= vote;
        this.id= post_id;
        return 1;
    }

    //Método que prepara las propiedades para lanzar la petición de
insertar un cambio en las opciones de habitación del usuario
    public int insertRoomService(String id, String valor){
        this.reset();
        this.type= "insertRoomService";
        this.id= id;
        this.colValues= valor;
        return 1;
    }
```

```
////////////////////// DELETES ////////////////////////////////

    //Método que prepara las propiedades para lanzar la petición de
eliminar un post
    public int deletePost(String postId){
        this.reset();
        this.type= "deletePost";
        this.id= postId;
        return 1;
    }

/*
    Método que lanza la petición adecuada según sea necesario.
    En función de la petición a realizar, añadirá en formato "Clave-
Valor" los datos necesarios y hará
    la petición pasando el tipo, que será recogido en PHP.
*/

    protected JSONArray doInBackground(String... urls) {
        String result = "";
        InputStream is = null;
//http post
        try{
            MCrypt mcrypt = new MCrypt();

            ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
            if(this.type!=null)nameValuePairs.add(new
BasicNameValuePair("type",MCrypt.bytesToHex(mcrypt.encrypt(this.type))
));
            if(this.query!=null)nameValuePairs.add(new
BasicNameValuePair("query",MCrypt.bytesToHex(mcrypt.encrypt(this.query
))));
            if(this.columns!=null)nameValuePairs.add(new
BasicNameValuePair("columns",MCrypt.bytesToHex(mcrypt.encrypt(this.col
umns))));
            if(this.colValues!=null)nameValuePairs.add(new
BasicNameValuePair("values",MCrypt.bytesToHex(mcrypt.encrypt(this.colV
alues))));
            if(this.table!=null)nameValuePairs.add(new
BasicNameValuePair("table",MCrypt.bytesToHex(mcrypt.encrypt(this.table
))));
            if(this.condition!=null)nameValuePairs.add(new
BasicNameValuePair("condition",MCrypt.bytesToHex(mcrypt.encrypt(this.c
ondition))));
            if(this.user!=null)nameValuePairs.add(new
BasicNameValuePair("user",MCrypt.bytesToHex(mcrypt.encrypt(this.user))
));
            if(this.password!=null)nameValuePairs.add(new
BasicNameValuePair("password",MCrypt.bytesToHex(mcrypt.encrypt(this.pa
ssword))));

            if(this.id!=null)nameValuePairs.add(new
BasicNameValuePair("id",MCrypt.bytesToHex(mcrypt.encrypt(this.id))));
            if(this.limit!=null)nameValuePairs.add(new
BasicNameValuePair("limit",MCrypt.bytesToHex(mcrypt.encrypt(this.limit
))));
```

```
            Globals global = Globals.getInstance();
            String user= global.getUser();
            String logincode= global.getLogincode();

            if(user!=null)nameValuePairs.add(new
BasicNameValuePair("user",MCrypt.bytesToHex(mcrypt.encrypt(user))));
            if(logincode!=null)nameValuePairs.add(new
BasicNameValuePair("logincode",MCrypt.bytesToHex(mcrypt.encrypt(loginc
ode))));



            HttpClient httpclient = new DefaultHttpClient();
            HttpPost httppost=null;

            httppost = new HttpPost(this.urlWebService +
"dbconnection.php");

            httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            is = entity.getContent();
        }catch(Exception e){
            Log.e("log_tag", "Error in http connection " +
e.toString());
            return null;
        }
        //Convertimos la respuesta a string
        try{
            BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();

            result=sb.toString();
        }catch(Exception e){
            Log.e("log_tag", "Error converting result "+e.toString());
            return null;
        }

        //Procesamos el JSON data
        try{
            JSONArray jArray = new JSONArray(result);
            JSONObject json_data= jArray.getJSONObject(0);

            if(json_data.has("response") &&
json_data.getString("response").equals("WRONG_LOGINWITHCODE")){
                Globals global = Globals.getInstance();
                global.reset();

                Activity a=(Activity) HissAcclaim.getContext();


                MainActivity.parseLogOut();
                a.deleteFile("logincode");
```

```
                Intent intent = new Intent(a.getApplicationContext(),
MainActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                intent.putExtra("EXIT", true);
                a.startActivity(intent);
            }


/*
            for(int i=0;i<jArray.length();i++){
                JSONObject json_data = jArray.getJSONObject(i);
                Log.i("log_tag","id: "+json_data.getInt("id")+
                                ", name:
"+json_data.getString("name")+
                                ", sex: "+json_data.getInt("sex")+
                                ", birthyear:
"+json_data.getInt("birthyear")
                );
            }
*/


            return jArray;
        }catch(JSONException e){
            Log.e("log_tag", "Error parsing data "+e.toString());
            return null;
        }


    }


    /*
///////////////// Sin uso aun ////////////////
    public int changeUrlMySQL(String url){
        this.urlWebService= url;
        return 1;
    }

    public String type_last_petition() {
        return this.type;
    }

    //This Method is called when Network-Request finished
    protected void onPostExecute(String serverData) {
        //textView.setText(serverData);

    }
//////////////////GENERICS/////////////////


    public int genericQuery(String table, String columns, String
condition, String colValues){
        this.reset();
        this.type= "genericQuery";
        this.table= table;
        this.columns= columns;
        this.condition= condition;
        this.colValues= colValues;
        return 1;
    }
```

```
    public int getTable(String table){
        this.reset();
        this.type= "getTable";
        this.table= table;
        return 1;
    }

    public int insertRow(String table, String columns, String
colValues){
        this.reset();
        this.type= "insertRow";
        this.table= table;
        this.columns= columns;
        this.colValues= colValues;
        return 1;
    }

    public int searchByColumn(String table, String columns, String
condition, String colValues){
        this.reset();
        this.type= "searchByColumn";
        this.table= table;
        this.columns= columns;
        this.condition= condition;
        this.colValues= colValues;
        return 1;
    }

    public int deleteRow(String table, String columns, String
condition, String colValues){
        this.reset();
        this.type= "deleteRow";
        this.table= table;
        this.columns= columns;
        this.condition= condition;
        this.colValues= colValues;
        return 1;
    }
*/


}
```

9.4 Clase PushChatBroadcastReceiver

Esta es la clase que se usa para gestionar todo lo que tiene que ver con las notificaciones, extiende de la clase básica de Android BroadcastReceiver y está declarada en el archivo AndroidManifest.xml para su correcto lanzamiento, incluso cuando la aplicación está en segundo plano o sin uso.

Se encarga de gestionar las notificaciones que llegan para indicar que se ha recibido un nuevo mensaje (que son enviadas desde la aplicación del usuario que envía el mensaje). En caso de que el usuario receptor no esté usando la aplicación en el momento de recibir la notificación, se lanza una notificación de sistema que aparece en la barra superior de notificaciones de Android.

Si por el contrario el usuario se encuentra usando la aplicación, la notificación no se lanza para no molestar innecesariamente.

En ambos casos encarga de indicar que hay mensajes nuevos activando un icono sobre el apartado Chats en el menú principal y en cada chat.

Además de gestionar las notificaciones de nuevo mensaje, se encarga de recibir y gestionar la notificación que indican que un chat ha sido leído por el receptor. package com.hissacclaim.pfc.pfchissacclaim;

```
import android.app.Activity;
import android.app.Fragment;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.media.RingtoneManager;
import android.util.Log;
import android.widget.Button;

import com.parse.ParseInstallation;

import org.json.JSONObject;

/**
 * Created by Gran__000 on 27/03/2015.
 */
public class PushChatBroadcastReceiver extends BroadcastReceiver {


    //Si tenemos la aplicación abierta, ponemos una marca en rojo en
los chats si es necesario
```

```
    @Override
    public void onReceive(Context context, Intent intent) {

        if(!HissAcclaim.ParseIsLogOut()){
            String alert = null;
            String chatid = null;
            String tipo = null;
            try {
                JSONObject json = new
JSONObject(intent.getExtras().getString("com.parse.Data"));

                if (json.has("text"))
                    alert = json.getString("text");
                if (json.has("chatid"))
                    chatid = json.getString("chatid");
                if (json.has("type"))
                    tipo = json.getString("type");
            } catch (Exception e) {
                Log.d("onReceivePush", "JSONException: " +
e.getMessage());
            }
            Globals global = Globals.getInstance();
            //Si tenemos un contexto
            if (HissAcclaim.getContext() != null) {
                Activity a = (Activity) HissAcclaim.getContext();
                Fragment chatMessages =
a.getFragmentManager().findFragmentByTag("LIST_MESSAGES");
                Fragment mainmenu =
a.getFragmentManager().findFragmentByTag("MAIN_MENU");
                Fragment chats =
a.getFragmentManager().findFragmentByTag("CHATS");

                //Añadimos un check Leído si ya ha sido leído el
mensaje
                if (tipo != null &&
tipo.equals("notificationChatLeido")) {
                    if (HissAcclaim.isActivityVisible() && chats !=
null && chats.isVisible())

a.getFragmentManager().beginTransaction().detach(chats).attach(chats).
commit();
                } else {
                    //Si no, añadimos una marca roja de nuevo mensaje
                    if (HissAcclaim.isActivityVisible() &&
chatMessages != null && chatMessages.isVisible() &&
global.getChat_id().equals(chatid)) {

a.getFragmentManager().beginTransaction().detach(chatMessages).attach(
chatMessages).commit();
                    } else {
                        if (((chats != null && !chats.isVisible()) ||
(chats == null)) && ((chatMessages == null) ||
!chatMessages.isVisible() || (chatMessages.isVisible() &&
!global.getChat_id().equals(chatid))) && mainmenu != null) {
                            Button icono = (Button)
a.findViewById(R.id.chats_menu);

icono.setCompoundDrawablesWithIntrinsicBounds(null,
a.getResources().getDrawable(R.drawable.ic_new_message_rojo), null,
null);
                        }
```

```
                    }


                    if (HissAcclaim.isActivityVisible() && chats !=
null && chats.isVisible()) {

a.getFragmentManager().beginTransaction().detach(chats).attach(chats).
commit();
                    }


                    if (!HissAcclaim.isActivityVisible())
                        generateNotification(context, R.drawable.logo,
alert);
                }
            } else {
                generateNotification(context, R.drawable.logo, alert);
            }
        }else{
            ParseInstallation installation =
ParseInstallation.getCurrentInstallation();
            installation.put("userid", "logout");
            installation.saveInBackground();
        }
    }
    //Método para mostrar la notificación en el sistema Android
    //Sólo se llama en caso de que no tengas la aplicación abierta y
visible

    public static void generateNotification(Context context, int icon,
String message) {
        // Show the notification
        long when = System.currentTimeMillis();
        NotificationManager notificationManager =
(NotificationManager)context.getSystemService(Context.NOTIFICATION_SER
VICE);
        Notification notification = new Notification(icon, message,
when);
        String title = context.getString(R.string.app_name);
        Intent notificationIntent = new Intent(context,
MainActivity.class);

        // set intent so it does not start a new activity
        notificationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        PendingIntent intent = PendingIntent.getActivity(context, 0,
notificationIntent, 0);
        notification.setLatestEventInfo(context, title, message,
intent);
        notification.vibrate = new long[] { 500, 500 };
        notification.sound =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        notification.flags =
                Notification.FLAG_AUTO_CANCEL |
                        Notification.FLAG_SHOW_LIGHTS;

        notificationManager.notify(0, notification);
    }

}
```

9.5 Server dbconecction.php

Este es el fichero principal de la interfaz con la base de datos para gestionar las peticiones del cliente Android. Se encarga de recoger una petición del cliente, comprobar las credenciales del usuario, identificar el tipo de la petición e incluir el fichero con las operaciones oportunas que además devolverá el valor oportuno al cliente.

```
<?php
//Se incluye la librería que nos permite desencriptar los datos
recibidos
    include_once('class.MCrypt.php');

//Configuramos la hora correcta para obtener las horas correctas de la
BD
    define('TIMEZONE', 'Europe/London');
    date_default_timezone_set(TIMEZONE);
    $now = new DateTime();
    $mins = $now->getOffset() / 60;
    $sgn = ($mins < 0 ? -1 : 1);
    $mins = abs($mins);
    $hrs = floor($mins / 60);
    $mins -= $hrs * 60;
    $offset = sprintf('%+d:%02d', $hrs*$sgn, $mins);

//Conectamos a la base de datos con el $offset
    mysql_connect("localhost","root","");
    mysql_select_db("hissacclaim");
    mysql_query("SET time_zone='$offset';");

//Creamos una variable de clase MCrypt que usaremos para desencriptar
(en esta clase
//ya están definidos los códigos oportunos)
    $mcrypt = new MCrypt();
//Obtenemos el tipo de la petición y lo desencriptamos
    $tipo= $mcrypt->decrypt($_REQUEST['type']);
    $user_id="null";

//A continuación iremos revisando el tipo de la petición y si es
alguna de
//las conocidas se incluirá el archivo que la gestiona y que devolverá
los
//valores correspondientes

//Aquí se puede controlar error de tipo de petición no asignado si
//fuera conveniente

    if($tipo==null)return;

//Comprobamos si el tipo es login en tal caso se realizarán acciones
para no
//tener que loguear de nuevo al usuario introduciendo su contraseña

    elseif($tipo=="login")include('login.php');
//Si la petición es tipo loginCode quiere decir que el usuario ya se
logueo
```

```
//con anterioridad con su contraseña. Comprobamos si su código
almacenado en
//el dispositivo es válido y le permitimos acceder
    elseif($tipo=="loginCode")include('LoginCode.php');

    elseif($tipo=="getCarta")include('GetCarta.php');
    elseif($tipo=="getGallery")include('GetGallery.php');
    elseif($tipo=="getServices")include('GetServices.php');
    elseif($tipo=="getActivities")include('GetActivities.php');
    elseif($tipo=="getActivityData")include('GetActivityData.php');
    elseif($tipo=="getServiceData")include('GetServiceData.php');

    else{
//En cualquier otro caso la petición no será de acceso sino de otro
tipo.

//Aquí cargaremos el fichero que comprueba si el usuario tiene un
código de
//acceso valido y luego pasamos a gestionar su petición, si lo tiene.

        include('loginwithcode.php');
//A continuación se va incluyendo el fichero correspondiente a la
operación
        if($tipo=="insertPost")include('InsertPost.php');
//En este caso concreto se configuran un par de variables para
insertar una
//valoración correctamente según su tipo
        elseif($tipo=="insertVote"){
            $tablaSecundaria= "posts";
            $cabeceraTablaSecundaria= "post";
            include('InsertVote.php');
        }
        elseif($tipo=="insertActivityVote"){
            $tablaSecundaria= "activities";
            $cabeceraTablaSecundaria= "activity";
            include('InsertVote.php');
        }
        elseif($tipo=="insertServiceVote"){
            $tablaSecundaria= "services";
            $cabeceraTablaSecundaria= "service";
            include('InsertVote.php');
        }
        elseif($tipo=="insertChat")include('InsertChat.php');
        elseif($tipo=="insertMessage")include('InsertMessage.php');

elseif($tipo=="insertRoomService")include('InsertRoomService.php');
        elseif($tipo=="getActualUser")include('GetActualUser.php');

elseif($tipo=="getUserProfileData")include('GetUserProfileData.php');
        elseif($tipo=="editMyProfile")include('EditMyProfile.php');
        elseif($tipo=="getPosts")include('GetPosts.php');

elseif($tipo=="getProfilePosts")include('GetProfilePosts.php');

elseif($tipo=="getActivityPosts")include('GetActivityPosts.php');

elseif($tipo=="getServicePosts")include('GetServicePosts.php');
        elseif($tipo=="getSearchPosts")include('GetSearchPosts.php');

elseif($tipo=="getSearchPeople")include('GetSearchPeople.php');
        elseif($tipo=="getChats")include('GetChats.php');
```

```
        elseif($tipo=="getMessages")include('GetMessages.php');

elseif($tipo=="getAvailablesActivities")include('GetAvailablesActiviti
es.php');


elseif($tipo=="getAvailablesServices")include('GetAvailablesServices.p
hp');
        elseif($tipo=="getRoomOptions")include('GetRoomOptions.php');


        elseif($tipo=="deletePost")include('DeletePost.php');


        elseif($tipo=="genericQuery")include('GenericQuery.php');
        elseif($tipo=="getTable")include('GetTable.php');
        elseif($tipo=="searchByColumn")include('SearchByColumn.php');
        elseif($tipo=="insertRow")include('InsertRow.php');
        elseif($tipo=="deleteRow")include('DeleteRow.php');
        else{
            print(json_encode(array('response' => 'ERROR',
'data'=>'Error al interpretar el tipo del query')));
            return;
        }
    }


    mysql_close();
?>
```

10 Resultados y conclusiones

Tal y como se explicó en el capítulo 3 Objetivos el objetivo principal de este proyecto es brindar, a los clientes de un hotel o cadena hotelera, una herramienta que les permita comunicarse con los demás clientes del hotel, además de interactuar con el propio hotel y poder encontrar información sobre sus servicios y actividades, a la vez que el hotel consigue mejorar los servicio en base a las opiniones. Además, es posible extraer conclusiones si evaluamos el resultado del proyecto desde diferentes perspectivas.

10.1 Cumplimiento de objetivos

Desde el punto de vista de los objetivos del proyecto, el grado de cumplimiento puede considerarse muy alto, según las siguientes consideraciones:

1. Los requisitos planteados en el punto 7.1.1 Requisitos han sido plenamente satisfechos.
2. Se ha desarrollado una aplicación que es soportada por la plataforma móvil más extendida del mercado.
3. El aprendizaje de programación para Android ha sido considerable y será de gran utilidad para el futuro.

10.2 Tecnologías

Desde el punto de vista de las tecnologías, la conclusión es negativa, ya que para dispositivos móviles siempre hay que tomar una decisión que es crucial: orientarte hacia Android o hacia IOS. Esto es debido a que desarrollar la aplicación para ambos sistemas conllevaría un gran esfuerzo e implicaría posiblemente utilizar un framework de desarrollo multiplataforma tipo "Phonegap". Sin embargo el empleo de un entorno de este tipo a su vez implicaría, muy probablemente, una pérdida de rendimiento de la aplicación al no haber sido implementada de forma nativa.

En cuanto a la experiencia en Android, la dificultad de desarrollo actual para formarse y realizar una aplicación con garantías es alta. Es extraño que Google no haya

puesto más empeño para facilitar a los desarrolladores su labor para realizar aplicaciones en Android.

Las principales dificultades que hemos encontrado en Android son:

- Versiones de Android

  Existen, hasta el momento, veintiuna versiones de Android diferentes. Muchas de ellas, obsoletas, pero las más nuevas se encuentran en un reducido porcentaje de dispositivos en el mercado. Es complejo elegir para qué versión realizar una aplicación, ya que tienes que intentar llegar al mayor número de dispositivos, pero a su vez existen muchas funciones para el desarrollador que no son válidas en versiones obsoletas.

- Interfaces en Android

  Las interfaces en Android son muy duras de implementar. Quizá en una aplicación muy básica en la que el número de vistas no supera las cuatro o cinco, no supone un mayor problema, pero según nuestra experiencia, es bastante complejo elaborar interfaces bonitas y de calidad debido a la gran variedad de resoluciones de pantalla y densidad de píxeles de los dispositivos que soportan Android.

- Falta de funcionalidad básica

  Muchas veces, para desarrollar algo que en otras plataformas o lenguajes de programación es muy sencillo, en Android ha sido incomprensiblemente costoso. El ejemplo más sencillo que tuvimos que sufrir fue al querer usar un “wait” para poder hacer comprobaciones, pero en innumerables casos han surgido problemas que complicaban el código de manera incomprensible.

- Entorno de programación

  Actualmente, los dos entornos de programación que más auge tienen para Android son Eclipse y Android Studio. Eclipse es un buen entorno de programación, pero consume demasiados recursos y no está orientado exclusivamente a Android por lo que no puedes exprimir su rendimiento, mientras que Android Studio sí lo está. El problema de este último es que hasta hace unos escasos cuatro meses estaba en

versión beta y, su cantidad de bugs y la inestabilidad del entorno, complicaban el desarrollo.

10.3 ¿Es posible llevar este proyecto a la práctica?

Sin duda alguna, sí. El mercado para el que está pensado el proyecto no está saturado con este tipo de desarrollos y, el desarrollo en sí mismo, tiene una base ya estructurada e implementada sobre la que sostenerse. Con añadirle algunas funcionalidades y algunas mejoras gráficas en la interfaz (con la ayuda de un experto en diseño) sería un producto perfectamente vendible en el mercado.

10.4 Tecnología web vs tecnología móvil

Es la pregunta de moda actualmente. ¿Lo implementas en versión web o en versión móvil?

El auge de los dispositivos móviles en los últimos diez años ha sido espectacular, y la mayoría de los usuarios de nivel básico acceden a la información desde este tipo de dispositivos y no desde un ordenador.

Por otra parte, el desarrollo web tiene más rodaje y conseguir una aplicación web de garantías supone muchísimo menos esfuerzo para un desarrollador, pudiendo lograr el mismo objetivo en un tiempo muy reducido.

Nuestra conclusión es que si necesitas realizar un proyecto en un espacio de tiempo demasiado corto, es mejor hacerlo vía web. Si tienes más tiempo de margen y quieres ganar en mayor número de usuarios potenciales, el desarrollo móvil es la mejor solución.

10.5 Satisfacción personal

La satisfacción personal de haber desarrollado un proyecto de esta envergadura sin tener conocimientos previos de Android es muy alta.

Hoy en día la demanda de aplicaciones móviles en el mercado laboral es importante, por lo que este proyecto vale oro para nuestra preparación y experiencia.

11 Trabajo futuro

Todo producto software es siempre mejorable y en un entorno como es el turístico las posibilidades de desarrollo y evolución son casi infinitas. Por ello vamos a realizar un listado de tareas que creemos que son la guía a seguir desde este punto del proyecto hacia adelante.

- Rediseño de la interfaz con la participación de un diseñador.
- Desarrollar funcionalidad de usuarios favoritos.
- Desarrollar funcionalidad de lista de usuarios bloqueados (para cada usuario).
- Permitir editar comentarios del timeline.
- Incluir traducciones.
- Edición del apartado de habitación para hoteles que incorporan en sus habitaciones elementos domóticos.
- Permitir eliminar mensajes de los chat.
- Desarrollar módulo de reservas de actividades o servicios.
- Desarrollar módulo para permitir a los hoteles enviar ofertas o promociones a sus usuarios a través del sistema de notificaciones ya implementado.
- Añadir un módulo de comercios locales recomendados que además permita abrir una nueva línea de negocio.
- Versión iOS.

12 Bibliografía

La bibliografía utilizada se ha obtenido exclusivamente por medios digitales y ha consistido principalmente en manuales de las distintas herramientas y lenguajes de programación empleados. A continuación se especifican las direcciones de consulta más habituales agrupadas por tipos:

12.1 Cliente aplicación Android

- Guia oficial de introducción a Android

  http://developer.android.com/guide/index.html
- Guia oficial de referencia para el desarrollador en Android

  http://developer.android.com/reference/packages.html
- Manual oficial de Parse

  https://www.parse.com/docs/push_guide
- Manual de Universal Image Loader

  https://github.com/nostra13/Android-Universal-Image-Loader/wiki/Quick-Setup
- Manual de PhotoView

  https://github.com/chrisbanes/PhotoView

12.2 Cliente panel web

- Manual de HTML

  http://www.w3schools.com/html/
- Manual de Bootstrap

  http://www.w3schools.com/bootstrap
- Manual de jQuery

  http://www.w3schools.com/jquery/
- Manual de Dynatable

  http://www.dynatable.com/

12.3 Servidor

- Manual MCrypt

  https://github.com/chuyskywalker/phpaes

- Manual de PHP

  http://www.w3schools.com/php/

12.4 Información para estado del arte y estadísticas

- Elandroidlibre

  http://www.elandroidelibre.com/2013/05/los-fabricantes-de-android-mas-fuertes-que-nunca-y-samsung-con-el-95-de-los-beneficios.html

- Xakatandroid

  http://www.xatakandroid.com/mercado/android-comanda-con-mano-de-hierro-el-mercado-europeo

- Androiddeity

  http://androideity.com/2012/07/16/5-lenguajes-para-programar-en-android/

12.5 Consultas generales

- Stackoverflow para consultas de programación generales

  http://stackoverflow.com/

- Wikipedia para información genérica

  https://es.wikipedia.org

Anexo I - Manuales de usuario

1 Manual de aplicación Android Hiss&Acclaim

1.1 Instalación

Las aplicaciones de los dispositivos Android se instalan a través de la Play Store.

Imagen 62: Google play

La play Store es la tienda oficial de Google para la adquisición de aplicaciones para dispositivos con sistema operativo Android. Todos los dispositivos que funcionan con este sistema tienen incorporada la tienda.

Una vez nos encontramos dentro de la tienda, hay que buscar la aplicación Hiss & Acclaim y seguir las instrucciones.

1.2 Inicio de la aplicación

Una vez tenemos ya la aplicación descargada en nuestro dispositivo, solamente hay que pulsar sobre su icono para que se ejecute.

Imagen 63: Icono Hiss&Acclaim

1.3 Login

Nada más arrancar la aplicación por primera vez, nos llevará a la siguiente vista para poder acceder a ella:

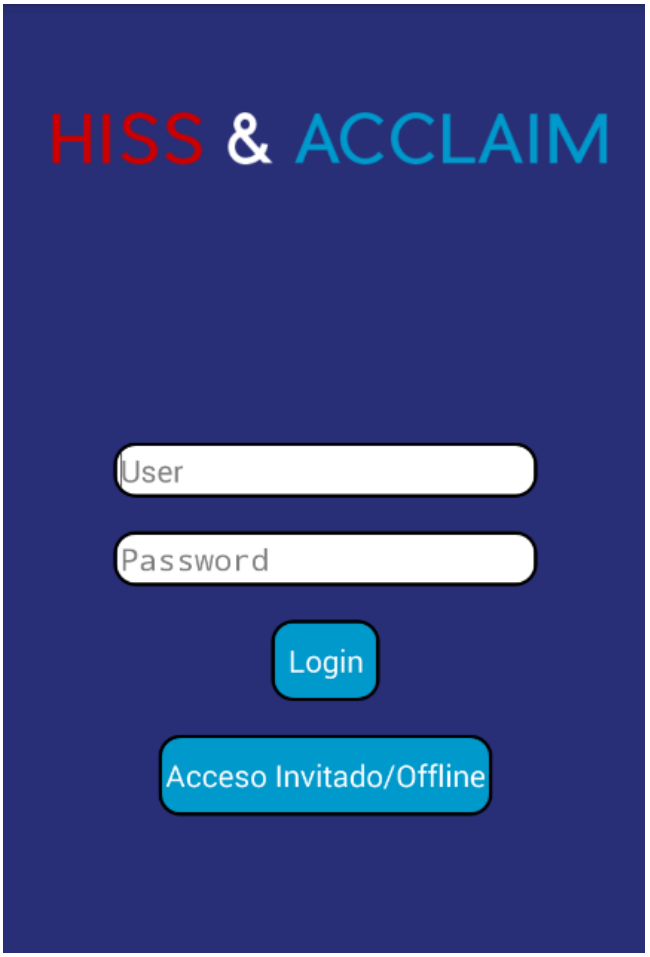


Imagen 64: Login Hiss&Acclaim

En ella, debemos introducir un usuario y una contraseña válidos. Si los datos introducidos son correctos, el sistema nos recordará y no será necesario volver a introducir los datos en el mismo dispositivo a menos que cerremos nuestra sesión o iniciemos sesión con el mismo usuario en otro dispositivo.

Si los datos son erróneos, nos saldrá un mensaje avisándonos de que hemos escrito mal nuestras credenciales.

En caso de no tener una conexión a internet disponible en el dispositivo, adicionalmente nos aparecerá un botón de acceso offline. Este acceso nos permitirá acceder directamente al apartado Hotel donde se podrán ver algunos datos sobre el

hotel que pueden ser de utilidad, como por ejemplo el mapa offline para llegar hasta el hotel.

1.4 Timeline

Al realizar un acceso correcto a la aplicación (con internet disponible), accederemos automáticamente al timeline.

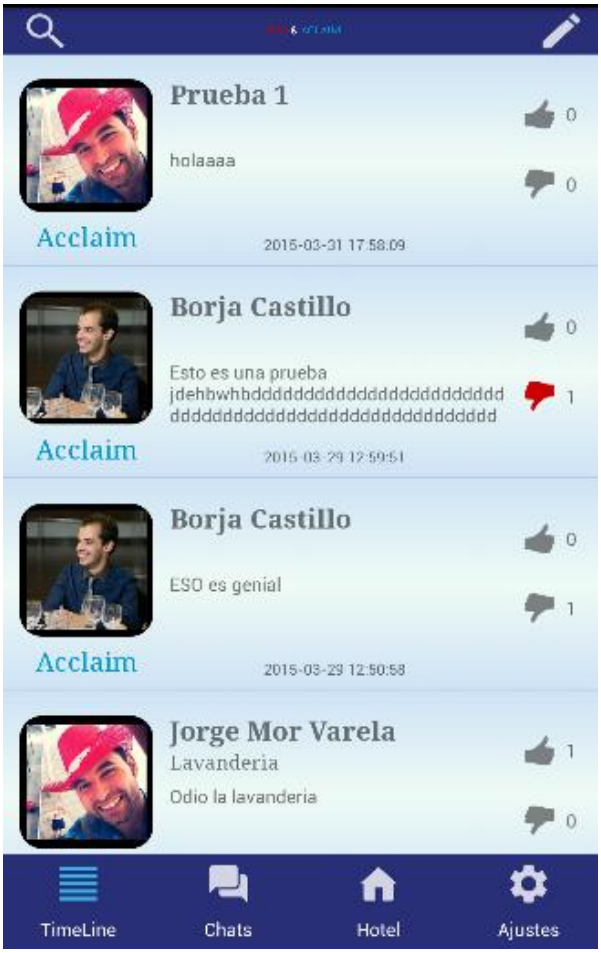


Imagen 65: Timeline Hiss&Acclaim

En esta vista tenemos un listado de comentarios de los usuarios del sistema y también nos aparecerá el menú principal en la parte inferior.

El menú principal consta de cuatro módulos: Timeline, chats, hotel y ajustes. Si queremos cambiar a otro módulo, basta con pinchar sobre dicha pestaña del menú inferior.

Dentro de la pestaña timeline, hay muchas acciones disponibles. Las iremos explicando de arriba a abajo y de izquierda a derecha.

1.4.1 *Buscar*

El icono de la lupa, en la esquina superior izquierda que podemos ver tambien a continuación, nos permite realizar búsquedas.

Imagen 66: Lupa Hiss&Acclaim

Si pinchamos sobre él, descenderá de la parte superior de la pantalla un recuadro que nos permitirá realizar la búsqueda.

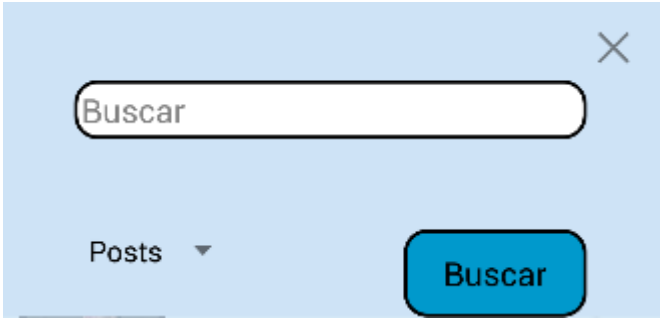


Imagen 67: Busqueda Hiss&Acclaim

Hay dos tipos de búsquedas, por persona y por comentario. En primer lugar, en el recuadro grande de fondo blanco podemos introducir el texto que queremos buscar. Debajo de él tenemos un desplegable sobre el que podemos pulsar para elegir cuál de las dos búsquedas queremos realizar. Una vez hayamos completado esto, bastará con pinchar sobre el botón “Buscar”.

Al buscar comentarios, la vista resultante será un nuevo timeline pero que contiene solamente los comentarios con los resultados de la búsqueda. En él, podemos hacer las funciones generales de un timeline que explicaremos más adelante en el punto “Funcionamiento general del timeline” o volver atrás pulsando en la flecha de la esquina superior izquierda.

Imagen 68: Atrás Hiss&Acclaim

Este evento de volver atrás se repetirá con asiduidad a lo largo de la aplicación.

Si buscamos personas, la vista resultante será un listado de personas (no un timeline). Podemos pulsar sobre el rectángulo de cada usuario y nos llevará al perfil

propio de ese usuario, que será explicado en el apartado “Acceso al perfil de un usuario”, o podemos de nuevo volver atrás.

1.4.2 Escribir comentario

Si pulsamos sobre el icono del lápiz de la esquina superior derecha, podremos escribir un nuevo comentario en el timeline.

Imagen 69: Lápiz Hiss&Acclaim

Una vez lo hayamos pulsado, descenderá de arriba una nueva vista con varias opciones que nos permitirá escribir el comentario.

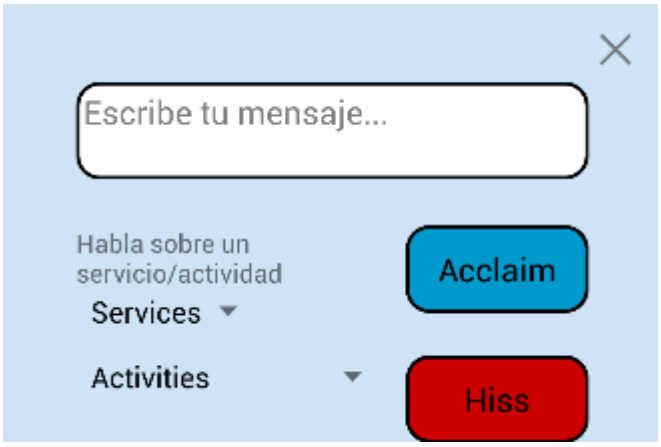


Imagen 70: Añadir publicación Hiss&Acclaim

En primer lugar, tenemos un recuadro con fondo blanco donde escribir nuestro mensaje. Éste tiene un límite de 140 caracteres.

Debajo a la izquierda, tenemos la opción de categorizar nuestro comentario. Podemos no seleccionar nada y dejarlo sin categoría, lo que implicaría que fuera un comentario genérico. También podemos seleccionar una actividad (pinchando sobre “Activities”, se abrirá un desplegable y elegimos una) o análogamente un servicio, pero nunca podremos elegir ambas. En caso de haber seleccionado ambas, nos saldrá un mensaje de error y debemos deseleccionar una de ellas.

Es importante señalar que en el caso del desplegable de actividades, solo aparecerán aquellas actividades cuya fecha de realización de actividad se haya cumplido hoy mismo o hace 3 días como máximo.

Por último, tenemos dos botones disponibles: “Hiss” y “Acclaim”. Solamente podemos pulsar uno de los dos. Acclaim sirve para indicar que es un mensaje bueno, mientras que Hiss sirve para indicar algo desgradable o queja. Tras pulsar sobre alguno de estos dos botones, volveremos a la vista general del timeline con nuestro mensaje ya publicado.

1.4.3 Funcionamiento general del timeline

En el timeline podemos ver un listado de comentarios de usuarios del sistema y, sobre cada rectángulo de la lista, podemos realizar las siguientes acciones:

1.4.3.1 Valoración.

A la derecha de cada comentario tenemos dos manos, una con el pulgar hacia arriba y otra con el pulgar hacia abajo.

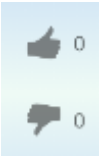


Imagen 71: Valoraciones Hiss&Acclaim

La mano con el pulgar hacia arriba sirve para valorar positivamente ese comentario, mientras que la que apunta hacia abajo hace lo contrario. Solamente puede haber una mano activa a la vez. Si ambas están en gris, la que pulsemos se activará. Sin embargo, si pulsamos sobre una que ya había sido activada, volveremos al estado original (es así como anulamos nuestra valoración). Además, podemos cambiar de opinión y pulsar directamente la mano opuesta, pero se cancelará la anterior.

1.4.3.2 Cargar más

Si bajamos por la vista del timeline, veremos que llegamos a una posición en la que aparece un botón con el mensaje “Cargar más”.

Imagen 72: Cargar más Hiss&Acclaim

El timeline va cargando mensajes de diez en diez. Cada vez que pulsemos este botón, aparecerán diez comentarios más en el listado. Si llegamos al final del todo y ya no hay más mensajes, nos aparecerá el mensaje: “No hay más resultados”.

Imagen 73: No hay más resultados Hiss&Acclaim

1.4.3.3 Acceso al perfil de un usuario

Si pinchamos en cualquier lugar de una publicación de la lista (sin ser las manos de valoración), accederemos al perfil propio de un usuario.

Imagen 74: Perfil de usuario Hiss&Acclaim

En él, veremos información general de su estado (nombre, apodo, imagen y estado). También, a la derecha de la imagen tenemos el botón “Chatear”, que nos

permitirá iniciar un chat con esa persona y nos llevará a una nueva vista de chat que será explicada más tarde en el punto “Abrir un chat iniciado anteriormente”.

Debajo de los datos de perfil, también se podrá ver un timeline que contendrá solamente comentarios de ese usuario y en él podremos desarrollar las funciones generales del timeline (a excepción de acceder a un perfil, dado que ya estamos en uno). Si estamos visualizando nuestro propio perfil podremos eliminar publicaciones hechas con anterioridad pulsando sobre la cruz que aparecerá en la esquina superior derecha de cada publicación.

Por último, también es posible volver atrás.

1.5 Chats

En el módulo de chats, tendremos un listado de conversaciones que hemos iniciado con otros usuarios. Los iconos de validación azules indican que la otra persona ya ha leído el chat, mientras que los “new” rojos indican que tenemos mensajes sin leer.

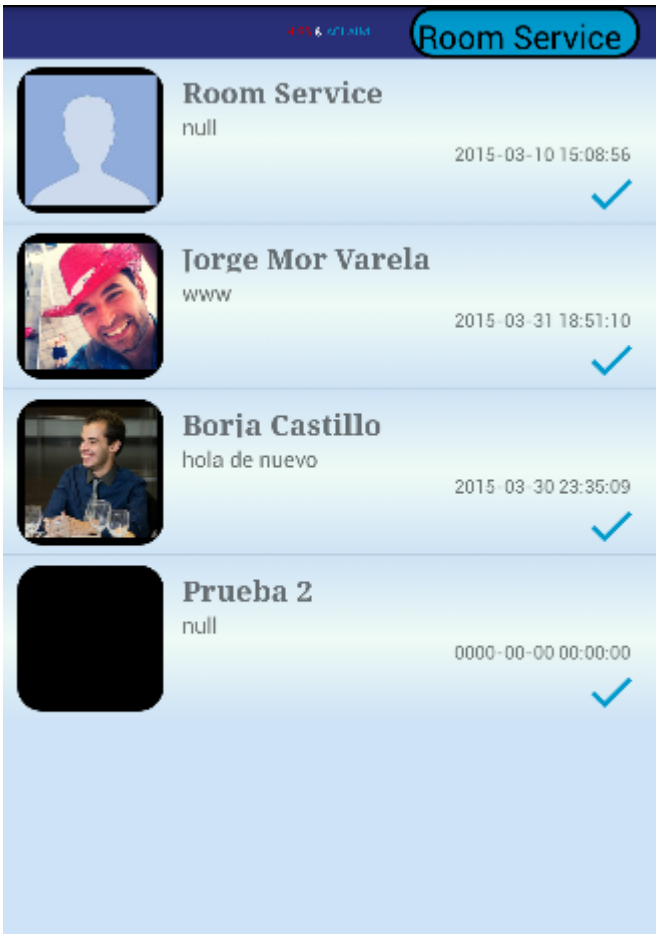


Imagen 75: Chats Hiss&Acclaim

En esta vista, tenemos dos funciones básicas, iniciar un chat con el servicio de habitaciones o abrir un chat ya iniciado anteriormente.

1.5.1 *Iniciar chat con el servicio de habitaciones*

En la esquina superior izquierda tenemos el botón "Room Service".

Imagen 76: Room Service Hiss&Acclaim

Si pulsamos sobre él, iniciaremos un chat con el servicio de habitaciones (a menos que ya lo tengamos abierto en la lista en cuyo caso nos avisará).

1.5.2 *Abrir un chat iniciado anteriormente*

Para ello, basta con pulsar sobre cualquier rectángulo del listado de chats. Esto nos llevará a una nueva vista del chat con la otra persona.

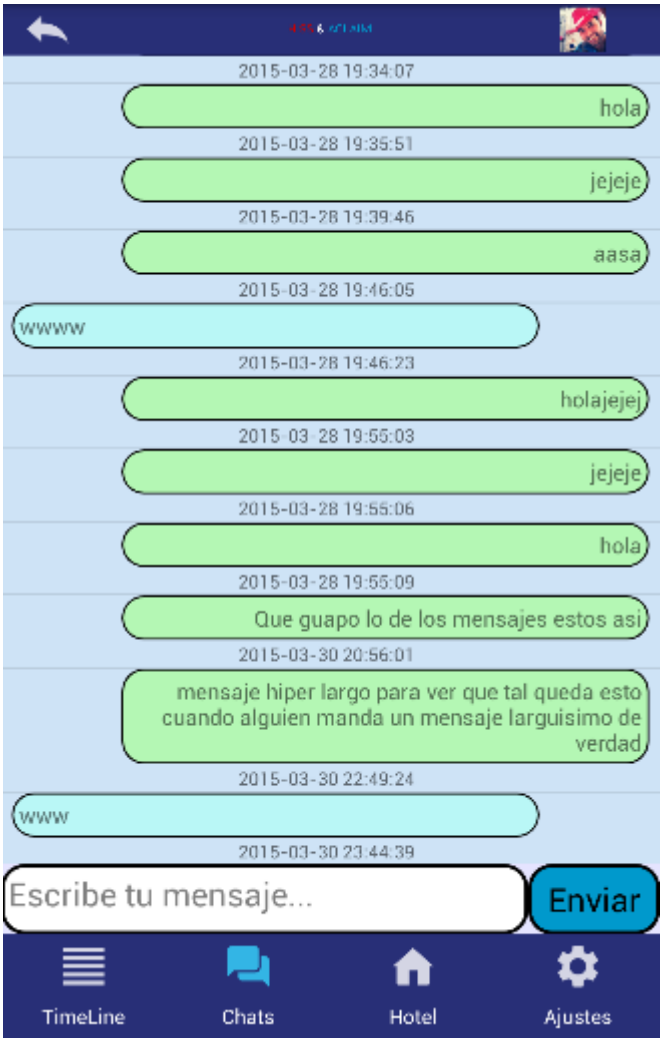


Imagen 77: Ventana de mensajes Hiss&Acclaim

1.5.3 Escribir un mensaje

Una vez hayamos accedido a la vista de un chat con una persona concreta, la acción principal que podemos realizar será enviar un mensaje. Para ello, basta con rellenar el campo con fondo blanco y pulsar sobre el botón “Enviar”.

También podemos volver atrás.

1.6 Hotel

Al acceder a la pestaña hotel, tenemos un listado con varias categorías accesibles. Para acceder a cada una de ellas, basta con pulsar una vez sobre la elegida.

Todas estas categorías se mantendrán disponibles offline para el usuario con la información disponible en cada apartado la última vez que accedió con una conexión a internet válida.

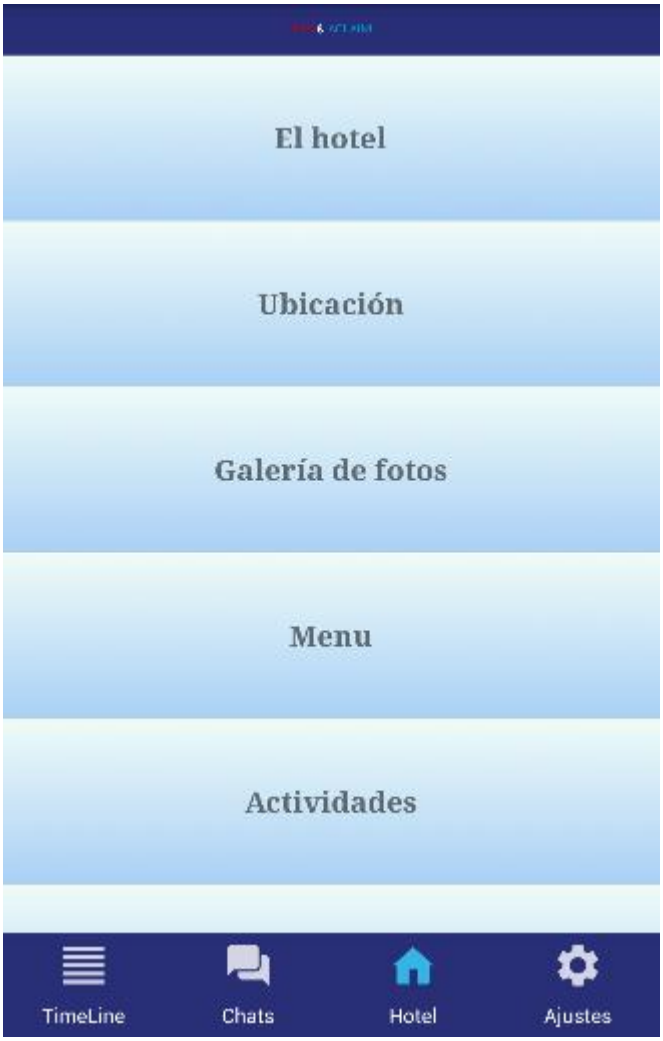


Imagen 78: Apartado Hotel Hiss&Acclaim

1.6.1 *El hotel*

Es una vista meramente informativa en la que la única acción posible es volver atrás.

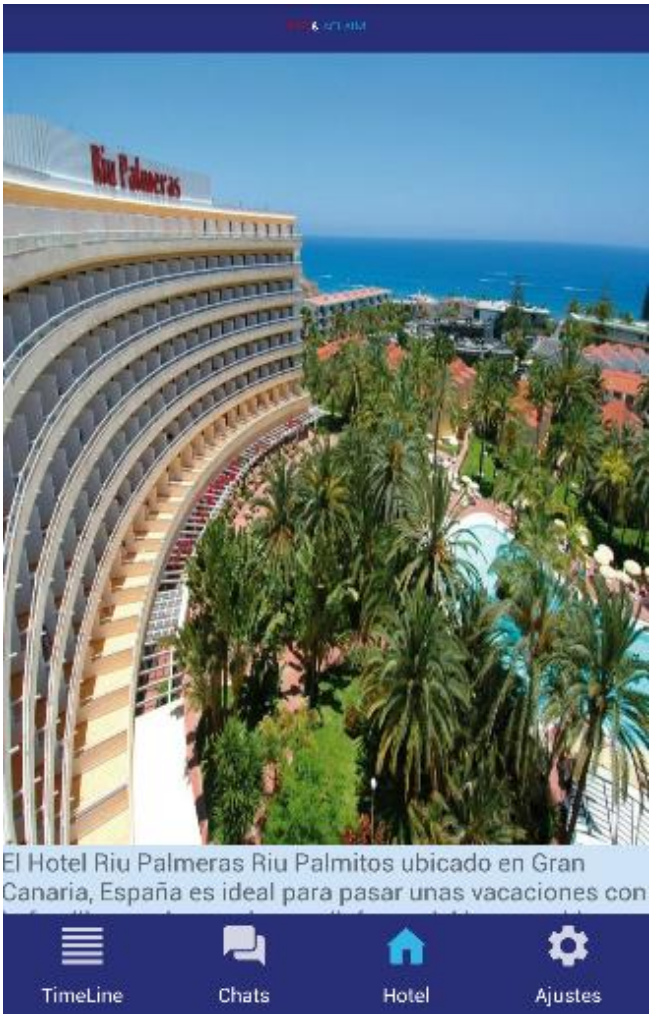


Imagen 79: Info Hotel Hiss&Acclaim

1.6.2 *Ubicación*

Las acciones sobre el apartado ubicación dependen de la disponibilidad de conexión a internet de nuestro dispositivo y de la versión del mismo. En caso de tener conexión, tendremos un mapa de Google Maps, mientras que en el modo sin conexión tendremos una imagen del mapa. Además hay que tener en cuenta que, si nuestro dispositivo es muy antiguo, es posible que el mapa de google no esté soportado o que tengamos que actualizar los permisos de Google para que funcione. En el modo sin conexión las únicas acciónes disponibles sobre la imagen del mapa serán ampliar o reducir la imagen con dos dedos y volver atrás.

Por otro lado, en el mapa de la versión con conexión podremos:

1.6.2.1 Ampliar y reducir el mapa

Para ello, utilizamos dos dedos pellizcando hacia dentro o hacia fuera, del modo clásico en dispositivos móviles. También hay un botón "+" y un botón "-" en el mapa para ello.

1.6.2.2 Desplazarnos

Podemos movernos por el mapa deslizando el dedo sobre él.

1.6.2.3 Marcar nuestra ubicación con GPS

Podemos marcar nuestra ubicación pulsando el botón circular. Esta función sólo funcionará si tienes el GPS activado.

1.6.3 Galería de fotos

Al acceder a la galería de fotos, podemos ver una vista general de las imágenes.

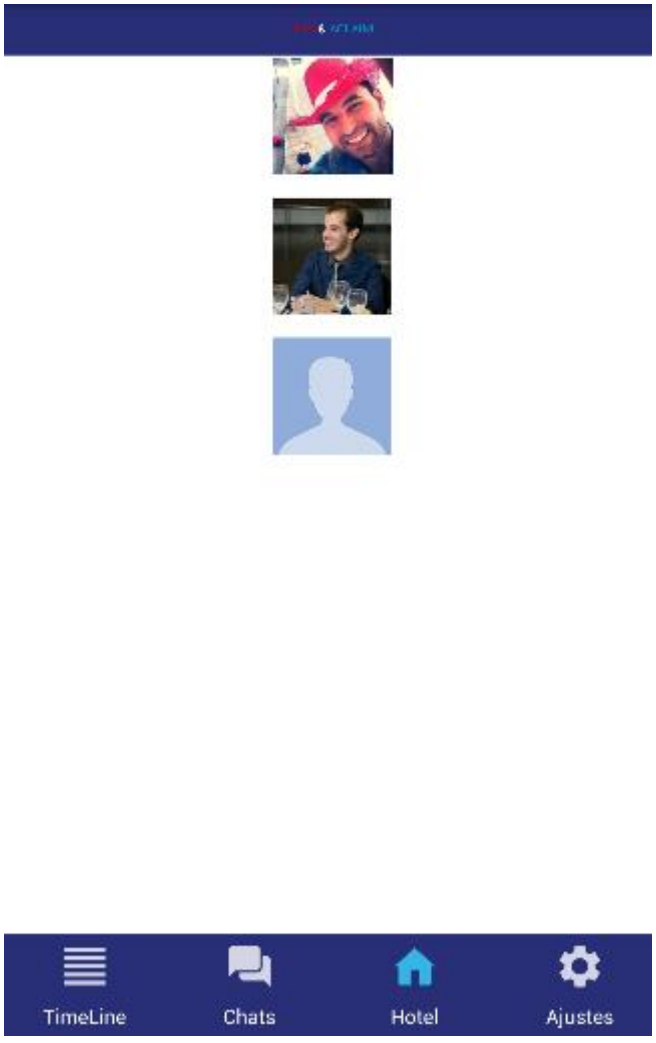


Imagen 80: Galería de imágenes Hiss&Acclaim

Para ver una imagen en grande, basta con pulsar sobre ella.

Podremos hacer zoom al estilo clásico, con dos dedos. También podemos volver atrás para regresar a la galería general.

Imagen 81: Vista de imagen-zoom Hiss&Acclaim

Estando en la galería general también podemos retroceder a la vista “Hotel”.

1.6.4 Menú

Al acceder a esta categoría, tenemos una imagen con la carta del hotel. En esta imagen podemos hacer zoom al estilo clásico, con dos dedos o pulsando sobre la imagen dos veces.

También podemos volver atrás.

1.6.5 *Actividades*

Al acceder a esta vista, tendremos una lista de actividades.

Imagen 82: Lista de actividades Hiss&Acclaim

En ella se mostrará la información general de las actividades disponibles y, tal y como está explicado en el apartado “Funcionamiento general del timeline”, se pueden realizar tambien aquí las funciones generales del timeline de valoración (pulsando sobre las manos) y acceso al “perfil” de la actividad pulsando sobre ella.

Imagen 83: Vista de una actividad Hiss&Acclaim

En esta vista, tenemos una visión general de la actividad y un timeline con todos los comentarios sobre esa actividad concreta. Estos comentarios son valorables. También se puede volver atrás.

En el timeline general de actividades tenemos la opción de retroceder.

1.6.6 Servicios

En la categoría servicios, todo es análogo al punto anterior "Actividades" pero su contenido cambia. Las acciones a realizar son exactamente las mismas.

1.7 Ajustes

Al acceder a la pestaña de ajustes, tenemos tres opciones disponibles en las que basta pulsar una vez para acceder a ellas:

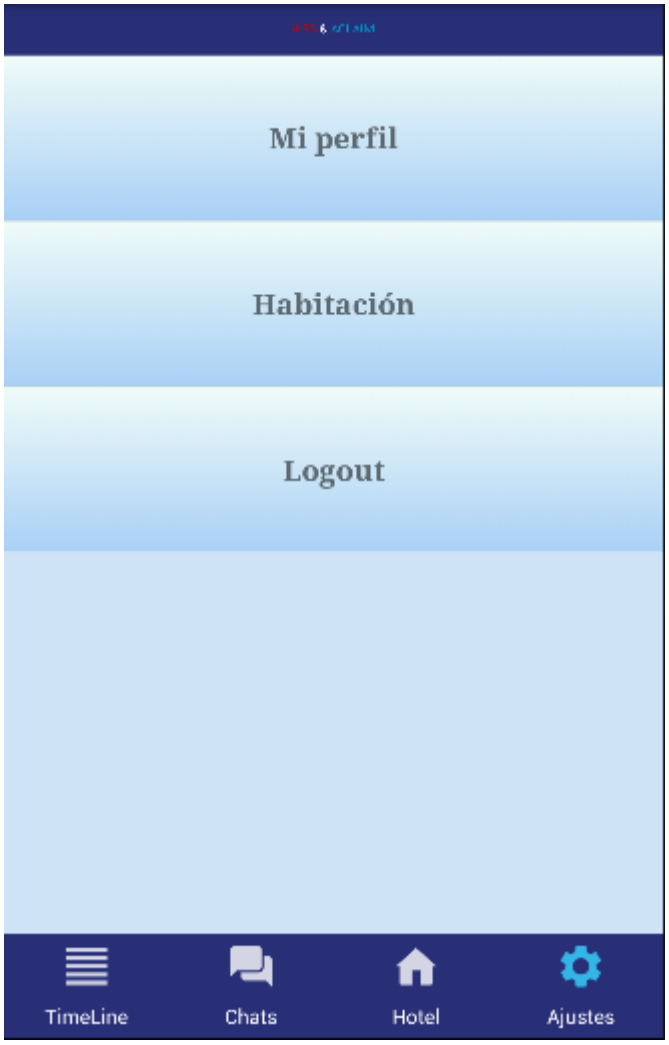


Imagen 84: Ajustes Hiss&Acclaim

1.7.1 *Mi perfil*

Esta vista sirve para configurar el perfil propio de usuario.

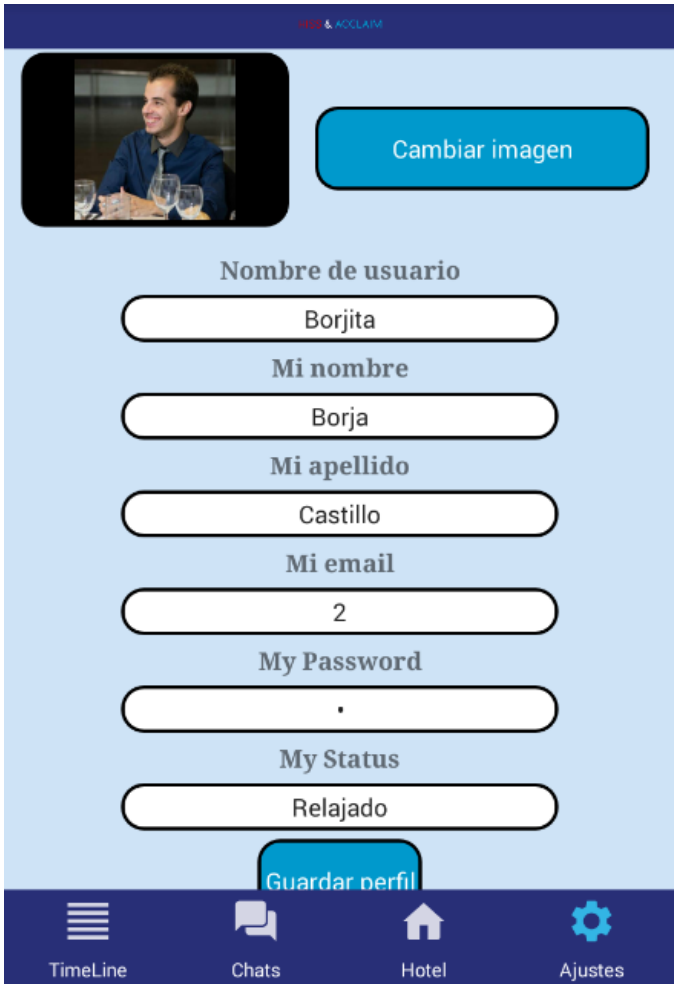


Imagen 85: Mi perfil Hiss&Acclaim

Para cambiar nuestra imagen, hay que pulsar sobre el botón “Cambiar imagen” y podremos sacar una nueva foto y cambiarla.

El resto de campos del perfil son editables y, al acabar de modificar los datos según nuestras necesidades, debemos pulsar el botón “Guardar perfil” que se encuentra al final de la vista.

Por último, podemos volver atrás.

1.7.2 *Habitación*

En esta vista, tenemos datos sobre nuestra habitación.

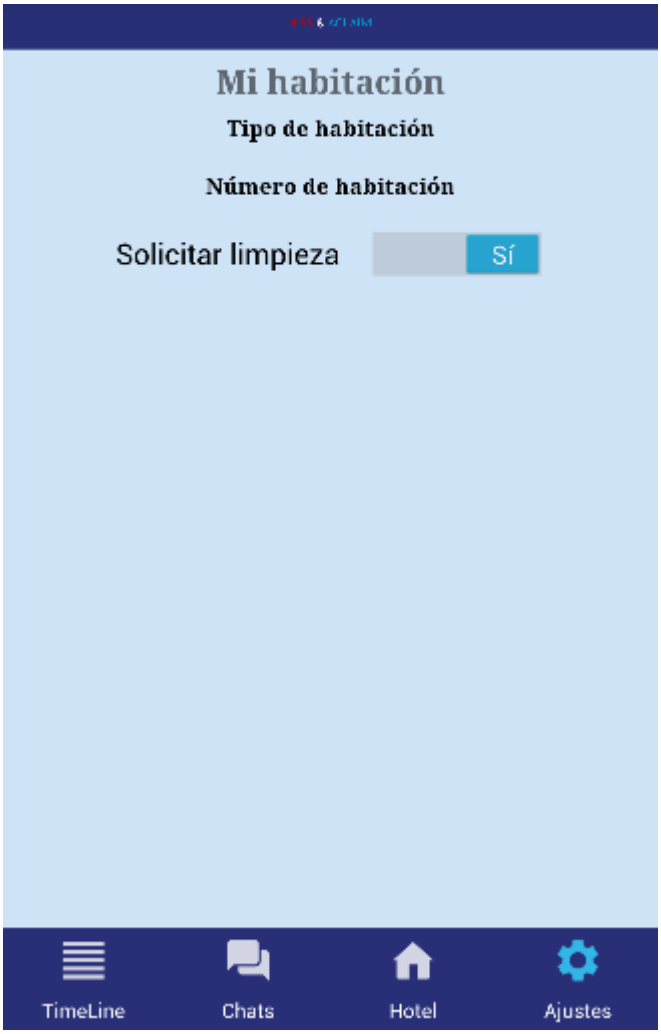


Imagen 86: Habitación Hiss&Acclaim

Podemos modificar desde la aplicación si queremos que el servicio de limpieza pase a limpiar nuestra habitación o no pulsando sobre el interruptor "Solicitar limpieza".

Actualmente solo se encuentra disponible esa opción aunque el número de opciones se podría incrementar en gran medida en función de las opciones que ofrezca el hotel o la habitación al cliente (pensando por ejemplo en habitaciones domotizadas).

También se puede volver atrás.

1.7.3 Logout

Al pulsar sobre "Logout" nos saldrá una ventana de confirmación de cerrar la sesión. Pulsaremos sí o no en función de nuestras necesidades.

2 Manual de panel de control HissAcclaim

2.1 Panel de control de administración

Para acceder al panel de control de administración necesitamos un navegador web y acceder a la dirección: "hissacclaim.morwebs.com/". Nos aparecerá la siguiente vista.

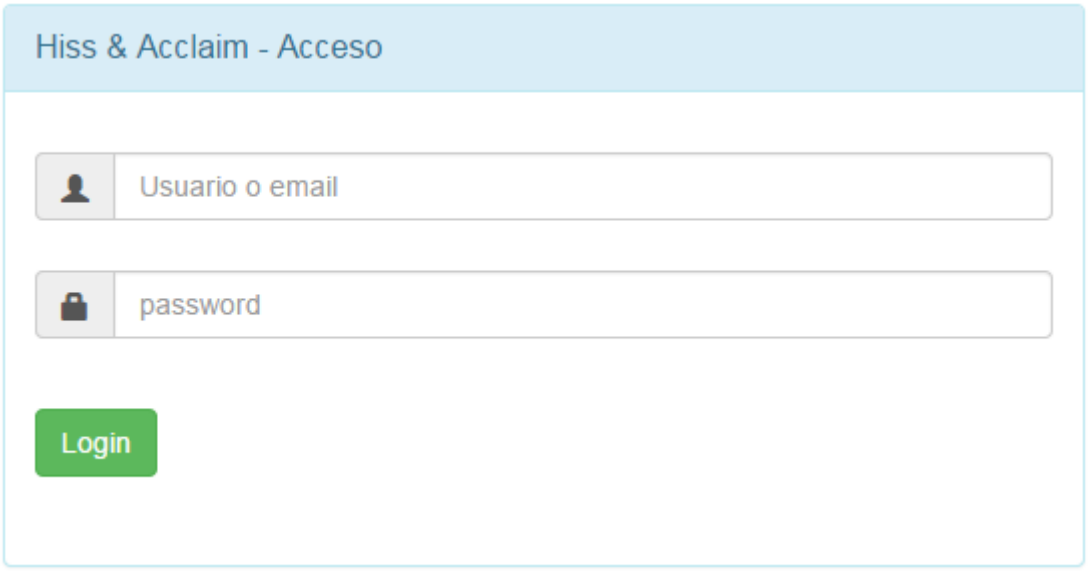


Imagen 87: Login panel Hiss&Acclaim

Para acceder al panel de control nos pedirá el usuario y una contraseña del administrador y, a continuación, pulsaremos el botón "Login". Si los datos son correctos, accederemos al panel de administración, si no lo son, nos saldrá un mensaje de error.

Una vez hayamos accedido al panel de control, esta será la vista general.

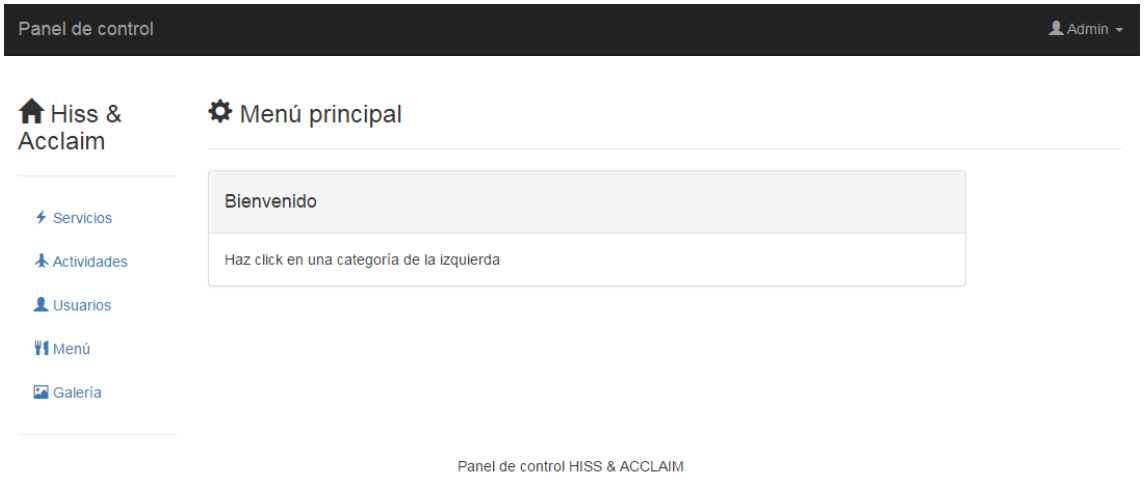


Imagen 88: Vista general panel Hiss&Acclaim

2.2 Acciones generales

Para ver los datos sobre cualquiera de las categorías de la izquierda, basta con pinchar sobre ella y se mostrarán en el contenedor principal los elementos existentes actualmente de esa categoría, paginados y con un buscador que permite buscar por cualquier campo de la tabla visible.

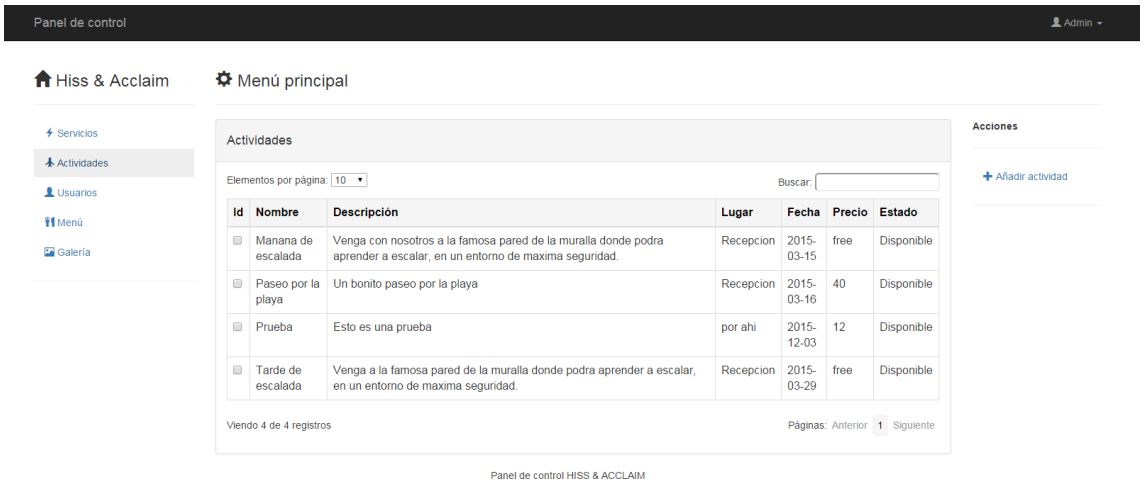


Imagen 89: Vista de una tabla Hiss&Acclaim

Para buscar, hay que rellenar el campo “Buscar” y pulsar Enter.

Para ver más elementos de la categoría, podemos pulsar “Siguiente” o “Anterior” en el menú de paginación o pulsar sobre una página concreta para ir a ella.

También es posible cambiar el número de elementos que se muestran por página en el desplegable “Elementos por página”.

Otra opción es seleccionar elementos de la tabla pulsando sobre la casilla “Id” correspondiente (o deseleccionarlas en caso necesario). Sobre los elementos seleccionados se pueden realizar acciones que estarán a la derecha del contenido y que se explicarán a continuación en cada apartado.

Para cambiar entre categorías, solamente tenemos que pulsar la nueva categoría a la que queremos acceder.

Si queremos cerrar sesión, pulsamos arriba a la derecha donde pone “Admin” y se nos abrirá un desplegable con la opción “Logout”. Pulsamos sobre ella y volveremos al “Login”.

2.3 Servicios

Al acceder a servicios, la acción inicial que podemos hacer es “Añadir servicio”. Si pulsamos sobre ella, se abrirá un modal que hay que rellenar para añadir el servicio. Al acabar de rellenar el formulario, pulsamos sobre “Añadir”.

Al marcar un elemento del listado de servicios, tenemos la opción “Editar servicio”. Si pulsamos sobre ella, nos saldrá un formulario con los datos actuales del servicio. Si queremos desechar los cambios, podemos pulsar la “x” en la esquina superior izquierda. Si queremos guardar los cambios pulsamos sobre “Guardar cambios”.

Si marcamos más de una opción en los servicios, la opción “Editar servicio” desaparecerá.

Mientras haya, al menos, un elemento marcado, tendremos la opción “Eliminar servicio”.

Imagen 90: Acciones Servicios Hiss&Acclaim

Al pulsar sobre ella, se abrirá un mensaje de confirmación para asegurarnos de que queremos eliminar los servicios que han sido seleccionados. Pulsaremos sí o no en función de nuestras necesidades.

2.4 Actividades

Las acciones que se pueden realizar sobre actividades son análogas a las de servicios.

Imagen 91: Acciones Actividades Hiss&Acclaim

2.5 Usuarios

En la categoría de usuarios, también podemos realizar las tres acciones explicadas en el apartado "Servicios" y, además, podemos bloquear y desbloquear usuarios. Para ello, debemos tener, al menos, un usuario seleccionado y pulsar sobre el botón "Bloquear usuario".

Imagen 92: Acciones usuarios Hiss&Acclaim

Si el usuario estaba bloqueado, se desbloqueará y viceversa.

2.6 Menú

En esta categoría tenemos las tres opciones explicadas anteriormente de "Añadir", "Editar" y "Eliminar".

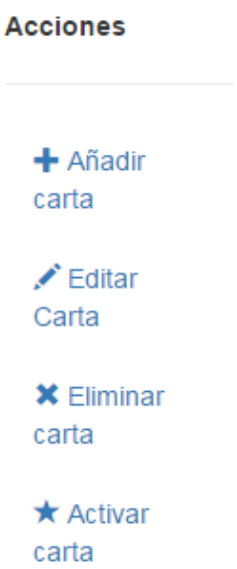


Imagen 93: Acciones Carta Menú Hiss&Acclaim

Además, tenemos la opción "Activar carta". Para ello, solo debemos tener una carta seleccionada. Nos aparecerá un mensaje de confirmación y pulsaremos "Sí" o "No" en función de nuestras necesidades. Solamente puede haber una carta activa (que es el primer elemento mostrado en el contenedor), por lo que si activas una nueva, la que estaba activada anteriormente se desactivará.

2.7 Galería

En este apartado solamente tenemos dos opciones “Añadir imagen” y “Eliminar imagen. La primera siempre estará visible, mientras que para poder acceder a la segunda debe haber sido seleccionada al menos una imagen en el contenedor. Sus procedimientos son análogos a los explicados en el apartado “Servicios”.

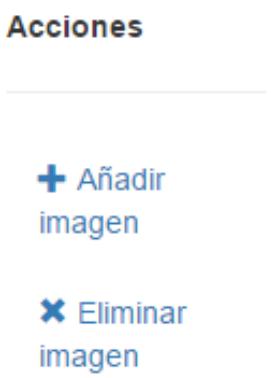


Imagen 94: Acciones Galería de fotos Hiss&Acclaim

Anexo II - Tablas de especificación de casos de uso desarrollados

Índice de tablas de casos de Uso

Sección	Actor Principal	ID. Caso de Uso
Actividades	Usuario	**1.** Ver lista actividades
		2. Ver información de una actividad
		3. Ver comentarios sobre una actividad concreta
		4. Valorar actividad
	Administrador	**5.** Ver lista actividades (administrador)
		6. Crear actividad
		7. Ver/Editar actividad
		8. Eliminar actividad
		9. Ver comentarios sobre una actividad concreta
		10. Valorar actividad
Restauración	Usuario	**11.** Ver carta
	Administrador	**12.** Activar carta.
		13. Ver listado de cartas
		14. Añadir Carta
		15. Eliminar carta
		16. Ver carta (administrador)
		17. Editar carta

Servicios	Usuario	**18.** Ver lista servicios
		19. Ver información de un servicio
		20. Ver comentarios sobre un servicio concreto
	Administrador	**21.** Ver lista de servicios (administrador)
		22. Crear servicio
		23. Ver/Editar servicio
		24. Eliminar servicio
		25. Ver comentarios sobre un servicio concreto
		26. Valorar servicio
Datos Cliente	Usuario	**27.** Ver datos cliente
		28. Modificar datos cliente
	Administrador	**29.** Ver lista de usuarios
		30. Ver datos cliente
		31. Modificar datos cliente
		32. Eliminar usuario
Social	Usuario	**33.** Ver timeline
		34. Crear comentario en el timeline
		35. Valorar un comentario del timeline

		36. Ver perfil propio
		37. Configurar perfil propio
		38. Buscar perfiles de otros usuarios
		39. Ver perfil de otro usuario
		40. Ver listado de comentarios de otro usuario
		41.Buscar mensajes en el timeline
		42. Ver lista de chats
		43. Iniciar/Ver chat con usuario
		44. Enviar mensaje a usuario
		45. Ver chat cliente-serv. habitación
		46. Escribir chat cliente-serv. habitación
	Administrador	**47.** Ver lista chats serv. habitación-clientes
		48. Ver chat serv. habitación-cliente
		49. Escribir chat serv. habitación-cliente
		50. Ver usuarios del sistema
		51. Bloquear/Desbloquear usuario del sistema
General	Usuario	**52.** Iniciar sesión
		53. Cerrar sesión.
		54. Ver información general del hotel.
		55. Ver galería de imágenes.

		56. Ver ubicación del hotel.
		57. Ver datos y opciones de la habitación.
		58. Modificar opciones de la habitación.
	Administrador	**59.** Añadir imagen a la galería
		60. Eliminar imagen de la galería.

Tablas de casos de uso

Nombre	Ver lista actividades	**ID**	1
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere ver todas las actividades disponibles.
Descripción:
El usuario podrá ver las actividades del hotel en modo lista. Tendrá a su disposición el nombre de la actividad, la fecha de realización, una breve descripción y la valoración de otros usuarios sobre la misma.

Si el usuario tiene conexión a internet, recibirá la última información actualizada. Si no, podrá visualizar los datos de la última vez que accedió a esa vista.
Trigger:
Pulsar sobre pestaña “Actividades”.
Precondición:
1. Estar autentificado mediante el usuario y contraseña.
Postcondición:
1. Aparecerá ventana con el listado de las actividades del hotel.
Flujo Normal:
1. Pinchar en la pestaña “Actividades”. 2. Aparece en pantalla el listado de actividades disponibles.
Flujo Alternativo:
Excepción:
1. Que no haya actividades listadas por el hotel.
Includes:
Requisitos especiales:
Notas:

Nombre	Ver información de una actividad	**ID**	2
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere desplegar la información de una actividad concreta.
Descripción:
El usuario podrá consultar la información de una manera más detallada sobre la actividad que desee.
Trigger:
Pulsar sobre la actividad que se quiere ver.
Precondición:
1. Estar autentificado mediante el usuario y contraseña que se proporciona en el momento de la llegada. 2. Haber accedido a la pestaña de actividades.
Postcondición:
1. Se despliega en la lista de actividades la información más concreta y detallada de la actividad seleccionada.

Flujo Normal:
1. Pulsar sobre la actividad de la que se quiere desplegar la información. 2. Sale por pantalla la información de dicha actividad desplegada.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver comentarios sobre una actividad concreta	ID	3
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:

Usuario
Personal involucrado o intereses:
Usuario: quiere ver los comentarios sobre una actividad concreta.
Descripción:
El usuario podrá ver un listado de comentarios en el timeline de otros usuarios sobre una actividad concreta para ver qué opina el resto de personas sobre dicha actividad.
Trigger:
Pulsar en una actividad concreta.
Precondición:
1. Estar autentificado mediante el usuario y contraseña. 2. Haber accedido a la pestaña de actividades.
Postcondición:
1. Ventana con el listado de comentarios de esa actividad.
Flujo Normal:
1. Pulsar sobre la actividad de la que se quiere ver el listado de comentarios. 2. Sale por pantalla dicho listado.
Flujo Alternativo:
Excepción:
1. Que no haya comentarios sobre la actividad.
Includes:

Requisitos especiales:
Notas:

Nombre	Valorar actividad	ID	4
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: El usuario quiere valorar una actividad.
Descripción:
El usuario puede valorar una actividad tanto positiva como negativamente. Para ello hay dos manos, una hacia arriba y otra hacia abajo. Sólo puede haber una valoración por actividad y por usuario, es decir, si un usuario ha votado ya positivamente y pulsa sobre el botón de votar negativamente, el primero de ambos será eliminado y solamente será contemplado este último. También se puede eliminar la valoración ya realizada pulsando sobre lo que ya se había valorado, lo que dejará dicha valoración nula.
Trigger:

Pulsar en el botón con la mano hacia arriba o hacia abajo.
Precondición:
1. Haber logueado.
Postcondición:
1. Sale la misma ventana con la valoración del usuario ya contemplada.
Flujo Normal:
1. Pinchar en la pestaña de actividades. 2. Pinchar sobre un botón de valoración.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver actividades (admin)	**ID**	5
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver la lista de actividades existentes.
Descripción:
El administrador puede ver el listado de actividades existentes.
Trigger:
Entrar a la pestaña de actividades.
Precondición:
1. Haber logueado como administrador.
Postcondición:
1. Sale una ventana con el listado de actividades.
Flujo Normal:
1. Pinchar en la pestaña de actividades.
Flujo Alternativo:

Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Crear actividad	ID	6
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere añadir una actividad a la lista de actividades disponibles.
Descripción:
Se creará una nueva actividad, habrá que seleccionar fecha, escribir nombre y datos de la actividad y guardar.

Trigger:
Pulsar el botón añadir actividad.
Precondición:
1. Estar logueado en el panel de control de administración. 2. Haber accedido a la categoría Actividades.
Postcondición:
1. Un listado de las actividades con la nueva ya añadida.
Flujo Normal:
1. Pinchar en el botón añadir actividad. 2. Rellenar los datos de la actividad (Nombre, descripción, etc.). 3. Confirmar la creación.
Flujo Alternativo:
Excepción:
1. Crear una actividad con el mismo nombre de otra ya existente. 2. Dejar sin rellenar algún campo obligatorio.
Includes:
Requisitos especiales:
Notas:

Nombre	Ver/Editar actividad	ID	7
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver o/y modificar los datos de una actividad ya existente.
Descripción:
El administrador puede ver y/o modificar los datos de una actividad en concreto guardarlos.
Trigger:
Pulsar el botón “Editar actividad”.
Precondición:
1. Estar logueado en el panel de control de administración. 2. Haber accedido a la categoría Actividades. 3. Seleccionar una y sólo una actividad en el panel.
Postcondición:
1. Se volverá a ver el listado de actividades.
Flujo Normal:
1. Pinchar en el botón editar actividad.

2. Modificar los datos necesarios. 3. Guardar los cambios realizados.
Flujo Alternativo:
1. Pinchar en el botón editar actividad. 2.Cerrar el modal.
Excepción:
1. Dejar sin rellenar un campo obligatorio.
Includes:
Requisitos especiales:
Notas:

Nombre	Eliminar actividad	**ID**	8
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Administrador

Personal involucrado o intereses:
Administrador: quiere eliminar una actividad del listado.
Descripción:
El administrador puede eliminar una o más de una actividad desde el panel de control de administración.
Trigger:
Pulsar el botón “Eliminar actividad”
Precondición:
1. Estar logueado en el panel de control de administración. 2. Haber accedido a la categoría Actividades. 3. Haber seleccionado, al menos, una actividad en el listado de actividades.
Postcondición:
1. Se habrán eliminado todas las actividades seleccionadas.
Flujo Normal:
1. Pinchar en el botón “Eliminar actividad”. 2. Pinchar sobre la confirmación del modal que aparece.
Flujo Alternativo:
Excepción:
Includes:

Requisitos especiales:
Notas:

Nombre	Ver comentarios sobre una actividad concreta	ID	9
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver los comentarios sobre una actividad concreta.
Descripción:
El usuario podrá ver un listado de comentarios en el timeline de otros usuarios sobre una actividad concreta para ver qué opina el resto de personas sobre dicha actividad.
Trigger:
Pulsar en una actividad concreta.
Precondición:

1. Estar autentificado mediante el usuario y contraseña. 2. Haber accedido a la pestaña de actividades.
Postcondición:
1. Ventana con el listado de comentarios de esa actividad.
Flujo Normal:
1. Pulsar sobre la actividad de la que se quiere ver el listado de comentarios. 2. Sale por pantalla dicho listado.
Flujo Alternativo:
Excepción:
1. Que no haya comentarios sobre la actividad.
Includes:
Requisitos especiales:
Notas:

Nombre	Valorar actividad	**ID**	10
Creado por		**Fecha**	

Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: El administrador quiere valorar una actividad.
Descripción:
El administrador puede valorar una actividad tanto positiva como negativamente. Para ello hay dos manos, una hacia arriba y otra hacia abajo. Sólo puede haber una valoración por actividad y por usuario, es decir, si un usuario ha votado ya positivamente y pulsa sobre el botón de votar negativamente, el primero de ambos será eliminado y solamente será contemplado este último. También se puede eliminar la valoración ya realizada pulsando sobre lo que ya se había valorado, lo que dejará dicha valoración nula.
Trigger:
Pulsar en el botón con la mano hacia arriba o hacia abajo.
Precondición:
1. Haber logueado como administrador.
Postcondición:
1. Sale la misma ventana con la valoración del administrador ya contemplada.
Flujo Normal:
1. Pinchar en la pestaña de actividades. 2. Pinchar sobre un botón de valoración.

Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver carta	ID	11
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere ver la carta del restaurante.

Descripción:
El usuario desea ver la carta del restaurante para saber platos y precios disponibles.
Trigger:
Pulsar el botón “Ver carta”.
Precondición:
1. Haber logueado como usuario del sistema. 2. Haber accedido a la pestaña Hotel.
Postcondición:
1. Aparecerá la carta o un mensaje notificando su inexistencia en caso de no existir.
Flujo Normal:
1. Pulsar el botón “ver carta”.
Flujo Alternativo:
Excepción:
1. Que no haya ninguna carta activa.
Includes:
Requisitos especiales:
Notas:

Nombre	Activar carta	**ID**	12
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere modificar la carta actual del restaurante.
Descripción:
En el hotel solamente habrá una carta disponible pero, dentro del panel de control, pueden haber guardadas muchas cartas (carta de verano, carta de fin de semana, etc.). El administrador podrá seleccionar cuál es la carta activa en ese momento.
Trigger:
Pulsar el botón “Activar carta”.
Precondición:
1. Haber logueado como administrador en el panel de administración del hotel. 2. Haber accedido a la categoría “Menú”.

Postcondición:
1. Aparecerá el listado de cartas en el panel.
Flujo Normal:
1. Pulsar el botón “Activar carta”.
Flujo Alternativo:
Excepción:
1. Que no haya ninguna carta.
Includes:
Requisitos especiales:
Notas:

Nombre	Ver listado de cartas	**ID**	13
Creado por		**Fecha**	
Modif. por		**Fecha**	

		Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:
Administrador: quiere ver el listado de cartas actuales en el sistema.
Descripción:
El administrador quiere el listado de cartas que hay actualmente en el sistema.
Trigger:
Pulsar la categoría “Menú”.
Precondición:
1.Haber logueado como administrador en el panel de control de administración.
Postcondición:
1. Aparecerá un listado con todas las cartas existentes en el sistema.
Flujo Normal:
1.Pulsar la categoría “Menú”.
Flujo Alternativo:
Excepción:
Includes:

Requisitos especiales:
Notas:

Nombre	Añadir carta	ID	14
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere añadir una carta al listado de cartas del restaurante del hotel.
Descripción:
El administrador tiene varias cartas en el sistema y puede añadir una más.
Trigger:
Pulsar el botón “Añadir carta”.
Precondición:

1. Haber logueado como administrador en el panel de administración. 2. Haber accedido a la categoría “Menú”.
Postcondición:
1. La carta quedará añadida al listado de cartas del sistema.
Flujo Normal:
1. Se pincha sobre la categoría “Menú”. 2. Se pincha sobre “Añadir carta”. 3. Se rellenan los campos. 4. Se confirma el formulario.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Eliminar carta	ID	15
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:
Administrador: quiere eliminar una o más cartas del sistema.
Descripción:
El administrador puede eliminar una o más cartas del sistema.
Trigger:
Pulsar el botón “Eliminar carta”.
Precondición:
1. Haber logueado como administrador en el panel de administrador. 2. Haber accedido a la categoría de restauración. 3. Haber seleccionado al menos una carta del sistema.
Postcondición:
1. Aparecerá un listado de cartas del restaurante del hotel con las restantes.
Flujo Normal:
1.Pulsar el botón “Eliminar carta”. 2.Confirmar la eliminación.

Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver carta	ID	16
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:

Administrador: quiere ver la carta del restaurante.
Descripción:
El administrador verá la carta actual del restaurante que esté activa en ese momento.
Trigger:
Pulsar el botón “Ver carta”.
Precondición:
1. Haber logueado como administrador. 2. Haber accedido a la pestaña de Hotel..
Postcondición:
1. Aparecerá la carta o un mensaje notificando su inexistencia en caso de que así sea.
Flujo Normal:
1. Pulsar el botón “ver carta”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Editar	ID	17
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:
Administrador: desea cambiar los datos de una carta del restaurante.
Descripción:
Permite al administrador modificar la carta seleccionada.
Trigger:
Pulsar el botón “Editar carta”.
Precondición:
1. Estar logueado en el sistema como administrador en el panel de administración. 2. Haber accedido a la pestaña Restauración. 3. Haber seleccionado una y sólo carta del sistema.
Postcondición:

1. Aparece el listado de cartas con la seleccionada ya modificada.
Flujo Normal:
1. Pulsar el botón “Editar carta”. 2. Editar los datos del formulario. 3. Confirmar los cambios.
Flujo Alternativo:
Excepción:
1. Que no haya cartas en el sistema.
Includes:
Requisitos especiales:
Notas:

Nombre	Ver servicios	**ID**	18
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere ver los servicios que puede contratar.
Descripción:
Cuando un cliente llega a un hotel desea ver los servicios que puede contratar. Se le dará la opción de verlos mediante un listado.
Trigger:
Acceder a la pestaña “Servicios”.
Precondición:
1. Haber logueado como usuario.
Postcondición:
1. Se abrirá un listado con todos los servicios (pestaña servicios).
Flujo Normal:
1. Picar sobre la pestaña “Servicios”.
Flujo Alternativo:
Excepción:
Includes:

Requisitos especiales:
Notas:

Nombre	Ver información de un servicio	ID	19
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere ver la información de un servicio concreto de la lista.
Descripción:
Un cliente desea ampliar la información sobre un servicio que aparece en la lista de servicios.
Trigger:
Pulsar sobre un servicio del listado.
Precondición:
1. Estar logueado como usuario.

2. Haber accedido a la pestaña “Servicios”.
Postcondición:
1. Se desplegará la información del servicio seleccionado.
Flujo Normal:
1. Picar sobre un servicio del listado de servicios.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver comentarios sobre un servicio concreto	**ID**	20
Creado por		**Fecha**	
Modif. por		**Fecha**	

		Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere ver los comentarios de otros usuarios sobre un servicio.
Descripción:
El usuario podrá ver todos los comentarios de otros usuarios sobre un servicio concreto para saber opiniones externas.
Trigger:
Pulsar sobre un servicio concreto dentro del listado de servicios.
Precondición:
1. Estar logueado como usuario. 2. Haber accedido a la pestaña de servicios
Postcondición:
1.Saldrá una vista con los comentarios del servicio.
Flujo Normal:
1. Se pincha sobre el servicio del cual se quieren ver los comentarios..
Flujo Alternativo:
Excepción:

Includes:
Requisitos especiales:
Notas:
Un servicio contratado pasa a aparecer como contratado en el listado de servicios.

Nombre	Ver servicios (admin)	ID	21
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: desea ver un listado de servicios.
Descripción:
El administrador puede ver un listado de servicios.
Trigger:
Picar en la pestaña servicios.

Precondición:
1. Haber logueado como administrador.
Postcondición:
1. Aparecerá la lista de servicios disponibles.
Flujo Normal:
1. Pulsar sobre la pestaña servicios.
Flujo Alternativo:
Excepción:
1. Si no existe ningún servicio aparecerá un mensaje notificándolo.
Includes:
Requisitos especiales:
Notas:

Nombre	Crear servicio	**ID**	22
Creado por		**Fecha**	

Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere añadir un servicio a la lista de servicios disponibles del hotel.
Descripción:
Permite añadir un nuevo servicio, rellenar sus datos (ubicación, horarios, nombre, etc…) y guardar.
Trigger:
Pulsar el botón “Añadir servicio”
Precondición:
1. Estar logueado en el sistema como administrador en el panel de control. 2. Haber accedido a la categoría Servicios.
Postcondición:
1. Aparece un listado de los servicios creados.
Flujo Normal:
1. Pulsar el botón “Añadir servicio”. 2. Rellenar los datos del servicio. 3. Guardar.
Flujo Alternativo:

Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver/Editar servicio	ID	23
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: desea ver o/y modificar los datos de un servicio ya creado.
Descripción:

Permite ver y/o modificar los datos de un servicio del listado.
Trigger:
Pulsar el botón “Editar servicio”.
Precondición:
1. Estar logueado en el sistema como administrador en el panel de administración. 2. Haber accedido a la pestaña Servicios. 3. Haber seleccionado uno y sólo uno de los servicios.
Postcondición:
1. En caso de que se realice una modificación aparecerá una ventana de confirmación de los cambios durante unos segundos.
Flujo Normal:
1. Pulsar el botón “Editar servicio”. 2. Ver y Modificar los datos necesarios. 3. Guardar cambios.
Flujo Alternativo:
Excepción:
1. Dejar algún campo obligatorio sin rellenar al hacer una modificación.
Includes:
Requisitos especiales:

Notas:

Nombre	Eliminar servicio	ID	24
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: desea eliminar uno o más servicios ya existentes.
Descripción:
Permite eliminar uno o más servicios servicios en el sistema.
Trigger:
Pulsar el botón “Eliminar servicio”
Precondición:
1. Estar logueado en el sistema como administrador en el panel de administración. 2. Haber accedido a la pestaña Servicios. 3. Haber seleccionado, al menos, un servicio en el listado.

Postcondición:
1. Aparece el listado de servicios disponibles después de la eliminación de los seleccionados.
Flujo Normal:
1. Pulsar el botón “Eliminar servicio” 2. Sale una ventana de confirmación de la eliminación. 3. Aceptar o declinar la confirmación.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver comentarios sobre un servicio seleccionado	**ID**	25
Creado por		**Fecha**	

Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver los comentarios sobre un servicio seleccionado.
Descripción:
El administrador podrá ver los comentarios de otros usuarios sobre un servicio concreto.
Trigger:
Pulsar sobre un servicio dentro del listado de servicios.
Precondición:
1. Haber logueado como administrador. 2. Haber accedido a la pestaña “Servicios”.
Postcondición:
1. Aparecerá una lista con los mensajes sobre el servicio pinchado.
Flujo Normal:
1. Pulsar sobre un servicio concreto.
Flujo Alternativo:
Excepción:

Includes:
Requisitos especiales:
Notas:

Nombre	Valorar servicio	ID	26
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere valorar un servicio concreto.
Descripción:
Un administrador puede valorar un servicio en concreto tanto positiva como negativamente.
Trigger:
Pulsar el botón correspondiente de mano hacia arriba o mano hacia abajo.

Precondición:
1. Haber logueado como administrador. 2. Haber accedido a la pestaña servicios.
Postcondición:
1. Aparecerá la misma vista general de servicios pero con la valoración ya contemplada.
Flujo Normal:
1. Pulsar el botón mano arriba o mano abajo correspondiente a la valoración.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver datos cliente	ID	27

Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: desea consultar sus datos personales, opciones de cuenta, etc.
Descripción:
El usuario podrá consultar en cualquier momento su información personal y sus opciones.
Trigger:
Pulsar la pestaña “Mi habitación”
Precondición:
1. Haber logueado como usuario. 2. Haber accedido a la pestaña “Ajustes”.
Postcondición:
1. Aparecen en pantalla todas las informaciones disponibles una bajo otra en apartados.
Flujo Normal:
1. Pulsar en la pestaña “Mi habitación”. 2. Se carga la información en pantalla.
Flujo Alternativo:

Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Modificar datos cliente	ID	28
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere modificar sus datos.
Descripción:

El usuario podrá en cualquier momento editar sus datos y opciones.
Trigger:
Pulsar el botón “Mi habitación”.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido a la pestaña “Mi habitación”. 3. Tener conexión a internet.
Postcondición:
1. Saldrá una ventana con los datos y opciones del usuario modificados.
Flujo Normal:
1.Pulsar el botón “Mi habitación”. 2.Modificar los datos pertinentes.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver lista de usuarios	ID	29
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver la lista de usuarios del sistema.
Descripción:
El administrador puede ver un listado de clientes.
Trigger:
Pulsar la categoría “Usuarios”.
Precondición:
1. Haber logueado como administrador en el panel de control de administración.
Postcondición:
1. Aparecerá una ventana con la lista de clientes.
Flujo Normal:
1.Pulsar la categoría “Usuarios”.

Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver datos cliente	ID	30
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver los datos de un cliente.

Descripción:
Permite al administrador ver todos los datos de un usuario concreto.
Trigger:
Pulsar el botón “Editar Usuario”.
Precondición:
1. Haber logueado como administrador al panel de control de administración del sistema. 2. Haber entrado en la categoría “Usuarios”.
Postcondición:
1. Una ventana con los datos del usuario.
Flujo Normal:
1. Pulsar la categoría “Usuarios”. 2. Seleccionar un usuario.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:

Notas:

Nombre	Modificar datos cliente	ID	31
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrado: quiere editar los datos del cliente.
Descripción:
Permite modificar los datos del cliente.
Trigger:
Pulsar el botón “Ver/editar cliente”.
Precondición:
1. Haber logueado como administrador. 2. Haber accedido a la categoría “Usuarios”. 3. Haber seleccionado uno y sólo un usuario.

Postcondición:
1. Aparecerá una ventana durante unos segundos confirmando la modificación de los datos (si ha habido modificación).
Flujo Normal:
1. Pulsar el botón “Editar usuario”. 2. Modificar los datos del cliente. 3. Confirmar la modificación.
Flujo Alternativo:
Excepción:
1. Dejar sin rellenar datos datos obligatorios.
Includes:
Requisitos especiales:
Notas:

Nombre	Eliminar usuario	**ID**	32
Creado por		**Fecha**	

Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere eliminar la cuenta y todos los datos del cliente.
Descripción:
Permite eliminar la cuenta del cliente totalmente.
Trigger:
Pulsar el botón “Eliminar usuario”.
Precondición:
1. Haber logueado como administrador en el panel de administración del usuario. 2. Haber entrado en la categoría “Usuarios”. 3. Haber seleccionado al menos un usuario.
Postcondición:
1. Quedan eliminados absolutamente todos los datos de la cuenta de los clientes seleccionados.
Flujo Normal:
1. Pulsar el botón “Eliminar”. 2. Se abre una ventana de confirmación en la que se tiene que marcar que se pretenden eliminar todos los datos de la cuenta. 3. Aceptar o declinar la confirmación.

Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver timeline	ID	33
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: desea ver el timeline.

Descripción:
En nuestra aplicación habrá un timeline en el que los usuarios hospedados en el hotel irán comentando y opinando sobre el servicio o los eventos. Todos los usuarios con acceso a la aplicación podrán verlo.
Trigger:
Acceder a la pestaña “Timeline”.
Precondición:
1. Haber logueado como usuario.
Postcondición:
1. Aparecerá el timeline en pantalla.
Flujo Normal:
1. Acceder a la pestaña “Timeline”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Crear comentario en el timeline	ID	34
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: desea dejar un comentario en el timeline.
Descripción:
El usuario podrá dejar un comentario en el timeline con un límite de caracteres, categoría y valoración (Hiss o Acclaim).
Trigger:
Pulsar el botón “Comentar” con símbolo de un lápiz presente en la parte superior del Timeline.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido a la pestaña “Timeline”.
Postcondición:
1. Saldrá el timeline con el comentario añadido.

Flujo Normal:
1. Pulsar el botón “Comentar”. 2. Se abrirá una ventana para redactar el mensaje con sus opciones. 3. Redactar. 4. Confirmar.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Valorar comentario en el timeline	ID	35
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: quiere valorar un comentario del timeline publicado anteriormente.
Descripción:
El usuario podrá valorar tanto positiva como negativamente un comentario del timeline.
Trigger:
Pulsar el botón mano arriba o mano abajo en el comentario correspondiente.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido al timeline.
Postcondición:
1. Saldrá el timeline con el comentario ya modificado.
Flujo Normal:
1. Pulsar el botón mano arriba o mano abajo.
Flujo Alternativo:
Excepción:
Includes:

Requisitos especiales:
Notas:

Nombre	Ver perfil propio	ID	36
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: desea ver su propio perfil.
Descripción:
El usuario podrá acceder a su perfil para verlo cómo lo verían los demás usuarios.
Trigger:
Pulsar el botón “Mi perfil” en la pestaña Ajustes.
Precondición:
1. Haber logueado como usuario.

2. Haber accedido a la pestaña “Ajustes”.
Postcondición:
1. Aparecerá una ventana con el perfil de la misma forma que lo ve el resto de usuarios.
Flujo Normal:
1. Pulsar el botón “Mi perfil”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Configurar perfil propio	**ID**	37
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: desea configurar su perfil social.
Descripción:
El usuario podrá personalizar su perfil. Desde imagen de perfil, nick, estado, etc.
Trigger:
Pulsar la categoría “Mi perfil”, presente en la pestaña Ajustes.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido a la pestaña Ajustes. 3. Haber accedido a su propio perfil.
Postcondición:
1. Aparecerá el perfil del usuario con los cambios realizados.
Flujo Normal:
1. Pulsar el botón “Mi perfil”. 2. Editar los apartados que se deseen editar (Imagen, Nick, Estado, etc.). 3. Confirmar la edición.
Flujo Alternativo:
Excepción:
1. Que se deje en blanco algún apartado obligatorio.

Includes:
Requisitos especiales:
Notas:

Nombre	Buscar perfiles de otros usuarios	ID	38
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: Desea buscar el perfil de otro usuario.
Descripción:
Un usuario podrá buscar el perfil de otro usuario filtrando por nombre y apellidos.
Trigger:
Pulsar el botón “Buscar usuario” presente en la parte superior del Timeline.

Precondición:
1. Haber logueado como usuario. 2. Haber accedido a la pestaña Timeline.
Postcondición:
1. Saldrá en pantalla una ventana con un conjunto de usuarios que concuerden con los criterios de búsqueda escogidos.
Flujo Normal:
1. Pulsar el botón “Buscar usuario”. 2. Introducir los criterios de búsqueda. 3. Pulsar el botón “Buscar”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver perfil de otro usuario	**ID**	39
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Usuario
Personal involucrado o intereses:
1. El usuario desea ver el perfil de otro usuario.
Descripción:
El usuario podrá ver el perfil de un usuario que ha buscado o ha visto en el timeline.
Trigger:
Pulsar la foto de perfil del usuario o un comentario suyo.
Precondición:
1. Haber logueado como usuario.
Postcondición:
1. Saldrá en una ventana el perfil de la persona indicada.
Flujo Normal:
1. Pulsar la foto de perfil del usuario o su nombre.
Flujo Alternativo:

Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver listado de comentarios de otro usuario	ID	40
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: quiere ver el listado de comentarios de un usuario concreto.
Descripción:
El usuario podrá ver un listado de los comentarios de una persona en concreto.

Trigger:
Pulsar sobre un comentario cualquiera de ese usuario.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido al perfil de otro usuario.
Postcondición:
1. Saldrá un listado con los mensajes del usuario deseado..
Flujo Normal:
1. Pulsar sobre un usuario.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Buscar mensajes del timeline	ID	41
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: puede buscar mensajes del timeline.
Descripción:
El usuario puede realizar una búsqueda por contenido en el listado de mensajes del timeline.
Trigger:
Pulsar el botón buscar.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido al timeline.
Postcondición:
1. Aparecerá un listado con todos los comentarios sobre el contenido buscado.
Flujo Normal:
1. Pulsar el botón buscar en el timeline. 2. Insertar fragmento de mensaje a buscar. 3. Pinchar en buscar.

Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver lista de chats	ID	42
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: desea ver la lista de chats abiertos con anterioridad con otros usuarios.

Descripción:
El usuario podrá ver chats anteriores que ha tenido con otros usuarios en cualquier momento.
Trigger:
Pulsar sobre la pestaña “Chats”.
Precondición:
1. Haber logueado como usuario.
Postcondición:
1. Aparecerá en pantalla una lista de chats abiertos con anterioridad.
Flujo Normal:
1. Pulsar sobre la pestaña Chats.
Flujo Alternativo:
Excepción:
1. Que no haya chats abiertos aún en cuyo caso aparecerá una lista vacía con el texto “no hay conversaciones abiertas”.
Includes:
Requisitos especiales:
Notas:

Nombre	Iniciar/Ver chat con usuario	ID	43
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: quiere poder iniciar o continuar una conversación con un usuario concreto.
Descripción:
El usuario inicia una conversación privada con otro usuario desde el perfil de ese usuario o continua una conversación abierta desde la pestaña “Mensajes”.
Trigger:
Pulsar el botón “Iniciar conversación” o pulsar sobre una conversación de la lista que aparece en la pestaña “Mensajes”.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido al perfil de un usuario para iniciar una nueva conversación o haber accedido a la pestaña “Mensajes” para continuar una conversación abierta con anterioridad.
Postcondición:
1. Aparece una ventana con la conversación (con su historial si existe) lista para enviar o

recibir mensajes.
Flujo Normal:
1. Picar en el botón Iniciar chat del perfil o sobre una conversación de la pestaña “Mensajes”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Enviar mensaje a usuario	ID	44
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:

Usuario
Personal involucrado o intereses:
Usuario: quiere enviar un mensaje dentro de una conversación.
Descripción:
Un usuario puede enviar un mensaje a otro usuario.
Trigger:
Pulsar sobre la casilla en blanco para escribir el mensaje pertinente en la conversación.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido a una conversación.
Postcondición:
1. Aparecerá el nuevo mensaje en el historial de la conversación.
Flujo Normal:
1. Pulsar sobre la casilla en blanco. 2. Escribir mensaje deseado. 3. Pulsar sobre enviar.
Flujo Alternativo:
Excepción:
1. Que no haya conexión a internet actualmente, lo cual se notificará.
Includes:

Requisitos especiales:
Notas:

Nombre	Ver chat cliente-servicio habitación	ID	45
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: quiere ver el chat con el servicio de habitaciones.
Descripción:
El usuario podrá ver un chat con el servicio de habitaciones en cualquier momento.
Trigger:
Pulsar la pestaña en la parte superior de “Room service”.
Precondición:
1. Haber logueado como usuario.

Postcondición:
1. Se abrirá una ventana en la que se verá el chat con el servicio de habitaciones.
Flujo Normal:
1. Pulsar la pestaña “Chat con el servicio de habitaciones”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Escribir chat cliente-servicio habitación	**ID**	46
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Usuario
Personal involucrado o intereses:
Usuario: quiere mandar un mensaje al servicio de habitaciones.
Descripción:
El usuario escribirá un mensaje al servicio de habitaciones en el chat privado.
Trigger:
Pulsar la caja de texto del chat.
Precondición:
1. Haber logueado como usuario. 2. Haber accedido a la pestaña del chat privado con el servicio de habitaciones.
Postcondición:
1. Aparecerá el mensaje en el historial de mensajes con el servicio de habitaciones.
Flujo Normal:
1. Picar en la caja blanca de introducción de texto. 2. Escribir el mensaje. 3. Pulsar enviar.
Flujo Alternativo:
Excepción:
1. Que no haya conexión a internet en este momento.
Includes:

Requisitos especiales:
Notas:

Nombre	Ver lista chats serv. habitación-clientes	ID	47
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere ver los chats que tiene el servicio de habitaciones con los usuarios.
Descripción:
El administrador tendrá acceso a todos los chats que tiene el servicio de habitaciones con los usuarios.
Trigger:
Pulsar la pestaña “Chats servicio de habitaciones”.

Precondición:
1. Haber logueado como administrador.
Postcondición:
1. Aparecerá el listado de chats.
Flujo Normal:
1. Pulsar la pestaña “chats servicio de habitaciones”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver chat servicio habitación-cliente	**ID**	48
Creado por		**Fecha**	
Modif. por		**Fecha**	

		Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:
Administrador: quiere ver un chat concreto con un cliente.
Descripción:
El administrador podrá acceder a un chat concreto con un usuario.
Trigger:
Pulsar un chat dentro del listado de chats de usuarios con el servicio de habitaciones.
Precondición:
1. Haber logueado como administrador. 2. Haber accedido al listado de chats servicio de habitaciones.
Postcondición:
1. Aparecerá una ventana con el chat histórico del usuario.
Flujo Normal:
1. Pulsar un chat dentro del listado de chats.
Flujo Alternativo:
Excepción:

Includes:
Requisitos especiales:
Notas:

Nombre	Escribir chat servicio habitación-cliente	ID	49
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:
Administrador: quiere enviar un mensaje a un usuario dentro de un chat del servicio de habitaciones.
Descripción:
El administrador puede responder cuestiones planteadas por los clientes al servicio de habitaciones.
Trigger:

Pulsar sobre la caja blanca de introducción de texto.
Precondición:
1. Haber logueado como administrador. 2. Haber accedido a la pestaña de chats del servicio de habitaciones. 3. Haber accedido a un chat concreto con un cliente.
Postcondición:
1. Aparecerá el mensaje enviado en el histórico de mensajes con el cliente.
Flujo Normal:
1. Pulsar sobre la caja blanca de introducción de texto. 2. Escribir respuesta. 3. Pulsar botón enviar.
Flujo Alternativo:
Excepción:
1. QUe no haya conexión a internet.
Includes:
Requisitos especiales:
Notas:

Nombre	Ver usuarios del sistema	ID	50
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:
Administrador: quiere ver los usuarios del sistema.
Descripción:
El administrador podrá, en cualquier momento, ver el listado de usuarios del sistema.
Trigger:
Pulsar la pestaña “Usuarios”.
Precondición:
1. Haber logueado como administrador en el panel de control de administración.
Postcondición:
1. Saldrá un listado con todos los usuarios registrados en el presente y pasado.
Flujo Normal:
1. Pulsar la categoría “Usuarios”.
Flujo Alternativo:

Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Bloquear/Desbloquear usuarios del sistema	ID	51
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador.
Personal involucrado o intereses:
Administrador: quiere bloquear/desbloquear a un usuario del sistema.
Descripción:
Un administrador puede, en cualquier momento, bloquear a un usuario del sistema, ya que éste puede estar causando problemas o incumpliendo normas. También podrá desbloquearlo

en caso de que haya errores o se corrija la actitud conflictiva del usuario. El bloqueo del usuario supone que este usuario no puede realizar ninguna actividad en la red social ni siquiera acceder a la misma.
Trigger:
Pulsar el botón “Bloquear usuario” en el panel de control de administración.
Precondición:
1. Haber logueado como administrador. 2. Haber accedido a la pestaña “Clientes”.
Postcondición:
1. Aparecerá un mensaje durante unos segundos de confirmación de bloqueo/desbloqueo.
Flujo Normal:
1. Pulsar el botón “Bloquear/Desbloquear usuario”. 2. Saldrá una ventana de confirmación. 3. Confirmar.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:

Notas:

Nombre	Iniciar sesión	ID	52
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
Usuario: quiere acceder a la aplicación.
Descripción:
El usuario quiere acceder a la aplicación. Para ello debe poner su nombre de usuario y su contraseña.
Trigger:
Pulsar el botón “Login”.
Precondición:
1. Haber iniciado la aplicación
Postcondición:
1. Aparecerá el timeline del hotel.

Flujo Normal:
1. Rellenar el formulario indicando el nombre de usuario y la contraseña. 2. Pulsar sobre el botón”Login”.
Flujo Alternativo:
Excepción:
1. Que no haya conexión a internet.
Includes:
Requisitos especiales:
Notas:

Nombre	Cerrar sesión	ID	53
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario

Personal involucrado o intereses:
El usuario desea cerrar su sesión de la aplicación.
Descripción:
El usuario desea cerrar la sesión y puede hacerlo desde la pestaña “Ajustes”.
Trigger:
Pulsar en “Logout”.
Precondición:
1.Tener conexión a internet.
Postcondición:
1. Aparecerá la vista de iniciar sesión.
Flujo Normal:
1. Ir a la pestaña “Ajustes”. 2. Pinchar en “Logout”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:

Notas:

Nombre	Ver información general del hotel	ID	54
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario.
Personal involucrado o intereses:
1. El usuario desea ver información general del hotel.
Descripción:
En la aplicación hay un apartado para ver información general del hotel que será accesible tanto con internet como en la versión offline.
Trigger:
Pulsar el botón “El hotel” dentro de la pestaña “Hotel”.
Precondición:
1. Haber accedido a la pestaña “Hotel”.
Postcondición:

1. Una vista con la información general del hotel.
Flujo Normal:
1. Ir a la pestaña “Hotel”. 2. Pinchar en el botón “El hotel”.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver galería de imágenes	ID	55
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
1. El usuario puede ver una galería de imágenes del hotel.
Descripción:
El hotel cuenta con una galería de imágenes. El usuario podrá ver las imágenes de la misma.
Trigger:
Pulsar sobre el botón “Galería” en el apartado “Hotel”.
Precondición:
1. Haber accedido a la pestaña “Hotel”.
Postcondición:
1. Una vista de la galería de imágenes.
Flujo Normal:
1. Haber accedido a la pestaña “Hotel”. 2. Pinchar sobre galería.
Flujo Alternativo:
Excepción:
Includes:

Requisitos especiales:
Notas:

Nombre	Ver ubicación del hotel	ID	56
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
1. Usuario: desea ver la ubicación del hotel.
Descripción:
Hay dos modalidades para este caso de uso: online y offline. En la versión online, se abrirá una nueva vista con un mapa de “Google Maps” donde se puede navegar. En caso de no tener conexión a internet, se mostratá una imagen estática de la posición del hotel.
Trigger:
Pulsar en “Ubicación”.
Precondición:

1. Haber accedido a la pestaña “Hotel”.
Postcondición:
1. Una vista con el mapa con la ubicación del hotel. 2.
Flujo Normal:
1. Pinchar sobre Hotel. 2. Pinchar sobre ubicación.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Ver datos y opciones de la habitación.	**ID**	57
Creado por		**Fecha**	

Modif. por		Fecha Modif.	

Actor principal:
Usuario
Personal involucrado o intereses:
1. Usuario: puede ver los datos de su habitación.
Descripción:
Hay varias opciones dentro de la habitación, como por ejemplo, que se limpie la habitación o no. Éstas serán activables desde este menú.
Trigger:
Pulsar el botón “Mi habitación.”
Precondición:
1.Haber accedido a “Ajustes”.
Postcondición:
1. Aparece una vista con los datos actuales de la habitación.
Flujo Normal:
1. Acceder a “Ajustes”. 2. Pinchar en “Mi habitación”.
Flujo Alternativo:
Excepción:

Includes:
Requisitos especiales:
Notas:

Nombre	Modificar opciones de la habitación.	ID	58
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Usuarios
Personal involucrado o intereses:
1. Usuario: desea modificar los datos de su habitación.
Descripción:
Hay varias opciones dentro de la habitación, como por ejemplo, que se limpie la habitación o no. Éstas serán modificables desde este menú.

Trigger:
Pulsar sobre cada opción que se desea modificar dentro de este menú.
Precondición:
1.Haber accedido a “Mi habitación”.
Postcondición:
1. La misma vista de las opciones de la habitación pero con los datos ya cambiados.
Flujo Normal:
1. Acceder a “Ajustes”. 2. Acceder a “Mi habitación”. 3. Modificar los datos necesarios (serán guardados).
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Añadir imagen a la galería	**ID**	59
Creado por		**Fecha**	
Modif. por		**Fecha Modif.**	

Actor principal:
Administrador
Personal involucrado o intereses:
1. Administrador: quiere añadir una imagen a la galería.
Descripción:
El administrador puede añadir una imagen a la galería de imágenes del hotel.
Trigger:
Pulsar el botón “Añadir imagen” dentro del panel de administración.
Precondición:
1. Haber accedido al panel de administración. 2. Haber pinchado en la categoría Galería.
Postcondición:
1. Una vista de las imágenes de la galería.
Flujo Normal:
1. Acceder al panel de control. 2. Pinchar sobre “Galería”. 3. Pinchar sobre “Añadir imagen”. 4. Elegir la imagen y darle un nombre.

5. Confirmar.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:
Notas:

Nombre	Eliminar imagen de la galería.	ID	60
Creado por		Fecha	
Modif. por		Fecha Modif.	

Actor principal:
Administrador
Personal involucrado o intereses:

1. El administrador quiere eliminar una o más imágenes de la galería.
Descripción:
El administrador tiene la capacidad de eliminar las imágenes que quiera de la galería.
Trigger:
Pinchar sobre el botón “Eliminar imagen.”
Precondición:
1. Haber accedido al panel de administración. 2. Haber accedido a la categoría “Galería”. 3. Haber seleccionado al menos una imagen de la galería.
Postcondición:
1. Una vista de las imágenes de la galería después de borrarlas.
Flujo Normal:
1. Seleccionar el conjunto de imágenes a eliminar. 2. Pinchar en eliminar imagen. 3. Confirmar.
Flujo Alternativo:
Excepción:
Includes:
Requisitos especiales:

Notas: