Universitat de les Illes Balears

escola politècnica superior

Trabajo de Fin de Grado

Grado en Ingeniería Informática

SSTTApp: Gestión de Incidencias Hoteleras

Carlos Tous Ferragut

Tutor / Supervisor
Miquel Mascaró Portells

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 10 de Septiembre de 2018

Gracias a todos los profesores que me he encontrado a lo largo de mis estudios por haberme enseñado de la mejor manera sus conocimientos y por conseguir que haya disfrutado realizando este trabajo de fin de grado.
Agradecer también a mi familia, por creer en mí y apoyarme en los momentos más difíciles en este largo recorrido.

Índice General

Índice de imágenes

Acrónimos

- EPS: Escola Politècnica Superior.
- TFG: Trabajo de fin de grado.
- SSTTApp: Nombre comercial de la plataforma desarrollada por el alumno.
- IDE: *Integrated Development Environment*. Entorno de desarrollo que facilita la programación al desarrollador.
- AWS: *Amazon Web Services*. Colección de servicios de computación en la nube pública.
- TIC: Tecnologías de información y comunicación.
- Check-out: Proceso de salida de clientes de un hotel.
- TPV: Terminal punto de venta.
- API: *Application Programming Interface*. Interfaz de programación de aplicaciones.
- SSTT: Departamento de servicios técnicos de un hotel.
- IEEE: *Institute of Electrical and Electronics Engineers*. Instituto de Ingeniería Electrónica y Eléctrica.
- RF: Requisito Funcional.
- RNF: Requisito no funcional.
- REST: *Representational state transfer*. Estilo de arquitectura software para sistemas distribuidos como la World Wide Web.
- HTTP: *Hypertext Transfer Protocol*. Protocolo de comunicación para la World Wide Web.
- Amazon EC2: *Amazon Elastic Compute Cloud*. Servicio de AWS que permite alquilar servidores virtuales.
- MPMs: *Multi-Processing Modules*. Módulos encargados de conectar con los puertos de red de un servidor, aceptar las peticiones, y generar los procesos hijo que se encargan de servirlas.
- CSS: *Cascading Style Sheets*. Hojas de estilo en cascada usadas para dar formato a las páginas web.
- DOM: *Document Object Model*. Interfaz de programación para representar documentos HTML, XHTML y XML
- AJAX: *Asynchronous JavaScript And XML*. Técnica que permite la transferencia asíncrona de datos entre un navegador y un servidor.

- JSON: *JavaScript Object Notation*. formato de texto ligero para el intercambio de datos.
- XML: eXtensible Markup Language. Meta-lenguaje utilizado para el almacenamiento y la comunicación de datos.
- SDK: *Software Development Kit*. Conjunto de herramientas de desarrollo de software.
- URL: *Uniform Resource Locator*. Identificador de recursos como por ejemplo una página web.

Resumen

En este trabajo de fin de grado (TFG) se implementa una aplicación para llevar a cabo la gestión integral de las incidencias producidas en una cadena hotelera. La aplicación se apoya en dos plataformas fundamentales: Android y una página web para la gestión y creación de las incidencias.

SSTTApp pretende facilitar la comunicación entre los empleados de la cadena, así como gestionar de manera eficiente todas las incidencias que se van produciendo en el día a día. Además, proporciona un resumen de los datos más significativos para los directores, con el fin de mejorar y aumentar la productividad de sus empleados.

Mediante la página web, los recepcionistas tienen a su disposición todas las incidencias que se han ido produciendo y tienen la posibilidad de crear nuevas para los encargados de servicios técnicos y/o camareras de piso. A través de la aplicación Android, los encargados de servicios técnicos y/o las camareras de piso van a recibir estas incidencias para darlas por finalizadas cuando las resuelvan.

Para el desarrollo de esta aplicación se ha utilizado el IDE oficial de Android, llamado Android Studio y se ha creado un servidor *cloud* en la plataforma AWS para alojar la web, la base de datos y la API REST para la comunicación entre los dispositivos.

Para la obtención de los requisitos del proyecto, se ha contado con la colaboración de todos los empleados de la cadena hotelera BQ Hoteles con los que se han podido averiguar las necesidades que el proyecto podía cubrir.

Palabras clave

TFG, Gestión, Incidencia, Hotel, Android, Aplicación web, Scrum, Firebase, API REST.

OLUTÍPAC 1

Introducción

El turismo es uno de los sectores que generan más empleo en España y que sustentan la economía del país. De hecho, en 2016 solo este sector concentró el 11,2% del PIB y el 13% de la ocupación. Además, el año 2017 se ha cerrado con 82 millones de llegadas de turistas internacionales [1].

Cada vez son más los establecimientos del sector hotelero que optan por el uso de las tecnologías como nuevas herramientas de trabajo. Entre las tecnologías más usadas en el sector destacan los teléfonos móviles, los TPV (Terminal Punto de Venta), las páginas web y los programas de gestión. Estas herramientas influyen notablemente en las perspectivas económicas del sector mejorando la producción y competitividad, ajustándose a las necesidades de los clientes.

En los últimos años, el uso de dispositivos móviles ha aumentado exponencialmente. En 2017 el número de usuarios de móviles en el mundo asciende a 4,9 mil millones, lo que significa que el 66% de la población mundial ya cuenta con un teléfono móvil. El sistema operativo más usado es Android (con una penetración del 83% mientras que iOS solo alcanza un 11%) [2].

La utilización del ordenador en la recepción de las cadenas hoteleras se ha vuelto imprescindible para agilizar las entradas masivas de clientes, disponer de un servicio de caja, gestionar las habitaciones y para realizar el *check-out* entre otras muchas funciones.

BQ Hoteles es una empresa del sector turístico cuya actividad principal es la gestión de establecimientos hoteleros en las Islas Baleares. Con el objetivo de mejora continua en la gestión y en el ámbito de las TIC le ofreció al alumno la posibilidad de desarrollar una aplicación capaz de gestionar las incidencias producidas en los distintos establecimientos.

La situación con respecto a las incidencias antes de empezar el proyecto era la siguiente: en el *backoffice* de cada hotel, disponían de formularios en forma de hojas de papel (Ver Figura 1) los cuales los empleados de cada departamento (SSTT/Pisos) debían ir a cumplimentar cada vez que solucionaban una incidencia.

Estas incidencias, se comunicaban a través del teléfono móvil o mediante el uso de la palabra. Cada vez que un cliente reportaba algún desperfecto/queja/sugerencia a la recepción del hotel, el recepcionista avisaba por teléfono al encargado para que este realizara la acción correctiva pertinente. De esta manera, al final de cada día se guardaba una hoja de papel con todas las incidencias producidas a lo largo de la jornada.

Con este modo de gestionar las incidencias, la dirección de BQ Hoteles no disponía de ningún mecanismo para detectar las averías más frecuentes, sacar datos estadísticos relevantes, ni siquiera poder consultar de manera rápida las incidencias producidas un día concreto.

En este contexto, SSTTApp aprovecha el uso de teléfonos móviles y ordenadores en los establecimientos para gestionar todas las incidencias en tiempo real, así como para obtener informes de rendimiento y datos estadísticos de interés.

BQ Amfora Beach****

PARTE DE SERVICIOS TECNICOS 17/08/2018

FECHA	HORA	AVISADO POR	NOMBRE RECEP.	HAB	PARTE DE AVERIA	FECHA OK	OBSERVACIONES
17/8	8:45	Cedar	M.C	—	Baño Hombres tomador clientes	17/8 OK	
17/8	11	d	"	—	Sofocar ducha moto	17/8 OK	

FECHA	HORA	AVISADO POR	NOMBRE RECEP.	HAB	PARTE DE AVERIA	FECHA OK	MOTIVO PDTE
27/07/2018	5:00	EDDY	EDDY		REVISAR RUEDAS CONTENEDOR BASURA		ESTAN PEDIDAS
14/08/2018	9:00	CLIENTE	CHOLI	125	PUERTA DE HABITACION HACE RUIDO		MUELLE PEDIDO
15/08/2018	9:30	GOBERNANTA	CHOLY		BAÑO MUJERES ABAJO CLIENTES PUERTA HINCHADA		AVISAR CARPINTERO

Figura 1: Parte de incidencias antes de empezar el proyecto.

CAPÍTULO 2
Objetivos

El objetivo general del proyecto es el desarrollo de una solución de software (plataforma web y Android) que permita gestionar de manera eficiente las incidencias que se producen en los establecimientos hoteleros de la cadena BQ Hoteles.

Podemos descomponer este objetivo en objetivos más concretos:

- Mejorar la comunicación y productividad entre los distintos departamentos, poniendo en contacto a recepcionistas, operarios de servicio técnico y camareras de piso, evitando así que los técnicos pasen la mayoría de su tiempo desplazándose entre el punto de información de la incidencia al punto de localización de la misma.
- Supervisar a tiempo real el estado de las incidencias y de las habitaciones del establecimiento, con la consecuente bajada de la tasa de errores cometidos al entregar una habitación como limpia cuando hay una incidencia abierta.
- Obtener datos estadísticos de relevancia para que el director del hotel pueda aplicar medidas correctivas, así como generar informes de rendimiento de los empleados.

CAPÍTOL 3

Estado del arte

Dado que el objetivo principal del proyecto es desarrollar una página web y una aplicación capaz de gestionar las incidencias producidas en los distintos establecimientos de la cadena, se va a realizar un pequeño estudio de las aplicaciones existentes en el mercado y las distintas plataformas utilizadas. Además, se analizarán las diferentes metodologías de trabajo actuales para el desarrollo de software.

3.1. Análisis de Sistemas Operativos para aplicaciones móviles

En 2010 se vendieron 297 millones de móviles en todo el mundo y el 41% de ellos usaban sistemas operativos propios o minoritarios. Android solo estaba en el 22.7% de los terminales, e iOS en el 15.7%. BlackBerry, hoy casi desaparecido, aún se usaba en el 14% de los móviles. En 2017 se vendieron cerca de 1500 millones de móviles, y Android es usado en el 85,9% de todos ellos [3]. Estos datos se ven reflejados en la siguiente gráfica:

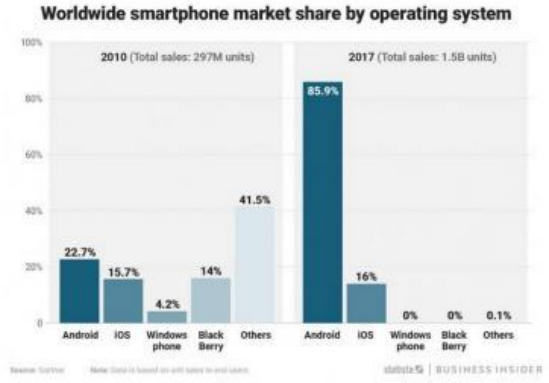

Figura 2: Comparativa de ventas entre Sistemas Operativos de móvil

En BQ Hoteles el 98% de los dispositivos móviles utilizan el sistema operativo Android. Dentro de la previsión de compras de la cadena, no está previsto cambiar de sistema operativo en el futuro, por lo que la aplicación que se ha desarrollado es exclusivamente para la plataforma Android.

3.2. Análisis de aplicaciones relacionadas con la gestión de incidencias hoteleras

Entre los productos existentes para la gestión de incidencias hoteleras destacan los siguientes:

- **GIMHOTEL**:

Figura 3: Gimhotel

Es una solución de software, para la gestión del mantenimiento del hotel y el control de los procesos realizados por el personal del departamento.
Entre otras funcionalidades ofrece: lista de incidencias, gestión de acciones correctivas, informes de incidencias, partes de trabajo, resúmenes mensuales, gestión de pisos y alertas y notificaciones.

- **EISIHOTEL**:

Figura 4: Eisihotel

Es otra de las herramientas que ha surgido en los últimos años para la gestión de incidencias, ofrecen entre otras cosas: registros de consumos, órdenes de trabajo con notificación, reparto y asignación de tareas, supervisión y asignación de tareas e informes directivos de hotel.

3.3. Análisis de las metodologías de trabajo actuales

En este apartado se definen las principales características de las metodologías más usadas. Estas metodologías se pueden agrupar en dos grandes grupos, la metodología tradicional y la metodología ágil.

- **Metodología tradicional**: en el inicio el desarrollo de software era artesanal en su totalidad, la fuerte necesidad de mejorar el proceso y llevar los proyectos a la meta deseada, se tuvo que importar la concepción y fundamentos de metodologías existentes en otras áreas y adaptarlas al desarrollo de software. Esta nueva etapa de adaptación contenía el desarrollo dividido en etapas secuenciales que mejoraba la necesidad latente en el campo del software. Esta metodología centra su atención en llevar una documentación exhaustiva de todo el proyecto y cumplir con un plan de proyecto definido en las fases iniciales del desarrollo. Se focalizan en la documentación, planificación y procesos (Plantillas, técnicas de administración, revisiones, etc.). Otra de las características importantes dentro de este enfoque son los altos costos al implementar un cambio y al no ofrecer una buena solución para proyectos donde el entorno es versátil [6].

- **Metodologías ágiles**: el desarrollo de software ágil es un concepto usado en el desarrollo de software para describir las metodologías de desarrollo incrementales (Cohen, Lindvall & Costa, 2003). Es una alternativa en la gestión tradicional de proyectos, donde se hace hincapié en el empoderamiento de las personas para colaborar y tomar decisiones en equipo, además potencia la planificación continua, pruebas permanentes y la integración conjunta del código y los despliegues.
  Las técnicas ágiles varían en prácticas y énfasis, pero comparten características comunes, incluyendo el desarrollo iterativo y un enfoque en la interacción, la comunicación y la reducción de artefactos intermedios que consumen muchos recursos. Desarrollar en iteraciones permite al equipo adaptarse rápidamente a las necesidades cambiantes. [7].

OLUTÍPAC 4

Metodología

Para la elección de la metodología de trabajo, se ha tenido en cuenta que se trata de un proyecto individual (siendo el alumno el único responsable del departamento de sistemas informáticos de la cadena hotelera) y que los requisitos recogidos inicialmente podían variar a lo largo de todo el proceso de desarrollo.

Con estos dos argumentos de peso, se ha optado por usar una de las metodologías ágiles más conocidas: Scrum, que fue propuesta por Ken Schwaber y Jeff Sutherland en 1995. Actualmente es la metodología ágil más usada y extendida en el mundo.

Los perfiles de esta metodología se pueden dividir en:

- ***Product Owner***: representa la voz de los interesados del proyecto. Se encarga de definir los objetivos del proyecto y de garantizar del modo adecuado para alcanzar dichos objetivos.
- ***Scrum Master***: es el encargado de asegurar que el resto del equipo no tiene problemas para abordar sus funciones y tareas.
- ***Scrum Team***: es el equipo encargado de desarrollar las iteraciones de cada sprint y el producto final.
- ***Stackeholders***: representa a los perfiles interesados en el proyecto.

Dado que el proyecto es individual, el alumno ha asumido los perfiles de *Product Owner*, y de *Scrum Team*, y bajo la supervisión del tutor (*Scrum Master*), ha decidido qué tareas y entregables debía realizar en cada iteración. Los *Stackeholders* representan a los empleados de todos los departamentos de la cadena: servicios técnicos, pisos, recepciones, directores y al de operaciones, el cual se nutrirá de los datos estadísticos relacionados con las incidencias para la toma estratégica de decisiones.

El proyecto comienza con la elaboración del llamado *Product Backlog*. Se trata de un archivo genérico que recoge el conjunto de tareas, los requisitos y las funcionalidades requeridas por la aplicación. Entre el tutor y el alumno, se asignaron prioridades a cada una de las tareas obtenidas como resultado de las reuniones iniciales entre la dirección de BQ Hoteles y el alumno.

La segunda etapa pasa por la definición del *Sprint Backlog*, documento que recoge las tareas a realizar, quién las desempeña y cuándo se realizan. El *Sprint* es el periodo en el que se realizan todas las acciones pactadas en el *Sprint Backlog* y supone entregas parciales para ir testeando el producto final. Bajo esta premisa, decidimos realizar *sprints* mensuales para la realización de las tareas definidas en el *Product Backlog* aprovechando la reunión mensual con el tutor. En estas reuniones se realizaban revisiones continuas del proyecto y se establecían las tareas a realizar en los próximos *sprints*.

El ciclo anterior, se irá repitiendo hasta que todos los elementos del *Backlog* estén entregados. Todas estas acciones han sido controladas en el *Burn Down*, donde se ha ido marcando el estado, la evolución de cada tarea y los requerimientos pendientes. Todo este proceso se aprecia en la figura que se muestra a continuación:

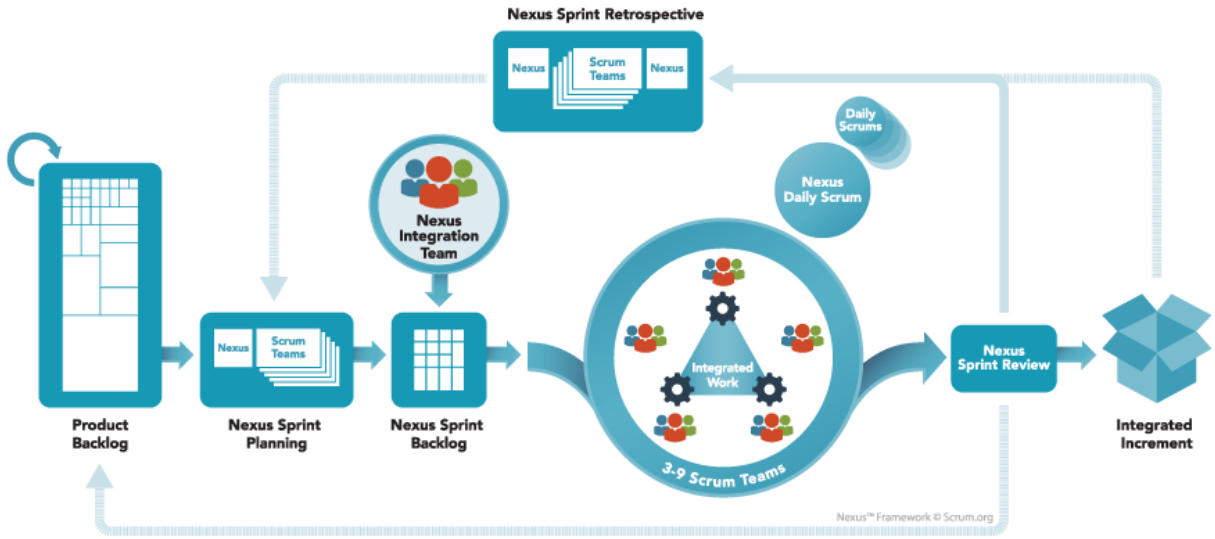

Figura 5: Diagrama de proceso SCRUM

CAPÍTULO 5

Desarrollo del proyecto

5.1. Requisitos y análisis

La definición más general de "Requisito" es la que brinda el Instituto de Ingeniería Electrónica y Eléctrica (IEEE):

- Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
- Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Durante la fase inicial del proyecto, el objetivo era intentar descubrir, en la medida posible, las necesidades o requisitos del proyecto, entender el dominio de la aplicación, las restricciones que debería tener y los usuarios que la iban a utilizar. La obtención de esta serie de necesidades se realizó mediante las siguientes practicas:

- **Entrevistas**: se realizó una entrevista con una selección de 5 empleados de cada departamento implicado: operarios de servicios técnicos, camareras de piso, recepcionistas, directores y operaciones. Para cada departamento el alumno discutió con los entrevistados las expectativas que tenían del sistema y de manera aislada, para el departamento de operaciones, se prepararon una serie de preguntas relacionadas con la captura de datos estadísticos de interés que les iba a proporcionar la aplicación.
- **Observación**: durante el tiempo trabajado en la empresa por parte del alumno, se ha ido observando la metodología tradicional de trabajo, concluyendo que existían una serie de carencias que se detallan a continuación:

- El exceso de llamadas para la comunicación entre los distintos departamentos suponía una pérdida de tiempo y dinero importantes.
- Baja productividad: los operarios de servicios técnicos pasan la mayoría de su tiempo desplazándose entre el punto de información de la incidencia al punto de localización de la misma.
- Abuso de hojas de papel para rellenar los formularios.
- Carencia de datos estadísticos para la toma de decisiones por parte del departamento de operaciones.

- Lluvia de ideas o ***BrainStorming***: se organizaron diversas reuniones con los jefes de cada departamento en las que todos participaron para aportar ideas de manera rápida y abierta.

Todas estas ideas, requisitos y necesidades han sido recogidas en el *Product Backlog* y se tratarán a continuación.

5.1.1. Roles de la aplicación

Podemos dividir en dos grandes roles los usuarios que van a usar la aplicación: usuarios de la página web y usuarios de la aplicación móvil para la plataforma Android. A su vez, estos dos roles, se pueden descomponer en roles más concretos:

Roles de la aplicación web:

- Recepcionistas: van a ser los encargados de generar las incidencias reportadas por los clientes y tendrán la posibilidad consultar la disponibilidad de las habitaciones.
- Directores: podrán gestionar sus hoteles trasladando la realidad del establecimiento a la aplicación: habitaciones, tipos de incidencia habituales, empleados, zonas y maquinaria. Además, tendrán la posibilidad de consultar algunos datos estadísticos de interés como el número de incidencias totales, porcentaje de incidencias resueltas o el empleado del mes.
- Administrador: en este caso, el alumno es el administrador. Tendrá acceso a todas las funcionalidades de la aplicación. Con respecto a los otros dos roles, el administrador podrá gestionar todos los establecimientos, así como todos los usuarios incluidos los directores.

Roles de la aplicación Android:

- Operarios de Servicio Técnico: van a ser los encargados de finalizar las incidencias con las observaciones pertinentes, así como de generar nuevas incidencias para otros operarios de servicios técnicos o para las camareras de piso. Además, podrán visualizar de manera detallada su lista de incidencias pendientes.
- Camareras de piso: cumplen rol similar al de servicios técnicos, podrán generar nuevas incidencias para los operarios de servicios técnicos y finalizar incidencias pendientes. También tendrán la posibilidad de visualizar sus incidencias.

5.1.2. Requisitos funcionales

En la siguiente lista se muestran los requisitos funcionales que debe cumplir la aplicación, agrupados según el tipo de usuario descrito en el apartado anterior:

Usuarios de la aplicación web:

- **Recepcionistas**:

  RF.01: Iniciar sesión en la aplicación mediante la inserción de su correo electrónico (único) y la contraseña asociada a la cuenta (debe contener letras mayúsculas, minúsculas, números y tener un mínimo de 6 caracteres de longitud).

  RF.02: Visualizar una lista de todas las incidencias (pendientes/realizadas) pertenecientes a su hotel con la posibilidad de modificar/crear nuevas incidencias con la siguiente información: título, tipo de incidencia, prioridad, habitación, zona, estado, máquina, operario asignado, fecha de inicio, fotografía y, para las finalizadas, fecha de finalización.

  Esta lista podrá ser ordenada y filtrada por palabras clave, así como por campo. Además, tendrá la posibilidad de cambiar el número de incidencias que aparezcan por página, facilitando así la búsqueda.

  RF.03: Visualizar un listado con el estado de las habitaciones del hotel el cual reflejará si las habitaciones están listas para la entrada de clientes o están bloqueadas por incidencia.

RF.04: Recibir una notificación cuando algún operario ha creado/finalizado una incidencia.

RF.05: El usuario podrá cerrar sesión saliendo de la aplicación.

- **Directores**:

RF.06: Iniciar sesión en la aplicación mediante la inserción de su correo electrónico (única) y la contraseña asociada a la cuenta (debe contener letras mayúsculas, minúsculas, números y tener un mínimo de 6 caracteres de longitud).

RF.07: Visualizar una lista con todos los hoteles a su cargo con la posibilidad de acceder a ellos para obtener información detallada.

RF.08: Podrá consultar los siguientes datos estadísticos representados mediante diferentes gráficas para cada uno de sus establecimientos: incidencias totales del hotel, incidencias resueltas, número de empleados, total de horas invertidas por sus empleados, el número de incidencias agrupadas por el mes del año y el empleado más productivo del mes.

RF.09: De igual manera que los recepcionistas, podrá consultar una lista de todas las incidencias (pendientes/realizadas) pertenecientes al hotel con la posibilidad.

RF.010: Visualizar una lista con todos los empleados del hotel con la posibilidad de eliminar/modificar/crear empleados. La información relativa a cada empleado es la siguiente: nombre, correo, contraseña, imagen, teléfono, DNI, fecha de nacimiento y el rol al que pertenece (Servicios técnicos, pisos o recepción).
Además, podrá visualizar una gráfica con el número de incidencias asignadas a cada empleado.

RF.011: Podrá crear/modificar/eliminar los tipos de incidencias, habitaciones, zonas y máquinas de su hotel siendo así capaz de personalizar el hotel a su gusto.

RF.012: El usuario podrá cerrar sesión saliendo de la aplicación.

- **Administrador**:

  RF.013: Iniciar sesión en la aplicación mediante la inserción de su correo electrónico (única) y la contraseña asociada a la cuenta (debe contener letras mayúsculas, minúsculas, números y tener un mínimo de 6 caracteres de longitud).

  RF.014: Visualizar una lista con todos los hoteles a su cargo con la posibilidad de acceder a ellos para obtener información detallada. Puede crear/eliminar/modificar los hoteles, así como modificar/eliminar/crear tipos de incidencias, habitaciones, máquinas y zonas.

  RF.015: Modificar/eliminar/crear usuarios de la aplicación.

  RF.016: Consultar datos estadísticos de interés relacionados con los hoteles como las incidencias totales, incidencias resueltas, incidencias por hotel o empleado del mes (teniendo en cuenta todos los hoteles).

  RF.017: El Administrador podrá cerrar sesión saliendo de la aplicación.

Usuarios de la aplicación Android:

- **Operarios de servicios técnicos/camareras de piso:**

  RF.018: Iniciar sesión en la aplicación mediante la inserción de su correo electrónico (única) y la contraseña asociada a la cuenta (debe contener letras mayúsculas, minúsculas, números y tener un mínimo de 6 caracteres de longitud).

  RF.019: Visualizar una lista de sus incidencias con la posibilidad de ver toda la información detallada relacionada con la incidencia.

  RF.020: Dará por finalizada una incidencia, con el correspondiente comentario asociado.

  RF.021: Crear una nueva incidencia con los siguientes campos: título, descripción, habitación, zona, máquina, tipo de incidencia, prioridad, empleado encargado de resolverla y opcionalmente podrá adjuntar una fotografía obtenida mediante el teléfono móvil.

  RF.022: Recibir una notificación cuando algún operario le ha asignado una nueva incidencia.

RF.023: Establecer como preferencia las incidencias más antiguas de: siete, treinta, ciento ochenta, trescientos sesenta días o todas.

RF.024: Establecer como preferencia la recepcion o no de las notificaciones asignadas, así como la posibilidad de hacer sonar o vibrar el dispositivo.

RF.025: El usuario podrá cerrar sesión saliendo de la aplicación.

5.1.3. Requisitos no funcionales

Los requisitos no funcionales son aquellos que especifican criterios para evaluar la operación del proyecto. A continuación, se describen los relacionados con la seguridad, eficiencia, usabilidad, mantenimiento, rendimiento y portabilidad:

Seguridad:

RNF.01: Las contraseñas de todos los usuarios de la aplicación se guardarán en la base de datos de manera encriptada mediante MD5.

RNF.02: Las conexiones con el servidor se realizan mediante conexión segura utilizando SSL.

RNF.03: El servidor estará localizado en un entorno protegido con algún sistema de seguridad.

RNF.04: Solo los usuarios que previamente hayan sido registrados tendrán acceso a la aplicación.

Eficiencia:

RNF.05: El tiempo de acceso a la aplicación Android no será mayor a 1 segundo.

RNF.06: La aplicación de Android deberá tener un tamaño máximo de 10 MB y deberá ser lo más ligera posible para disminuir el uso de batería y almacenamiento.

RNF.07: El tiempo de procesamiento del servidor entre que recibe la consulta hasta que devuelve los datos y el usuario los puede ver en la aplicación no será superior a dos segundos.

Usabilidad:

RNF.08: La página web deberá ser *responsive* y adaptable a cualquier dispositivo y pantalla.

RNF.09: La página web y la aplicación Android deberán informar mediante errores mostrados al usuario cualquier acción no permitida o errónea.

RNF.010: Tanto la página web como la aplicación Android deberán ser capaces de recibir y mandar notificaciones PUSH.

RNF.011: Las diferentes pantallas de la aplicación de Android seguirán el mismo patrón de estilo y diseño para facilitar la adaptación del usuario final. Además, se seguirá la filosofía *Material Design* creada por Google.

RNF.012: La página web deberá mandar un correo de bienvenida con el usuario y la contraseña a los nuevos usuarios y/o un correo de información con la nueva contraseña en caso de cambio.

Mantenibilidad:

RNF.013: El sistema deberá ser escalable, adaptándose al posible crecimiento de la base de datos.

Portabilidad:

RNF.014: El sistema deberá ser accesible desde cualquier parte del mundo mientras se cuente con una conexión a internet.

RNF.015: La aplicación Android deberá estar publicada en Play Store.

5.2. Diseño y arquitectura

Identificados todos los requisitos funcionales y no funcionales que deberá tener la aplicación, en esta sección se especificará el diseño de la aplicación web, la aplicación Android y la API desarrollada para la comunicación entre la base de datos y el dispositivo móvil. Esta etapa ha sido realizada antes del primer sprint mensual.

5.2.1. Diseño de la aplicación

A continuación, se muestran los mapas de navegación para las dos plataformas en las que se apoya la aplicación y las funciones que deberá tener la API:

Mapa de navegación de la página web: a partir de las figuras 6 y 7 se describe la navegación que deberá tener un usuario cuando acceda a la página. La primera página que ve el usuario es la página de login, en función del usuario introducido (administrador, recepcionista o director), se le redirige hacia una página u otra.

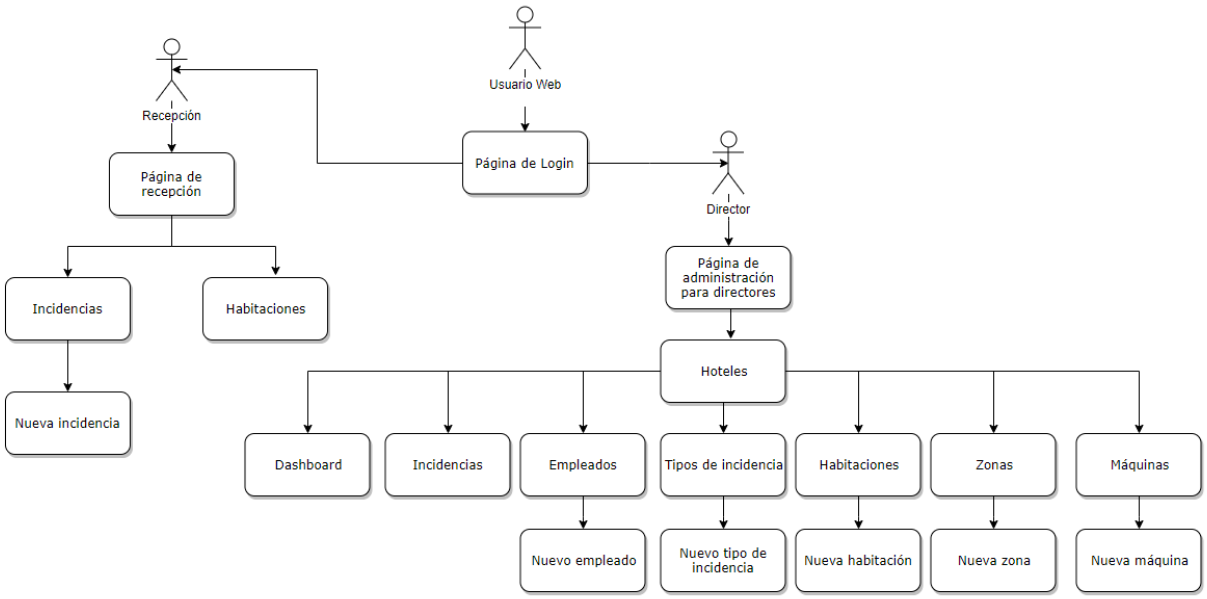

Figura 6: Mapa de navegación de la aplicación web

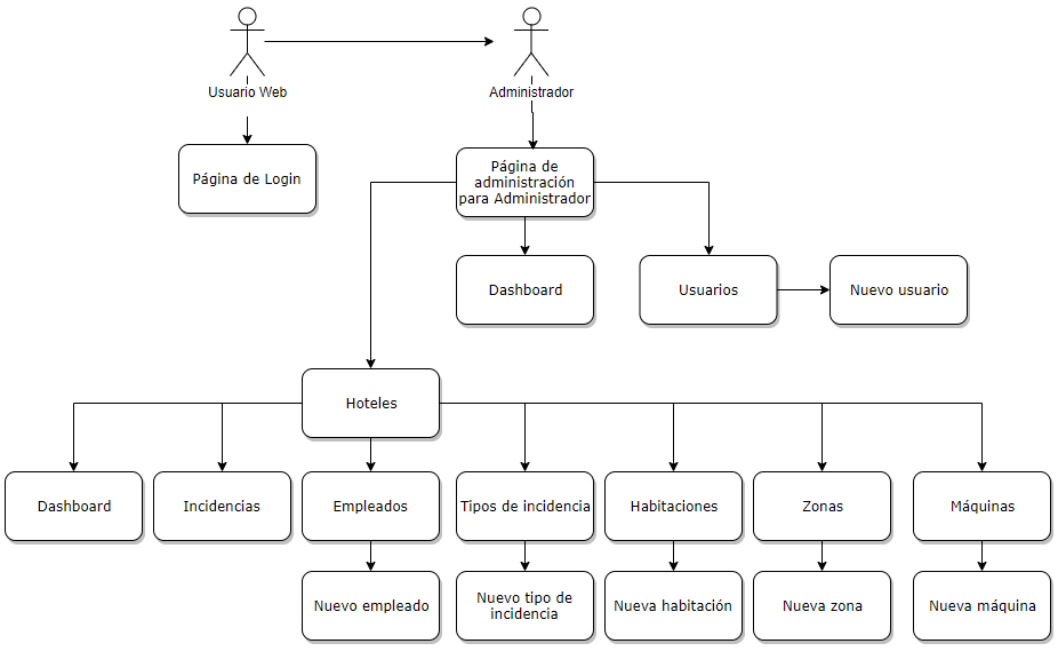

Figura 7: Mapa de navegación de la aplicación web

Mapa de navegación de la aplicación Android: en la figura 8 se puede observar el recorrido lógico de la aplicación.

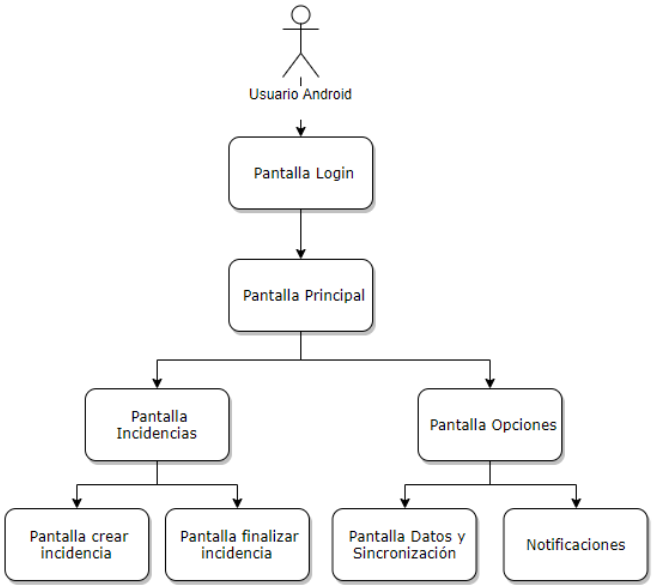

Figura 8: Mapa de navegación de la aplicación Android.

Funciones de la API REST: el término REST (*Representational State Transfer*) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.

Las operaciones más importantes que nos permitirán manipular los recursos son cuatro: GET para consultar y leer, POST para crear, PUT para editar y DELETE para eliminar. Para obtener tiempos de respuesta cumpliendo los requisitos de la aplicación se ha decido usar el formato JSON tanto para peticiones al servidor como para las respuestas.

Para este proyecto, se va a utilizar la API REST para la comunicación entre los dispositivos móviles que tengan instalada la aplicación Android y el servidor. Las funciones que se van a utilizar en la API están definidas en la tabla 1.

Funciones	Operación	Parámetros de la petición	Respuesta del servidor
Login de usuario	POST	Email y Password.	Si el usuario es correcto: se devuelve la información relativa al usuario. Si no es correcto se devuelve una respuesta negativa. En caso de error se devuelve el correspondiente mensaje de error.
Obtener incidencias de un usuario	GET	Estado de las incidencias, identificador del usuario y antigüedad de las incidencias.	Devuelve todas las incidencias solicitadas que cumplan las condiciones solicitadas. En caso de error se devuelve el correspondiente mensaje de error.

Obtener información sobre una incidencia concreta	GET	Identificador de la incidencia.	Devuelve toda la información almacenada en la base de datos relacionada con la incidencia solicitada. En caso de error se devuelve el correspondiente mensaje de error.
Finalizar una incidencia	PUT	Identificador de la incidencia, identificador del usuario que la ha realizado, el comentario asociado y el identificador del hotel al que pertenece la incidencia	Devuelve un mensaje de aceptación en caso de que se haya finalizado la incidencia correctamente. En caso de error se devuelve el correspondiente mensaje de error.
Registrar el token generado por Firebase para un usuario concreto	PUT	Identificador del usuario y el token asociado al dispositivo.	Devuelve un mensaje de aceptación en caso de que se haya guardado el token correctamente en la base de datos. En caso de error se devuelve el correspondiente mensaje de error.
Crear nueva incidencia	POST	Título, descripción, imagen, prioridad, tipo de incidencia, zona, máquina, habitación, usuario creador, usuario realizador y el hotel al que pertenece la incidencia.	Devuelve un mensaje de aceptación en caso de que se haya registrado la incidencia correctamente en la base de datos. En caso de error se devuelve el correspondiente mensaje de error.

Tabla 1: especificación de funciones para la API REST

5.2.2. Arquitectura de la aplicación

La arquitectura del sistema define los elementos estructurales principales del software, así como las relaciones existentes entre los mismos.

Para la arquitectura de la aplicación se ha decidido usar la programación por capas. La programación por capas es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente-servidor: lógica de negocios, capa de presentación y capa de datos. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo afectará al nivel requerido sin tener que revisar entre el código fuente de otros módulos.

- **Capa de presentación**: Es la capa que ve el usuario final (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario.
- **Capa de negocio**: es donde residen los scripts que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.
- **Capa de datos**: es la capa donde residen los datos y es la encargada de acceder a los mismos. Estará formada por un gestor de bases de datos que realiza todo el almacenamiento de datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio [10].

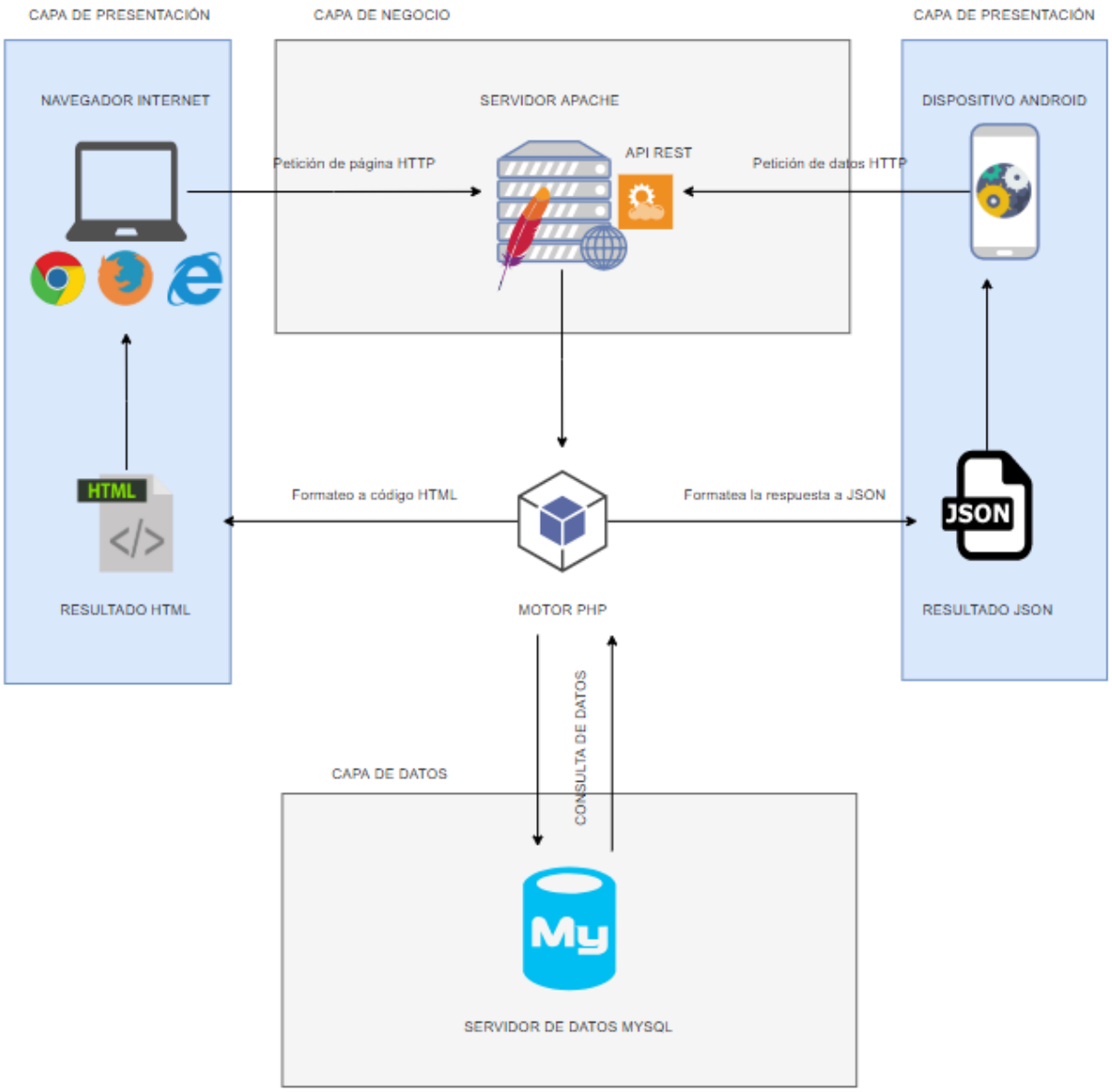

Figura 9: Arquitectura de SSTTApp.

5.2.3. Infraestructura del sistema

La infraestructura del sistema se puede definir como el conjunto de elementos de hardware, software y servicios necesarios para construir la aplicación cumpliendo con los requisitos establecidos. A continuación se explican todos los elementos utilizados:

- **Servidor Cloud:** para alojar y gestionar el servidor se ha decidido utilizar Amazon EC2. *Amazon Elastic Compute Cloud* (Amazon EC2) es un servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable. Está diseñado para facilitar a los desarrolladores el uso de la informática en la nube a escala de la Web.
  La elección de EC2 se debe a las ventajas que ofrece Amazon con este producto frente a la utilización de un servidor físico tradicional: permite aumentar o disminuir la capacidad en minutos, ofrece la posibilidad de elegir el sistema operativo deseado, proporciona una funcionalidad de red sólida y segura para los recursos.
  El sistema operativo para el proyecto es Microsoft Windows Server 2012 R2 *Standart Edition* con una arquitectura de 64 bits. La elección de este sistema operativo frente a la distribución Linux tiene que ver con las siguientes razones:
  - Gestión: los servidores Linux están basados en líneas de comandos mientras que los servidores de Windows incluyen la funcionalidad de Escritorio Remoto, que facilita el acceso y la administración debido a la interfaz gráfica proporcionada.
  - Complejidad: los servidores y el software de Windows generalmente tienen más características y elementos extra, que el software de código abierto.
  - Soporte técnico: si bien los servidores de Windows y las aplicaciones de Windows tienen un costo, también suelen contar con un buen soporte técnico.

- **Servidor web:** el servidor web será el encargado de contener todas las páginas y los Scripts de la aplicación, así como de la API REST. Si bien hay una gran complejidad que apuntala cómo funciona un servidor web, el trabajo básico de todos los servidores web es aceptar solicitudes de clientes y luego enviar una respuesta a esa solicitud en forma de página web. A continuación, se presenta una

comparativa entre los dos servidores gratuitos más utilizados en la actualidad como refleja la figura 10: Apache y Gnix.

- **Gnix**: el proyecto Nginx —pronunciado en inglés como "Engine x" o motor x— es un servidor web desarrollado por Igor Sysoev. El desarrollo de Nginx comenzó en 2002, y se creó con el fin de responder a sitios de muy alto tráfico. Gnix puede ejecutarse en varios sistemas operativos, incluyendo OS X, Linux, Unix, Solaris y Windows y es un software gratuito y de código abierto.

  Entre las características principales de Nginx se incluyen: opciones de almacenamiento en caché, estabilidad, balanceo de carga y utilidad para sitios con alto tráfico
- **Apache**: es el servidor web por excelencia, su desarrollo comenzó en 1995 y continúa hasta el día de hoy. Apache nace a finales de febrero del año 1995 con el fin de ampliar el desarrollo del demonio HTTP, el cual se había estancado debido a que su desarrollador principal —Robert McCool— había dejado el proyecto. Un grupo de desarrolladores se unió para continuar el proyecto hasta el lanzamiento de la primera versión de Apache en abril de 1995. Menos de un año después, Apache se catalogó como el primer servidor web a nivel mundial, puesto que disputa hasta el día de hoy.

  Algunas características comunes vistas en Apache incluyen: .htacces, ipv6, ftp, Perl, Lua y PHP, webDAV, balanceo de carga, re-escritura de URLs, rastreo de sesión y Geolocalización basada en IP.

Para este proyecto se ha decidido la utilización de Apache por los siguientes motivos: es el más utilizado en la actualidad por lo que podemos encontrar gran cantidad de guías, manuales, tutoriales e incluso libros que abordan distintos temas del servidor, es flexible y puede manejar contenido estático y dinámico sin problemas mediante el uso de MPMs y por último, la baja concurrencia de usuarios en la aplicación.

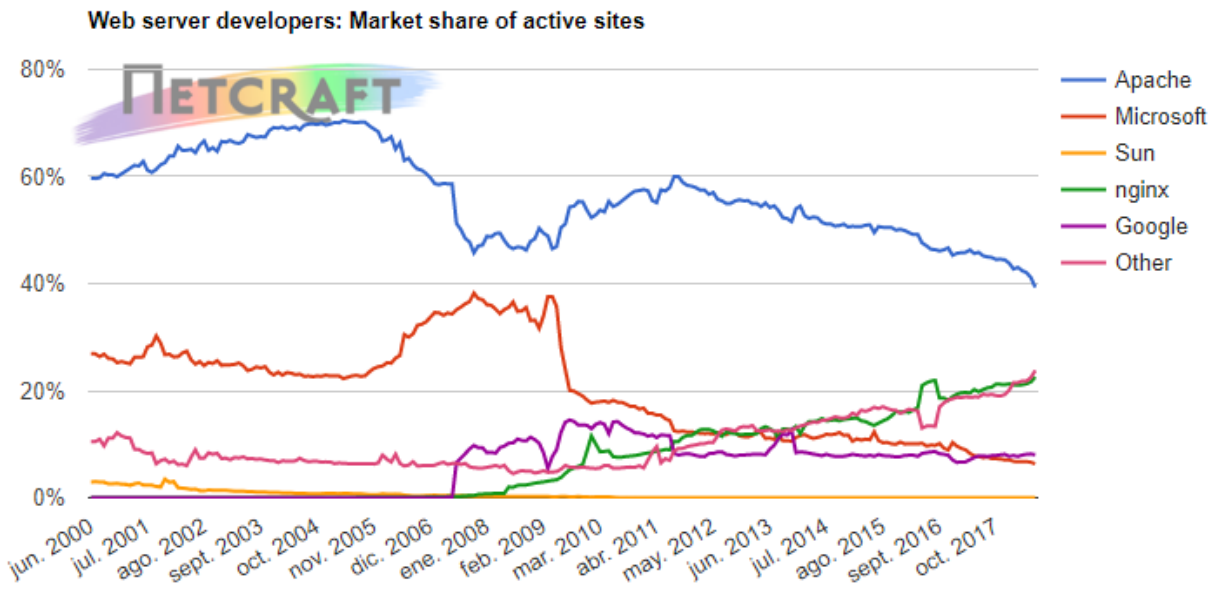

Figura 10: Comparativa de servidores web para las webs activas.

- **Servidor de datos**: aquí residen todos los datos de la aplicación, es decir todas las tablas (entidades, registros relacionales y atributos). El motor de base de datos escogido frente a PostgreSQL es MySQL. Esta elección se debe a las siguientes razones: la velocidad a la hora de realizar las consultas, el bajo consumo de memoria, las utilidades de administración que incorpora y la baja probabilidad de la corrupción de los datos.
- **Core de la aplicación**: es el núcleo de la aplicación, en ocasiones conocido como *Server-side scripting*. Existen varias tecnologías en la actualidad para desarrollar código del lado del servidor, a continuación, se comparan dos de las más utilizadas en la actualidad: PHP y ASP.NET.

  PHP o *Hypertext Pre-processor* fue creado en 1994 y en sus comienzos estaba diseñado para plataformas Linux, aunque actualmente ya es compatible con cualquier sistema operativo. Por su parte, ASP o *Active Server Pages*, también conocido como ASP clásico, es una tecnología de Microsoft especialmente diseñada para funcionar solo en Microsoft Windows [16].

  Una de las ventajas que presenta PHP es que su código es libre y abierto, por lo que está disponible de manera gratuita y es accesible a todo el mundo. Por su parte, para programar en ASP hay que disponer de un paquete de Windows y abonar las licencias correspondientes.

El hecho de que PHP sea de código abierto implica que hay publicada mucha más información y manuales que los que se pueden encontrar para ASP. Asimismo, también hay disponibles numerosas librerías y recursos que crean los usuarios de PHP que se pueden utilizar para facilitar la programación.
Para el desarrollo del proyecto se ha decido utilizar PHP en base a las ventajas comentadas anteriormente.

- **Código del lado del cliente**: también llamado *Client-side scripting* es el ámbito en el cual un cliente ejecuta sus operaciones en una relación cliente-servidor dentro de una aplicación. Para el desarrollo de la aplicación web se van a utilizar los siguientes lenguajes y tecnologías: HTML5, CSS3, Bootstrap y jQuery.
- **API REST**: para la forma en que se construyen las peticiones y las respuestas se decidió utilizar JSON frente a XML, las razones se describen a continuación: JSON utiliza un formato sumamente simple, la velocidad de procesamiento es más elevada que XML y, por último, los archivos JSON tienen un tamaño menor.

5.2.4. Diseño de la base de datos

Analizando todas las características, requisitos y necesidades que emanaban del proyecto se determinaron las tablas en las que se guardaría toda la información generada por la aplicación, así como las relaciones entre ellas:

- **Director**(idDirector, nombre, email, password, fecha_creacion, superAdmin, activo): es la tabla encargada de guardar toda la información relativa a los directores. El identificador se va incrementando automáticamente y los atributos email y nombre son únicos. Además, el atributo superAdmin identifica a los administradores, que tendrán unos privilegios superiores a los directores.

  La tabla Director está relacionada con Hotel de manera que un director puede dirigir varios hoteles, pero un hotel solo puede ser dirigido por un director.

- **Empleado**(idEmpleado, nombre, fecha_nacimiento, teléfono, dni, email, rol, password, coste, imagen, token_firebase, fecha_creacion, idHotel, activo): tabla en la que se guardara toda la información relativa a los usuarios. El identificador se va incrementando automáticamente y los atributos nombre y email son únicos. El atributo rol es un enumerado y puede adquirir los valores: “recepcion”, “sstt” y “pisos”. Tiene una clave foránea “idHotel” que hace referencia al identificador idHotel de la tabla Hotel.

La tabla Empleado está relacionada con Hotel y con Incidencia, de manera que un empleado realiza o crea varias incidencias, pero una incidencia solo puede ser creada o asignada a un solo empleado. De igual manera, un empleado pertenece a un solo hotel y un hotel puede tener varios empleados.

- **Habitacion**(idHabitacion, nombre, tipo, fecha_creacion, idHotel, activo): tabla encargada de guardar todas las habitaciones del hotel. El identificador se va incrementando automáticamente y tiene una clave foránea: "idHotel" que hace referencia al identificador idHotel de la tabla Hotel.

  La tabla Habitacion está relacionada con Hotel y con Incidencia, de manera que una habitación puede estar involucrada en varias incidencias, pero una incidencia solo puede estar relacionada con una habitación. Respecto a Hotel, un hotel puede tener varias habitaciones, pero una habitación solo puede pertenecer a un hotel.

- **Hotel**(idHotel, nombre, direccion, población, nombre_fiscal, calidad, descripción, cif, imagen, teléfono, email, fecha_creacion, activo): tabla encargada de guardar toda la información de los hoteles registrados en la aplicación. El identificador se va incrementando automáticamente y los atributos nombre, nombre_fiscal y CIF son únicos.

  La tabla hotel está relacionada con Director y con Tipo_incidencia, Zona, Maquina, Habitacion y Empleado de tal manera que un hotel es dirigido por un solo director. La relación con las otras tablas es de uno a muchos ya que un hotel puede tener varios tipos de incidencia, habitaciones, máquinas, zonas y empleados.

- **Maquina**(id_Maquina, nombre, fecha_creacion, idZona, activo): es la tabla encargada de guardar toda la información de las máquinas de los hoteles. El identificador se va incrementando automáticamente y tiene una clave foránea: "idZona" que hace referencia al identificador "idZona" de la tabla Zona.

  La tabla Maquina está relacionada con Incidencia y con Zona, de modo que una máquina pertenece a una sola zona y una máquina puede pertenecer a varias incidencias, pero una incidencia solo tiene una máquina asociada.

- **Tipo_incidencia**(idTipo_incidencia, nombre, fecha_creacion, idHotel, activo): tabla en la que se almacena toda la información relativa a los tipos de incidencia más comunes ocurridos en los hoteles. El identificador se va incrementando

automáticamente y tiene una clave foránea: “idHotel” que hace referencia al identificador idHotel de la tabla Hotel.

La tabla Tipo_incidencia está relacionada con Incidencia y con Hotel, de tal manera que una incidenica solo puede tener un tipo de incidencia asociado y un tipo de incidencia solo puede pertenecer a un hotel.

- **Zona**(idZona, nombre, fecha_creacion, idHotel, activo): tabla en la que se almacenan todas las zonas pertenecientes a cada establecimiento hotelero. El identificador se va incrementando automáticamente y tiene una clave foránea: “idHotel” que hace referencia al identificador idHotel de la tabla Hotel.

  La tabla Zona está relacionada con Hotel y con Incidencia, de manera que una zona puede estar involucrada en varias incidencias, pero una incidencia solo puede estar relacionada con una zona. Respecto a Hotel, un hotel puede tener varias zonas, pero una zona solo puede pertenecer a un hotel.

- **Incidencia**(idIncidencia, titulo, descripción, imagen, prioridad, estado, fecha_inicio, fecha_fin, comentario, idTipo_incidencia, idZona, idMaquina, idEmpleado_creador, idEmpleado_realizador, idHotel, activo): es la tabla más importante de la base de datos, en ella se almacenarán todas las incidencias que se vayan produciendo en los hoteles. El identificador se va incrementando automáticamente y tiene las siguientes claves foráneas que hacen referencia a las otras tablas descritas anteriormente: “idTipo_incidencia”, “idZona”, “idMaquina”, “idEmpleado_creador”, “idEmpleado_realizador” e “idHotel”.

  La tabla incidencia está relacionada con Hotel, Tipo de incidencia, Zona, Maquina, Habitacion, y doblemente con Empleado. En cuanto a la relación con Hotel, se puede decir que un hotel puede tener muchas incidencias, pero una incidencia solo pertenece a un hotel. Las relaciones con las otras tablas son uno a uno ya que una incidencia solo puede tener un tipo de incidencia, una zona, una máquina y una habitación asociada. En cuanto a la relación doblemente enlazada entre incidencia y empleado se debe a que un empleado es el que crea la incidencia y otro empleado la realiza, de modo que un empleado puede crear varias incidencias y una incidencia solo puede ser creada por un usuario, lo mismo pasa con la realización, es decir, un empleado puede realizar varias incidencias, pero una incidencia solo puede ser realizada por un empleado.

5.2.5. Diseño de la interfaz de usuario

En el entorno de interacción usuario-aplicación, la interfaz (o interfaz de usuario) es lo que permite que la interacción entre el usuario y la aplicación ocurra. Es decir, la interfaz permite:

1. Que el usuario pueda controlar efectivamente la aplicación.
2. Que el usuario reciba respuestas de la aplicación que le permitan saber si la interacción es correcta y cómo seguir actuando [17].

Para diseñar la interfaz de usuario de SSTTApp se ha utilizado una plataforma en la nube llamada "Draw.io". Esta aplicación permite crear bocetos, así como maquetas de nuestras páginas web y pantallas de Android incluyendo la mayoría de los elementos visuales necesarios. A continuación, se presentarán de manera separada algunas de las maquetas realizadas durante el diseño de la aplicación:

Maquetas generadas para la aplicación Android:

Figura 11: Pantalla Login.

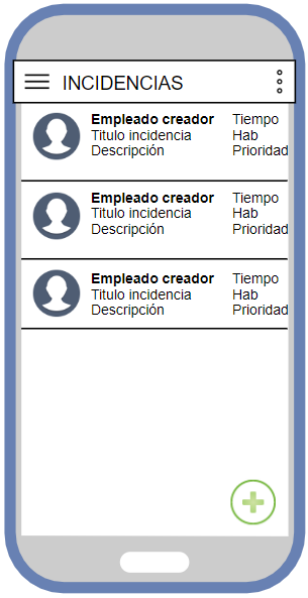

Figura 12: Pantalla principal.

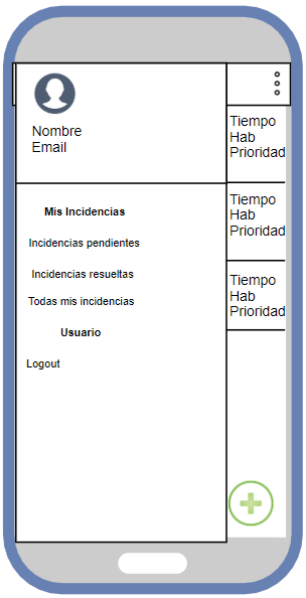

Figura 13: Navigation Drawer.

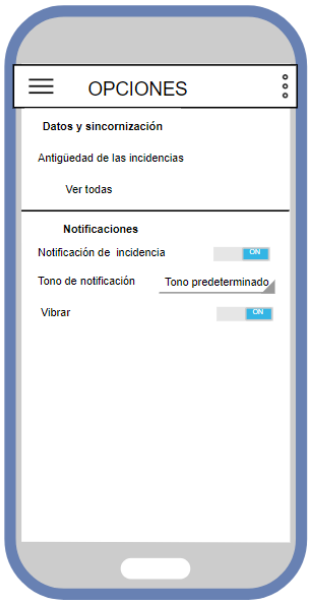

Figura 14: Pantalla Opciones.

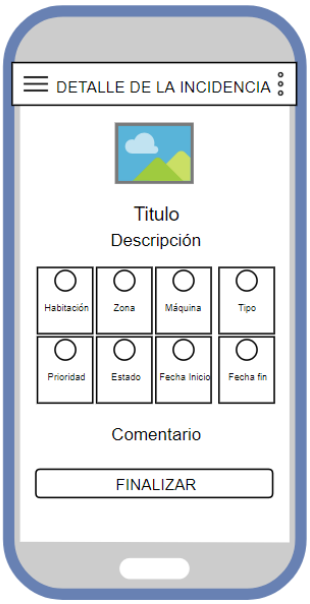

Figura 15: Detalle incidencia.

Figura 16: Nueva Incidencia.

Maquetas generadas para la página web:

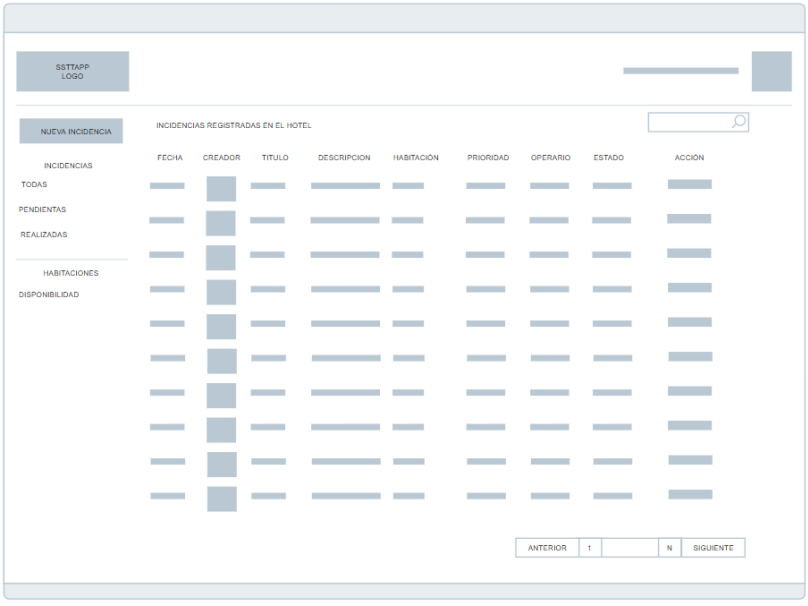

Figura 17: Página principal para los usuarios Recepcionistas.

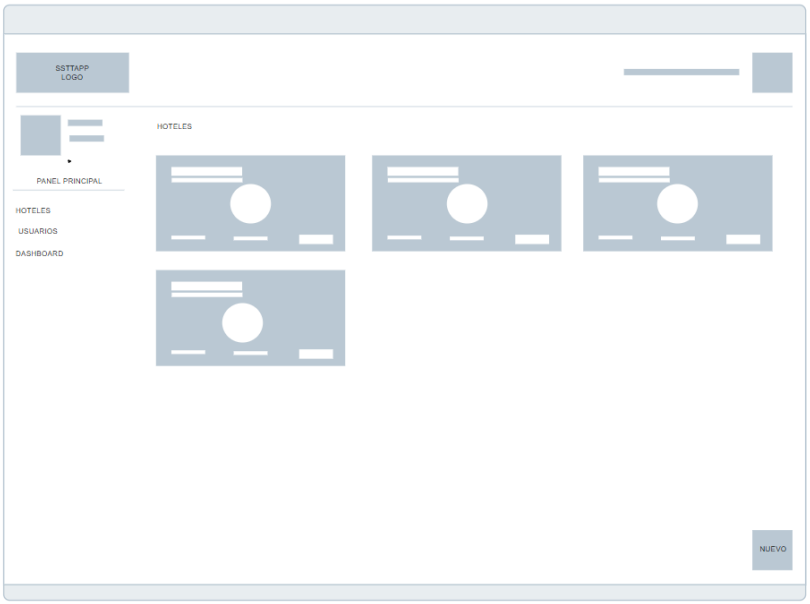

Figura 18: Pantalla principal de administración para el rol “Administrador”.

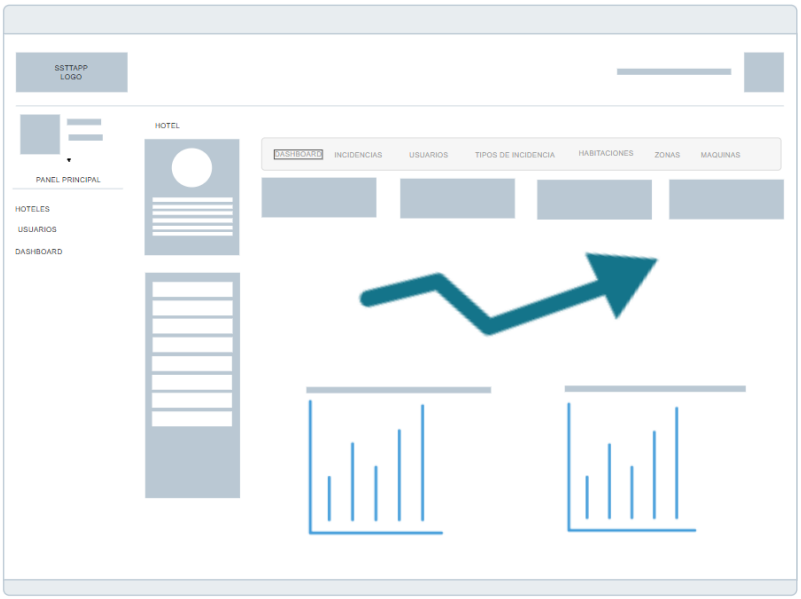

Figura 19: Pantalla de Hotel seleccionado para el rol “Administrador”.

5.3. Desarrollo de la aplicación

En este capítulo se va a explicar y presentar todo lo relacionado con el desarrollo de cada *User Story* cumpliendo con los diseños especificados en el apartado anterior. En primer lugar, se describirán las tecnologías utilizadas y posteriormente, con la ayuda de fragmentos de código, los tres grandes bloques de la aplicación: la página web, la API REST y la aplicación Android. Además, se explicarán las funcionalidades principales necesarias para que la plataforma funcione de la manera especificada.

5.3.1. Tecnologías y herramientas

- **HTML5**: HTML, sigla en inglés de *HyperText Markup Language* (lenguaje de marcas de hipertexto), es un lenguaje de marcado utilizado para la creación de páginas web. Define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. HTML5 (*HyperText Markup Language*, versión 5) es la quinta revisión importante de HTML y establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos [18].
- **CSS3**: siglas en inglés de *Cascading Style Sheets* es un lenguaje de diseño gráfico utilizado para definir la vista de un documento HTML. Es usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML [19].
- **Bootstrap**: es un *framework* de código abierto para desarrollar con HTML, CSS y JS. Permite hacer prototipos rápidos y adaptables a todos los tamaños de pantalla [20].
- **JavaScript**: es un lenguaje interpretado orientado a objetos desarrollado por Netscape que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativo [21].
- **jQuery**: es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web [22].
- **Ajax**: acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas

aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones [23].

- **PHP**: (acrónimo recursivo de PHP: *Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. [24].
- **Slim**: es un framework basado en PHP que permite crear API REST de manera rápida y sencilla. Este framework ha ayudado en la creación de la API y la comunicación con el cliente, es decir: la validación de los tipos de petición (GET, POST, PUT, DELETE ...) y la creación de las rutas para generar las peticiones.
- **Json**: acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente. Se ha utilizado para la comunicación de la API con el cliente y viceversa.
- **MySQL**: es un sistema de gestión de bases de datos relacional de código abierto desarrollado por Oracle Corporation. Es el sistema de gestión de bases de datos abierto mas popular en la actualidad.
- **Firebase**: es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por James Tamplin y Andrew Lee en 2011 y adquirida por Google en 2014. Para la aplicación se ha utilizado uno de sus componentes más famosos: "*firebase messaging*", que ha permitido la creación de mensajes *push* para los dispositivos móviles y aplicación web en forma de notificaciones [28].
- **Android Studio**: es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA [29]. Se ha utilizado durante todo el desarrollo de la aplicación Android SSTAapp.
- **XAMPP**: es una distribución de Apache gratuita y fácil de instalar que contiene MariaDB, PHP y Perl [30]. Se ha utilizado dentro de la instancia EC2 para crear el servidor web y el servidor de base de datos.

5.3.2. Plataforma web

Para la construcción de la plataforma web, siguiendo con la arquitectura definida en el apartado 5.2.2, decidimos separar el desarrollo en las siguientes capas: capa de presentación, capa de negocio y capa de datos.

Esta separación queda reflejada en la estructura del directorio de la página web que podemos observar en la figura 20.

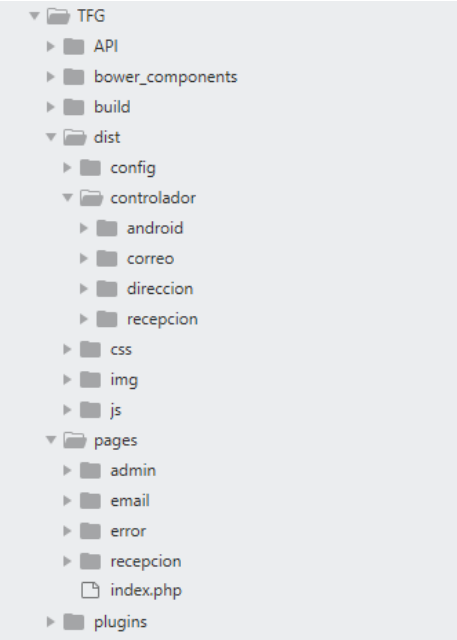

Figura 20: Estructura del directorio de la página web.

Como podemos observar el grueso de la plataforma está dividida en las siguientes carpetas:

- **TFG/pages**: contiene las páginas html y php visibles por el usuario. Dentro de esta carpeta podemos encontrar las siguientes subcarpetas: admin, email, error, recepcion y la página más importante, index.html que será la primera página que visite el usuario.

  Dentro de la carpeta admin, se encuentran todas las páginas relacionadas con la gestión de los hoteles tanto para los administradores como para los directores.

En la carpeta Email, encontraremos las páginas relativas al correo de bienvenida y modificación de contraseña que genera automáticamente el sistema cada vez que se crea/modifica un usuario.

En la carpeta error, podemos encontrar las páginas de error 404 (página no encontrada) y 405 (no se disponen de permisos para el acceso a la carpeta/fichero).

Para finalizar, la carpeta Recepcion, contiene todas las páginas que van a utilizar los recepcionistas, tanto para la visualización de las incidencias como para la creación de las mismas.

- **TFG/dist/config**: contiene toda la información relativa a la conexión de la capa de presentación con la capa de datos, es decir con la base de datos, así como variables globales utilizados por otros scripts, a continuación, se muestra el fragmento de código encargado de realizar dicha conexión:

```php
<?php
   // Definimos las variables globales para la conexion
   define('DB_SERVER', 'server_host');
   define('DB_USERNAME', 'username');
   define('DB_PASSWORD', 'password');
   define('DB_DATABASE', 'database');
   // Realizamos conexion con la base de datos
   $db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);
   // Establecemos la codificacion de caracteres UTF8
   mysqli_set_charset($db,"utf8");
   // Comprobamos la conexion
    if (!$db) {
        die("Connection failed: " . mysqli_connect_error());
    }
?>

```

Figura 21: Fragmento de código encargado de realizar la conexión con la BDD.

- **TFG/dist/controlador**: contiene todos los scripts php encargados de hacer las consultas correspondientes a la base de datos e incrustar el código html generado por el servidor en las páginas visibles por el usuario. Esta carpeta contiene las siguientes subcarpetas: android, correo, direccion y recepcion.

  La carpeta android contiene los scripts necesarios para la interacción con Firebase. Estos scripts se encargan de mandar una notificación al dispositivo Android del empleado asignado para solucionar la incidencia.

En la carpeta correo podemos encontrar todos los scripts relacionados con la creación de los correos de bienvenida y de modificación de contraseña en caso de que el usuario así lo haya decidido.

Las carpetas direccion y recepcion contienen todos los scripts necesarios para la interacción entre la base de datos y las páginas visibles por los directores/administradores y los recepcionistas respectivamente. El siguiente fragmento de código es un ejemplo en el que se realiza una consulta a la base de datos desde la página de recepcion para obtener todas las habitaciones del hotel en el que trabaja el recepcionista:

```php
<?php
  include($_SERVER["DOCUMENT_ROOT"] . "/TFG/dist/controlador/recepcion/sessions_empleado.php");
  $idEmpleado = $_SESSION['idEmpleado'];
  $idHotel = $_SESSION['idHotel'];
  // Realizamos la consulta para saber que habitaciones tiene el hotel
  $sql = "SELECT nombre FROM habitacion WHERE idHotel='$idHotel' AND activo='1'";
  $result = $db->query($sql);
  // Si la consulta devuelve habitaciones las enseñamos al usuario,
  // sino mostramos mensaje de que no hay habitaciones
  if ($result->num_rows > 0) {
      while($row = $result->fetch_assoc()) {
        echo "<option>".$row['nombre']."</option>";
      }
  } else {
      echo "<option>El hotel no dispone de habitaciones.</option>";
  }
  // ...
?>
```

Figura 22: Fragmento de código que muestra las habitaciones de un hotel.

Una vez explicada la estructura de la página, se detallan algunas de las funcionalidades más importantes de la página web, así como el código encargado de levarlas a cabo:

- **Crear una nueva incidencia:** es una de las principales funcionalidades de la página de recepción. Rellenando el formulario de incidencia (título, descripción, operario, máquina, tipo, prioridad, habitación y zona), la página web se encarga de generar la incidencia, guardarla en la base de datos y enviar una notificación al operario encargado de resolverla.

La figura 23 demuestra cómo se manda la notificación al servidor de Firebase para que este compruebe el token del dispositivo que va a recibir la notificación existe y es válido. Si todo es correcto, Firebase manda la notificación al dispositivo como vemos reflejado en la figura 24.

```php
<?php
    function enviar_notificacion($idEmpleado_realizador,$db,$titulo,$idEmpleado_creador,$modo){
        define( 'API_ACCESS_KEY', 'API_ACCES_KEY' );
        // Obtenemos el token del dispositivo del empleado asignado para realizar la incidencia
        $registrationIds = get_empleado_token($idEmpleado_realizador, $db);

        // ...

        // Definimos el mensaje que va a recibir el usuario
        $msg = array(
          'body'  => $texto.$titulo,
          'title' => idCreador_a_creador($idEmpleado_creador,$db),
          'icon'  => 'myIcon',
          'sound' => 'mySound'
        );
        // Preparamos el mensaje con el token del usuario
        $fields = array(
          'to'    => $registrationIds,
          'notification'  => $msg
        );
        // Añadimos cabezeras
        $headers = array(
          'Authorization: key=' . API_ACCESS_KEY,
          'Content-Type: application/json'
        );
        // Enviamos notificación al servidor de FireBase
        $ch = curl_init();
        curl_setopt( $ch,CURLOPT_URL, 'https://fcm.googleapis.com/fcm/send' );
        curl_setopt( $ch,CURLOPT_POST, true );
        curl_setopt( $ch,CURLOPT_HTTPHEADER, $headers );
        curl_setopt( $ch,CURLOPT_RETURNTRANSFER, true );
        curl_setopt( $ch,CURLOPT_SSL_VERIFYPEER, false );
        curl_setopt( $ch,CURLOPT_POSTFIELDS, json_encode( $fields ) );
        $result = curl_exec($ch );
        curl_close( $ch );
    }
?>
```

Figura 23: Código para notificar una incidencia nueva.

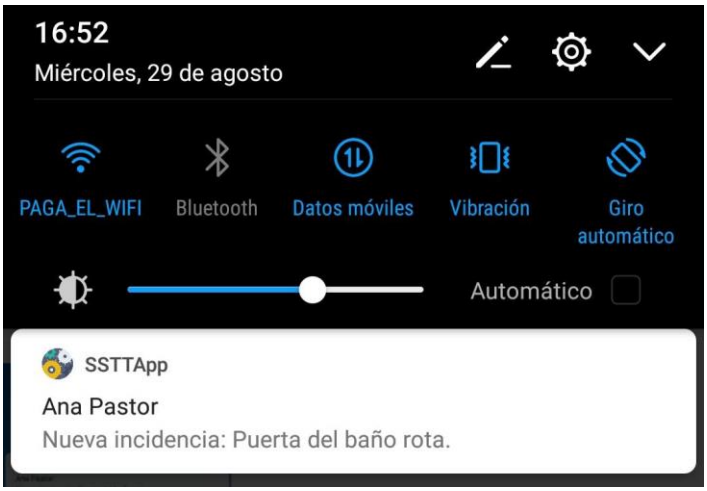

Figura 24: Notificación generada por Android.

- **Generar notificación** al crear/finalizar una incidencia: cuando el operario asignado para realizar la incidencia termina o genera alguna incidencia nueva, el sistema notifica la acción al recepcionista mediante el fragmento de JavaScript de la Figura 26. La notificación generada en el navegador del recepcionista aparece en siguiente figura:

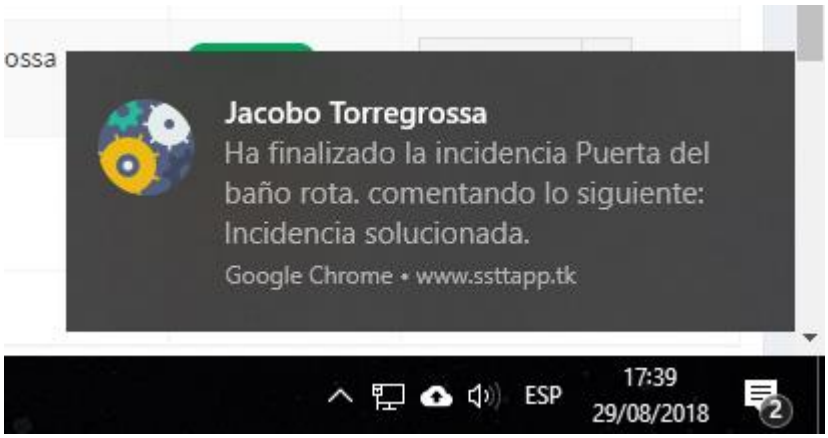

Figura 25: Notificación generada por la página web de recepción.

```
27        // Cuando el dispositivo recibe la notificación la mostramos por pantalla
28        messaging.onMessage(function(payload) {
29          console.log('Mensaje receibido. ', payload);
30          var notificationTitle = payload.notification.title;
31          var notificationOptions = {
32            body: payload.notification.body,
33            icon: payload.notification.icon,
34            image:  payload.notification.image
35          };
36          var notification = new Notification(notificationTitle,notificationOptions);
37          location.reload();
38        });
39        // Funcion que guarda el token del dispositivo del recepcionista en la base de datos
40        function sendTokenToServer(currentToken){
41          var parametros = {
42              "tokenFirebase" : currentToken
43            };
44            $.ajax({
45              url: '../../dist/controlador/recepcion/actualizar_token.php',
46              data: parametros,
47              type: 'POST',
48              success: function(result) {
49                console.log(result);
50              }
51            });
52        }
53        // Funcion que obtiene el token del dispositivo.
54        function get_Token(argument) {
55          messaging.getToken().then(function(currentToken) {
56            if (currentToken) {
57              sendTokenToServer(currentToken);
58            } else {
59              console.log('El token no esta disponible.');
60              setTokenSentToServer(false);
61            }
62          }).catch(function(err) {
63            console.log('Ha ocurrido un error mientras se obtenia el token. ', err);
64            setTokenSentToServer(false);
65          });
66        }
```

Figura 26: Notificación en la página de recepción.

Para finalizar este apartado, se explican algunas de las librerías que se han utilizado para el desarrollo de la plataforma web:

- Datepicker: proporciona un calendario para que el usuario pueda elegir una fecha.
- ChartJS: librería utilizada para las gráficas del panel de administrador.
- DataTable: plugin de jQuery que agrega funcionalidades avanzadas a una tabla como: buscar por texto, ordenar por columna, filtrar por páginas, etc.

5.3.3. API REST

El objetivo de la API es acceder a los datos de la capa de datos desde la aplicación Android. Desde los distintos dispositivos se realizan peticiones http para obtener estos datos.

Para el desarrollo de la API se ha utilizado el *framework* Slim. Las funciones han sido diseñadas en el apartado 5.2.1 y a continuación se explicará el funcionamiento de una de ellas: obtener la información de un empleado. La figura 28 ilustra el funcionamiento de la petición: el dispositivo Android realiza una petición GET a la ruta /empleado/{idEmpleado}. El parámetro {idEmpleado} es el identificador del empleado en la base de datos. Gracias a esta petición, la API realizaría la consulta pertinente y devolvería la siguiente respuesta en JSON:


```
{ ⊟
  "empleados":[ ⊟
    { ⊟
      "idEmpleado":"2",
      "nombre":"Ana Pastor",
      "email":"ana@ana.com",
      "rol":"recepcion",
      "imagen":"/TFG/dist/img/74f786d76705f7578498a8836d86e8eauser3-128x128.jpg",
      "tokenFirebase":"TOKEN_FIREBASE_DISPOSITIVO",
      "idHotel":"1"
    }
  ]
}
```


Figura 27: Respuesta del servidor en JSON.

Esta respuesta es interpretada por el dispositivo que ya esta en disposicion de mostrar los datos al usuario final. Esta petición es invisible para el usuario.

```
// Definicion de la ruta con el parámetro {id} del empleado
// Esta funcion devuelve toda la información del usuario con idEmpleado={id}
$app->get('/empleado/{id}', function(Request $request, Response $response){
  // Obtenemos el parametro pasado por http
  $idEmpleado = $request->getAttribute('id');
  $consulta = "SELECT idEmpleado,nombre,email,rol,imagen,tokenFirebase,idHotel
               FROM empleado
               WHERE idEmpleado=$idEmpleado AND activo=1";
  try{
    // Instanciamos la base de datos
    $db = new db();
    // Conexion con la base de datos
    $db = $db->conectar();
    // Realizamos la consulta
    $ejecutar = $db->query($consulta);
    $empleado = $ejecutar->fetchAll(PDO::FETCH_OBJ);
    $db = null;
    // Exportamos el resultado a JSON
    echo '{ "empleados" : '.json_encode($empleado, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES).'}';
  }catch(PDOException $e){
    echo '{"error": {"text": '.$e->getMessage().'}}';
  }
});
```

Figura 28: Función de la API que devuelve la información de un empleado.

Durante el desarrollo de la API todas y cada una de las funciones fueron probadas mediante una herramienta gratuita llamada RestEasy 1.2. Es una extensión de Google Chrome orientada a desarrolladores que permite realizar peticiones y procesar las respuestas de una API.

5.3.4. Aplicación Android

La aplicación de Android se ha desarrollado de manera nativa con el IDE Android Studio. Como ya se ha explicado en el apartado 5.3.1, para la programación de la aplicación se ha utilizado Java para las funcionalidades de la aplicación y XML para el diseño de los *Layouts* (pantallas). La estructura del proyecto queda de la siguiente manera:

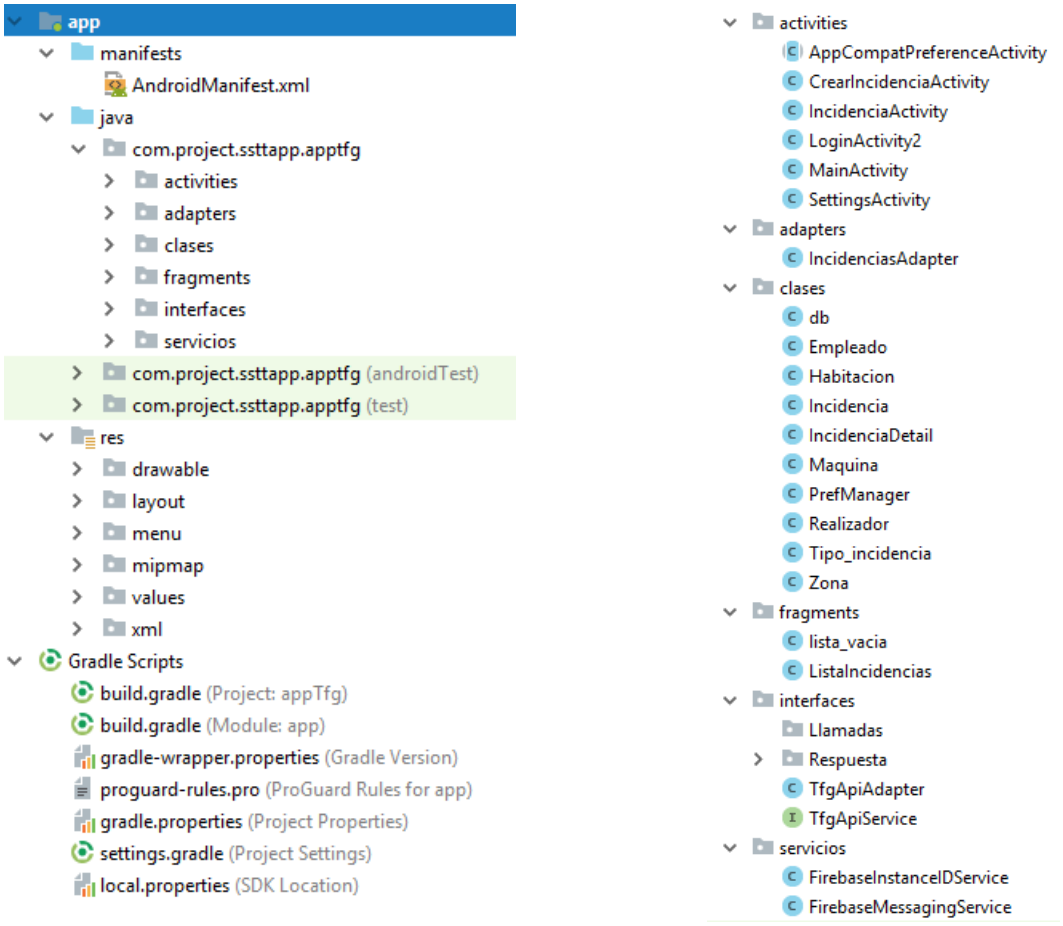

Figura 29: Estructura del proyecto Android SSTTApp.

Como se puede apreciar, la aplicación está dividida en tres carpetas: java, res y manifest. La carpeta java contendrá todo el código fuente de la aplicación, clases auxiliares, interfaces, servicios, etc. La carpeta res contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, *layouts*, y cadenas de texto entre otros elementos. Para

finalizar, la carpeta manifest contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, etc), sus componentes (pantallas, servicios, etc), o los permisos necesarios para su ejecución.

Se ha considerado importante explicar algunos de los ficheros más relevantes de la aplicación:

- **FirebirdInstanceIDService**: es un servicio que extiende de FirebaseMessagingService. Cuando se inicia la app por primera vez, el SDK de Firebase genera un token de registro para la instancia de la app mediante esta clase de java. El token se registra tanto en las preferencias de la aplicación como en el servidor. Gracias a él se podrán enviar notificaciones PUSH al empleado para avisarle de que se le ha asignado una nueva incidencia.
- **FirebaseMessagingService**: es el servicio encargado de crear una nueva notificación cuando el empleado tiene una nueva incidencia asignada. Cuando se crea una incidencia, se envía una notificación al token del dispositivo del usuario y este servicio se encarga de generar la notificación correspondiente.
- **TfgApiAdapter**: clase encargada de generar las peticiones http a la API mediante la librería Retrofit. Esta librería además de generar estas peticiones permite parsear las respuestas JSON que devuelve la API a objetos java directamente.
- **TfgApiService:** interficie que contiene todas las peticiones http que se van a ejecutar desde la clase TfgApiService. En la figura 30 se puede apreciar un fragmento de código que refleja estas peticiones.

```
1  public interface TfgApiService {
2
3      @FormUrlEncoded
4      @POST("login")
5      Call<LoginResponse> login(
6              @Field("email") String email, @Field("password") String password
7      );
8
9      @GET("empleado/{idEmpleado}")
10     Call<EmpleadoResponse> getEmpleado(
11             @Path("idEmpleado") Integer idEmpleado
12     );
13
14     // Otras peticiones ...
15
16 }
```

Figura 30: Fragmento de código de la clase TfgApiService.

- **IncidenciasAdapter**: es un adaptador personalizado con el que se rellena la lista de incidencias de la app y extiende de la clase RecyclerView.Adapter. Se ha destacado esta clase porque, pasándole por parámetro una lista de incidencias en su método constructor, se encarga de mostrarlas en forma de lista por la pantalla del dispositivo. En cada ítem de la lista, se mostrará la foto de empleado creador, el título, la descripción, el tiempo que ha pasado desde que se creó, la habitación y la prioridad.
- **Incidencia**: clase que modela una incidencia y contiene los campos necesarios para su visualización: titulo, descripción, empleado creador, fecha de inicio, etc.

Se han utilizado algunas de las librerías externas para el desarrollo de la aplicación, a continuación, se explican las más importantes:

- Picasso: librería encargada de descargar y guardar imágenes en la caché del dispositivo.
- Retrofit: librería utilizada para realizar las peticiones http a la API REST.
- PhotoView: librería que permite realizar un efecto de zoom sobre una imagen.

OLUTÍPAC 6

Resultados

Ya implementado el proyecto en base a los requisitos, necesidades y funcionalidades requeridas en el apartado 5.1, este apartado muestra algunas de las páginas de la aplicación.

6.1. Plataforma web

Página de Login: la figura 31 muestra la primera toma de contacto del usuario con la aplicación web. Al escribir la URL de la página en el navegador se encuentra con este formulario de Login.

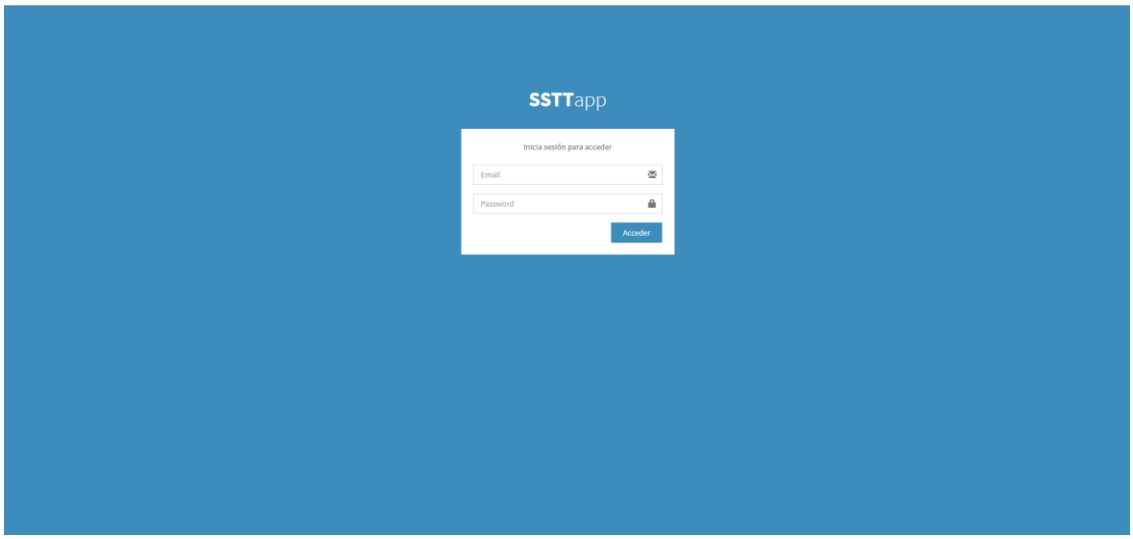

Figura 31: Página de Login.

Página de recepción: es la página que usarán mayoritariamente los recepcionistas de los establecimientos hoteleros. La página tiene un panel de navegación lateral donde se puede crear nuevas incidencias o filtrarlas por el estado además de consultar la disponibilidad de las habitaciones.

Las incidencias pueden ser ordenadas por columna, filtradas por número de incidencias y se pone a disposición del usuario un buscador para que realice las consultas necesarias.

Para cada incidencia, el recepcionista puede ver los siguientes campos: fecha de creación, creador de la incidencia, descripción, habitación, prioridad, operario asignado, estado de la incidencia y un botón desplegable con el que puede visualizar toda la información de la incidencia incluida la fotografía o puede modificar la información siempre y cuando la incidencia este pendiente de ser realizada.

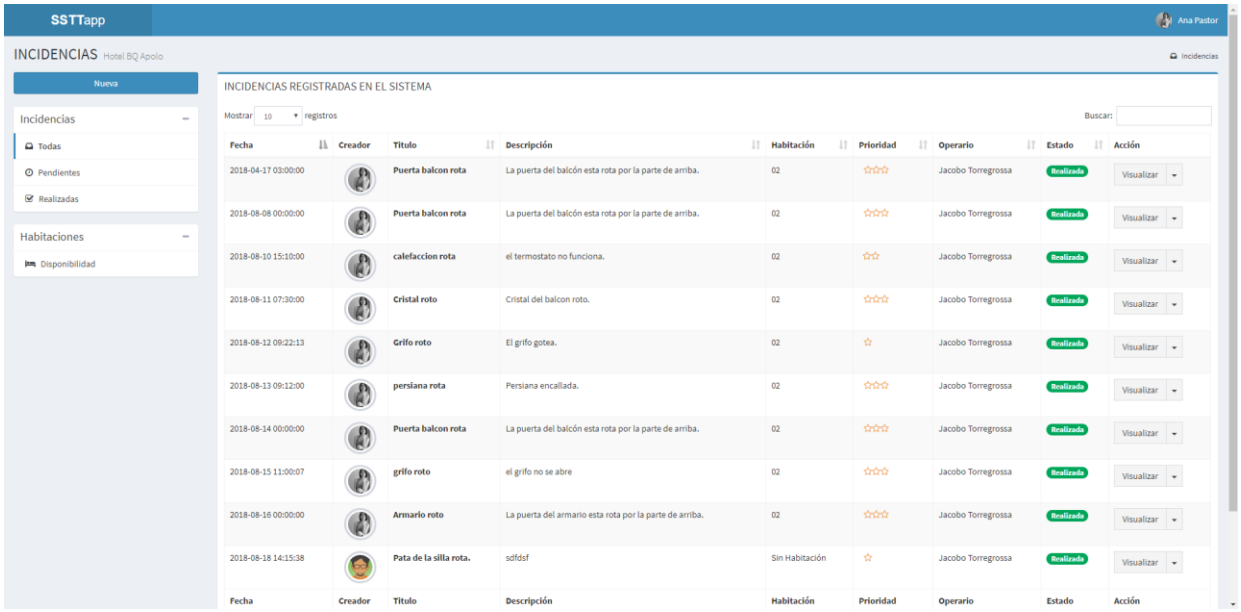

Figura 32: Página de recepción.

Nueva incidencia: página para crear una nueva incidencia y asignarla a algún operario/camarera de piso del establecimiento. Cuando el usuario desea notificar, el sistema genera una notificación en la aplicación Android SSTTApp para el encargado en asignado.

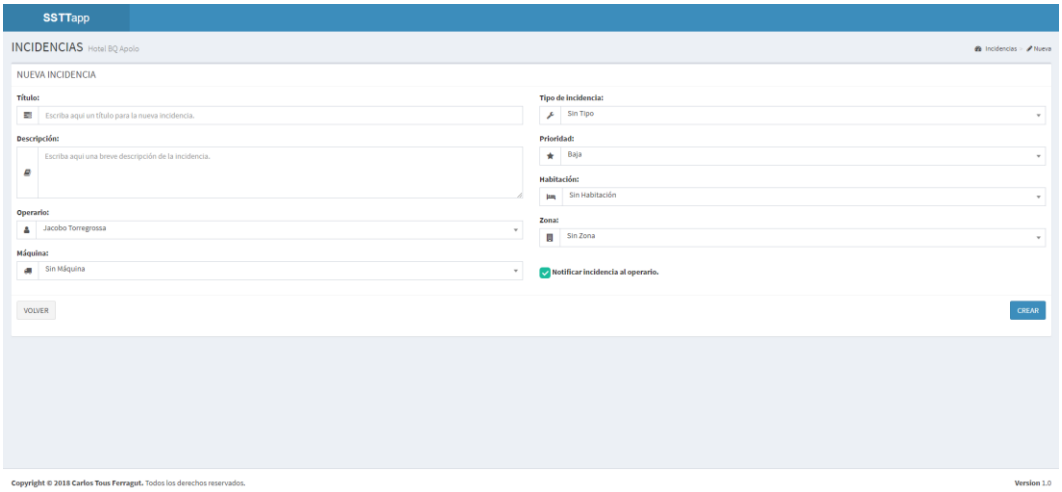

Figura 33: Página para crear una nueva incidencia

Modificación de los datos de usuario: los recepcionistas pueden modificar los datos de su cuenta mediante la página de la figura 34.

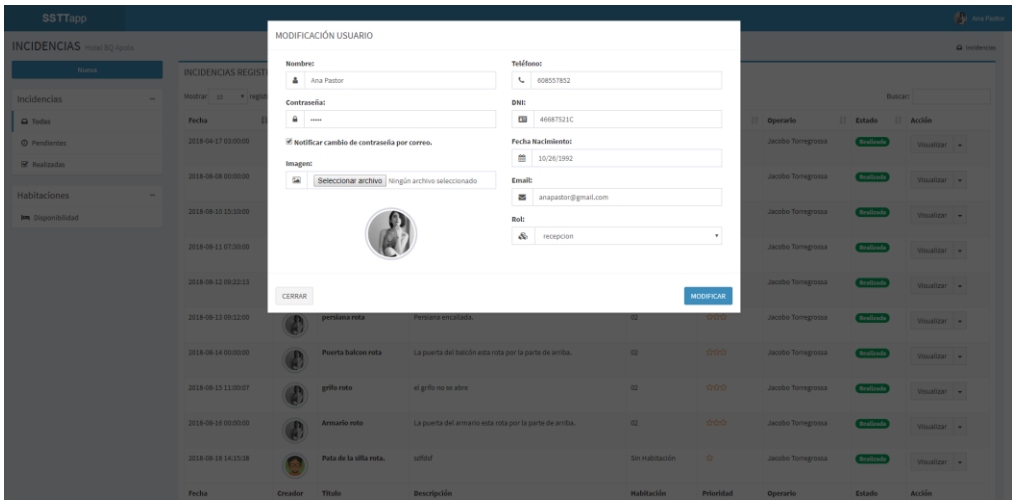

Figura 34: Página para modificar la información de un usuario.

Visualización de una incidencia: el recepcionista puede consultar la información de una incidencia como se observa en la figura 35.

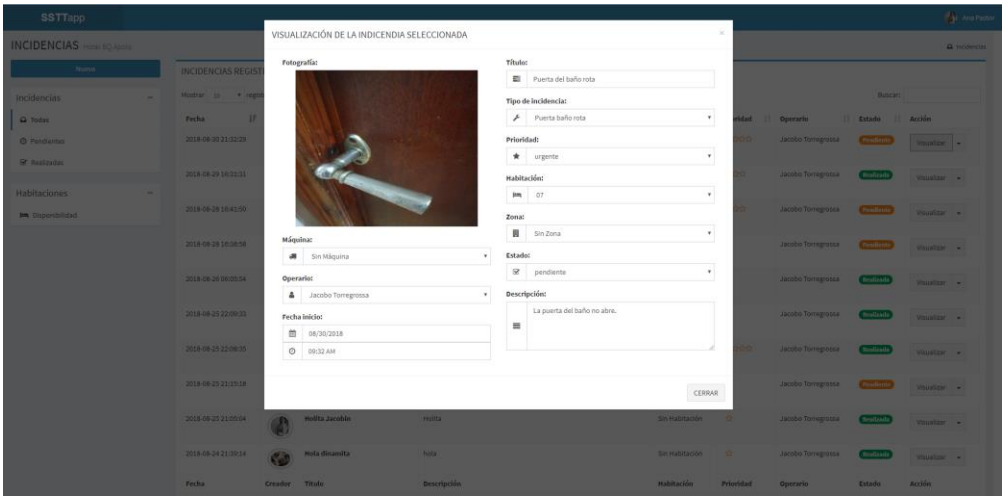

Figura 35: Página para la visualización de una incidencia.

Gestión de hoteles: los administradores/directores disponen de un panel en el que le aparecen todos sus hoteles. Los administradores, tienen privilegios para crear nuevos o eliminar ya creados:

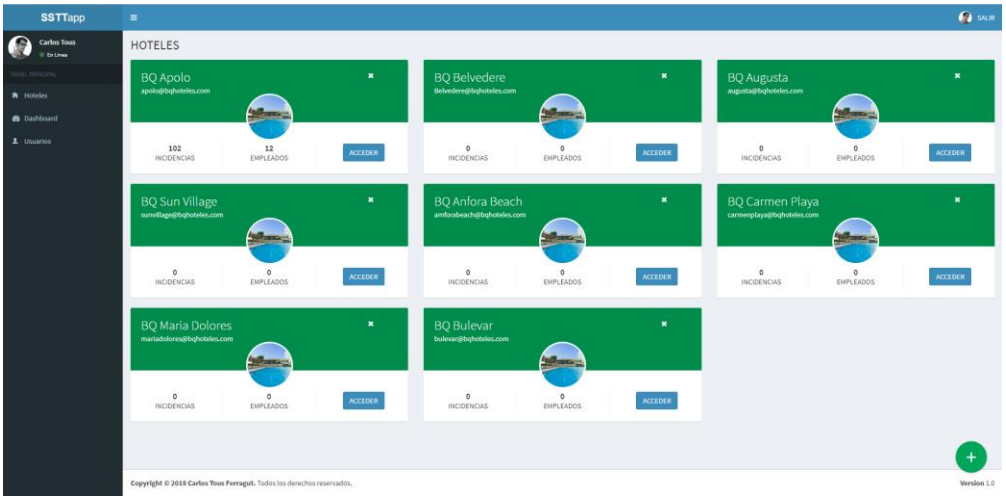

Figura 36: Página para la gestión de hoteles.

Dashboard: muestra la información relativa al establecimiento, así como una serie de gráficas de interés para que los directores puedan tomar decisiones en base a las estadísticas generadas por la aplicación.

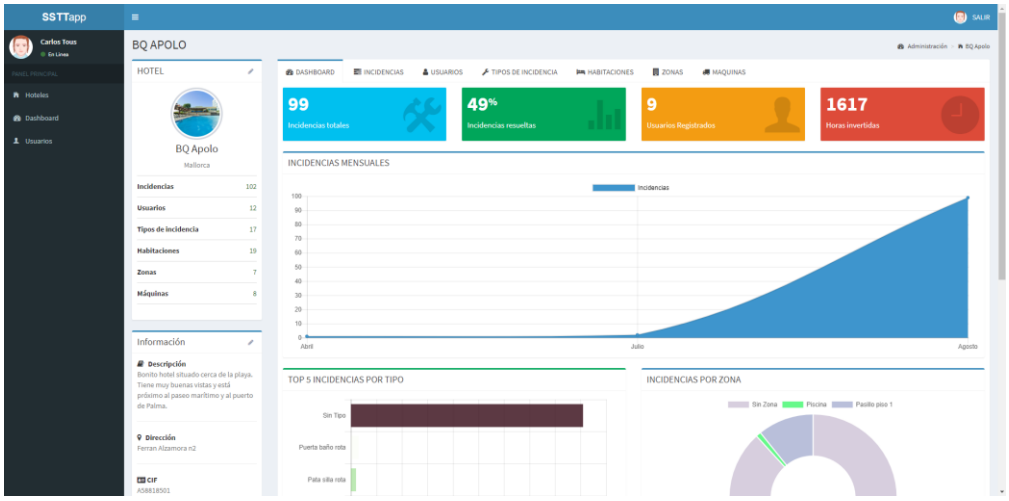

Figura 37: Dashboard de un establecimiento hotelero.

Administración de empleados de un hotel: el director puede gestionar todos sus empleados a través de esta página. Puede modificar, eliminar o crear nuevos empleados.

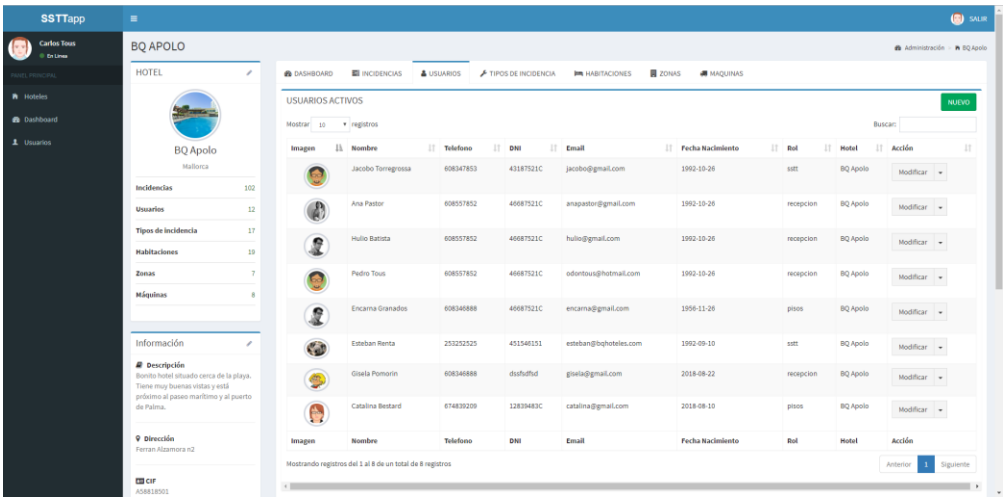

Figura 38: Gestión de empleados de un hotel.

Gestión de habitaciones: página con la que se pueden gestionar todas las habitaciones del establecimiento. El director puede crear/modificar/eliminar habitaciones de su hotel.

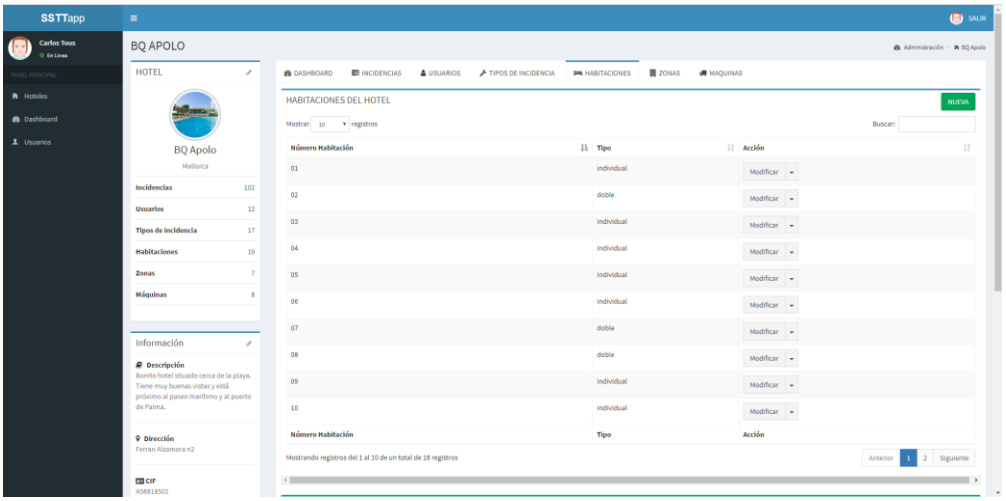

Figura 39: Gestión de habitaciones de un hotel.

Página de error: cuando en el servidor web se genera un error 404 o un error 405, el sistema muestra una pantalla de error como la siguiente:

Figura 40: Página de error 404.

6.2. Aplicación Android

Uno de los requisitos no funcionales del proyecto era subir la aplicación a Google Play de manera que todos los empleados pudieran descargarla. El resultado queda ilustrado en la siguiente figura:

Figura 41: Aplicación SSTTApp en Google Play.

A continuación, se muestran las pantallas más importantes de la aplicación SSTTApp con las que se puede apreciar el diseño *material design* y la facilidad de uso que presenta.

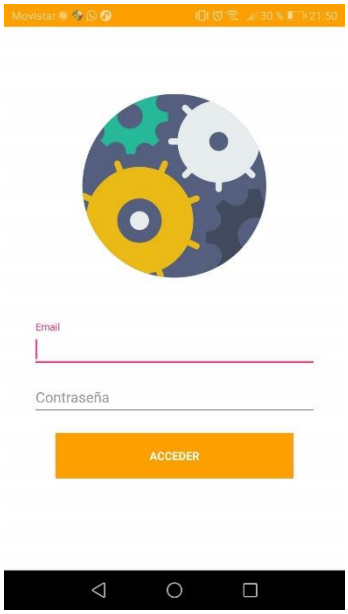

Figura 42: Pantalla Login.

Figura 43: Pantalla incidencias.

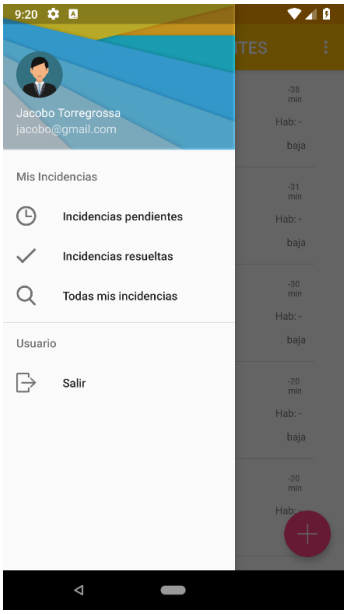

Figura 44: Navigation drawer.

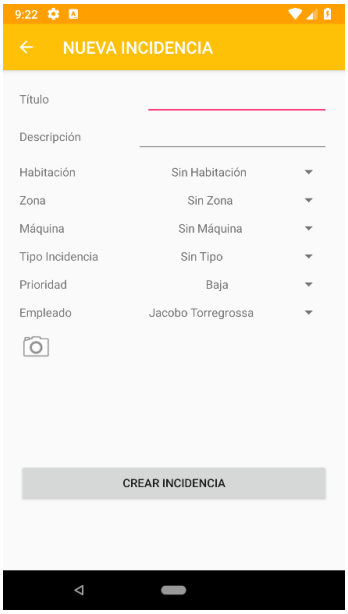

Figura 45: Nueva incidencia.

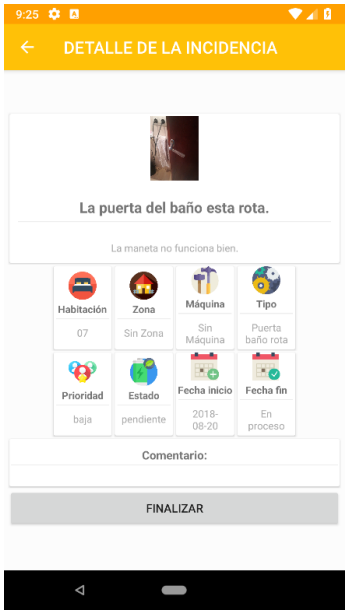

Figura 46: Detalle de una incidencia.

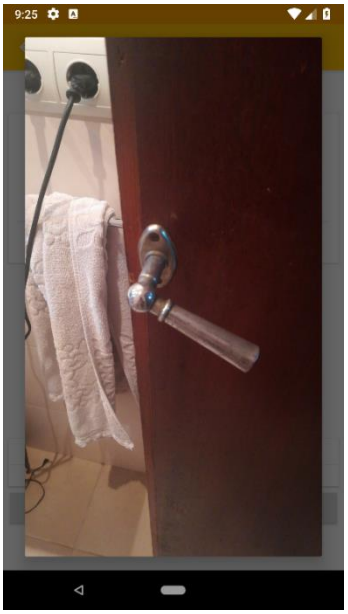

Figura 47: Zoom fotografía.

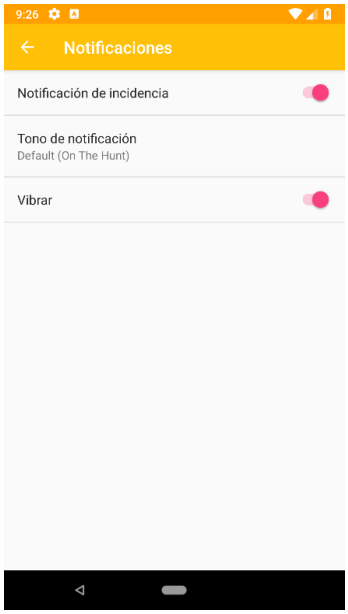

Figura 48: Preferencias de notificaciones.

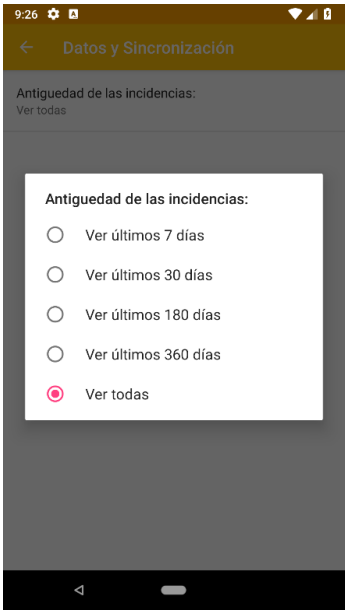

Figura 49: Preferencias de datos

OLUTÍPAC 7

Conclusiones

En el presente documento se ha descrito la implementación de un proyecto para la gestión integral de las incidencias que se generan en los establecimientos hoteleros.

Esta solución de software pretendía cubrir la necesidad de informatizar todos los partes de incidencias que se generaban de manera tradicional durante la jornada laboral en la cadena hotelera con el fin de mejorar aspectos como la eficiencia, productividad y la comunicación entre los distintos departamentos.

A modo de resumen, los objetivos que se marcaron en la fase inicial del proyecto eran los siguientes:

- Mejorar la comunicación y productividad entre los empleados de la cadena.
- Controlar el estado de las incidencias y las habitaciones en tiempo real.
- Obtener datos estadísticos para la ayuda en la toma de decisiones.

En base a estos objetivos, se decidió que la manera óptima de cumplirlos era mediante una página web para gestionar las incidencias y una aplicación Android para la generación de nuevos partes de incidencias.

Una vez desarrolladas sendas plataformas de software, se realizaron diferentes pruebas a fin de comprobar su correcto funcionamiento. Al llegar a la fase final del proyecto y finalizar las pruebas de manera satisfactoria, se puede concluir que la solución propuesta cumple con los requisitos y objetivos marcados al inicio de este.

Por lo tanto y en consecuencia, el plan implementado también solventa y cubre la necesidad de informatizar y agilizar el sistema de gestión de incidencias en el seno de una cadena hotelera.

Personalmente me interesa mucho aprender nuevas herramientas y tecnologías para la construcción de software y con este proyecto he tenido la ocasión de utilizar y consolidar muchas de ellas (Firebase, Slim, Bootstrap, etc). En relación a lo aprendido durante toda mi etapa universitaria, puedo decir que he adquirido los conocimientos básicos y necesarios para desarrollar una herramienta de software que cubra una necesidad en la sociedad.

Por otro lado, los conocimientos adquiridos durante el desarrollo del proyecto se pueden resumir en los siguientes:

- Utilizar *frameworks* de código abierto para el desarrollo de software seleccionando las funcionalidades que mejor se adaptan a las necesidades del proyecto (Slim).
- Consolidar los conocimientos relacionados con la comunicación entre plataformas mediante protocolos HTTP y cadenas JSON.
- Preparar, inicializar y gestionar un servidor de manera integral.
- Mejorar el desarrollo y la implementación de aplicaciones basadas en la plataforma Android.

Para finalizar, concluir y recalcar que desarrollar este proyecto y, en definitiva, este Trabajo Final de Grado ha sido una experiencia agradable, positiva y educativa de la cual he aprendido mucho y me servirá en el futuro para mejorar, no solo como estudiante e ingeniero sino como persona.

Bibliografía

[1] Sanromán (2018). La importancia de las TIC en el turismo, 01-03.

https://cambiatucurso.es/blog/2018/02/07/la-importancia-de-las-tic-en-el-turismo/

[2] Ditrendia (2017). Informe Mobile en España y en el mundo 2017.

[3] ComputerHoy (2018). La guerra de los smartphones en cifras.

https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-271447

[4] Eisihotel (2018). Software en nube para digitalizar y controlar un hotel.

http://www.eisihotel.com/

[5] Gimhotel (2018). Gestión integral del mantenimiento de un hotel.

https://www.gimhotel.com/

[6] Roberth G. Figueroa, Camilo J. Solís. Metodologías tradicionales vs. metodologías ágiles.

[7]. EvaluandoSoftware (2018). ¿Qué es desarrollo de software ágil? 01-04, 08-12

http://www.evaluandosoftware.com/desarrollo-software-agil/

[8]. Eduardo Martinez (2013). Agile y Scrum.

https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

[9]. Jonathan Ordoñez (2018). ¿Qué es una API REST?

https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/

[10]. Wikipedia (2018). Programación por capas.

https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

[11]. Amazon (2018). Amazon Elástic Compute Cloud.

https://aws.amazon.com/es/ec2/

[12]. BlueHosting (2016). Apache vs. Nginx.

https://docs.bluehosting.cl/tutoriales/servidores/apache-versus-nginx--cual-es-el-servidor-web-ideal.html

[13]. Netcraft (2018). Web server survey.

https://news.netcraft.com/archives/2018/07/19/july-2018-web-server-survey.html

[14]. Php.net (2018). General Information of PHP.

http://us.php.net/manual/en/faq.general.php

[15]. Apache (2018). Módulos de MultiProcesamiento.

https://httpd.apache.org/docs/2.4/es/mpm.html

[16] Jose Mª Baquero (2015). PHP vs. ASP. 07-10

https://www.arsys.es/blog/programacion/php-o-asp-que-elijo-para-mi-web/

[17] Universitat Oberta de Catalunya (2018). Diseño de interfaces. 01-07

http://multimedia.uoc.edu/blogs/dii/es/que-es-una-interficie/

[18] Wikipedia (2018). HTML.

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

[19] Wikipedia (2018). Hoja de estilos en cascada.

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

[20] Bootstrap (2018). Bootstrap.

https://getbootstrap.com/

[21] Mozilla (2018). JavaScript.

https://developer.mozilla.org/es/docs/Web/JavaScript

[22] Wikipedia (2018). jQuery.

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

[23] Wikipedia (2018). AJAX.

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

[24] Php.net (2018). ¿Qué es PHP?. 01-02

http://www.php.net/manual/es/intro-whatis.php

[25] Slim (2018). Slim.

https://www.slimframework.com/

[26] Wikipedia (2018). JavaScript Object Notation.

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

[27] Developers Mysql (2018). Manual de referencia. What is MySQL?.

https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[28] Wikipedia (2018). Firebase.

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

[29] Developers Android (2018). Conoce Android Studio. 01-02

https://developer.android.com/studio/intro/?hl=es-419

[30] Apache Friends (2018). ¿Qué es XAMPP?.

https://www.apachefriends.org/es/index.html