TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

"Aplicación para la gestión de comandas de locales para clientes y empleados"

Curso 2016/2017

Alumno/a:

Carlos Garrido Vergara

Director/es:

José Antonio Piedra Fernández

Autor	Carlos Garrido Vergara.
Titulación	Grado en Ingeniería Informática.
Tutor	José Antonio Piedra Fernández.
Departamento	Departamento de Informática.
Modalidad	Técnico.
Palabras clave	Aplicación Móvil, Restaurante, Gestión.

Figuras

1. Introducción.

1.1. Análisis del problema.

Haciendo un estudio de mercado, se ha detectado que existen, por una parte, diversas páginas y aplicaciones para localizar restaurantes en una zona, ver sus críticas o reservar mesa (pensadas para el cliente), y por otra parte aplicaciones para gestionar las comandas de los restaurantes (pensadas para el restaurante), pero se ha observado una escasez en aplicaciones o servicios web que dan soporte a la total gestión de los restaurantes e interacción con el cliente (pensadas para el cliente y el restaurante), por lo que se ha decidido desarrollar un servicio que permita gestionar las actividades diarias de un restaurante, así como la interacción con el cliente.

Las metas de este Servicio Web son poder gestionar las actividades diarias de un restaurante y mejorar el servicio prestado al cliente así como mejorar la experiencia del cliente.

1.2. Objetivos.

Se han establecido una serie de objetivos para las distintas metas, que involucran a distintos usuarios de la aplicación.

1.2.1. Para gestionar las actividades: (Administrador)

- Probar la aplicación en una versión Demo.
- Añadir, leer, modificar y eliminar mesas.
- Añadir, leer, modificar y eliminar espacios del local.
- Añadir, Leer, Modificar y Eliminar (CRUD) productos.
- Añadir, leer, modificar y eliminar ingredientes.
- Añadir, leer, modificar y eliminar categorías de productos.
- Añadir, leer, modificar y eliminar menús y platos especiales.
- Añadir y eliminar empleados.
- Leer y modificar el perfil de usuario.
- Poder ubicarse en el mapa para que los clientes encuentren el local mediante el buscador de la aplicación.

1.2.2. Para mejorar el servicio prestado al cliente: (Empleado)

- Llevar a cabo peticiones de comandas a cocina.
- Modificar los productos pedidos, pudiendo eliminar ingredientes, añadirlos o cambiarlos por otros según guste el cliente.
- Tener información sobre el estado actual de cada mesa, mejorando así la eficiencia del servicio.
- Ver el estado de las comandas y un historial de comandas por mesas.
- Informar sobre el stock de los productos en tiempo real.
- Gestionar el pago de los comensales, pudiendo dividir la cuenta para beneficiar a los comensales.
- Al finalizar la comida, los datos pasan a almacenarse a una base de datos externa, para mejorar la eficiencia de la aplicación.

1.2.3. Para mejorar la experiencia del cliente (Cliente)

- Ver la información general del restaurante.
- Conocer la carta.
- Buscar restaurantes mediante el buscador de la aplicación en el mapa.
- Escanear la mesa de un local
- Pedir comanda de forma automática.
- Ver el historial de sus comandas.
- Solicitar la asistencia de un camarero.
- Pagar la cuenta de forma automática.
- Valorar el servicio prestado por el restaurante.
- Realizar sugerencias o quejas.

1.3. Metodología y materiales.

1.3.1. Metodología.

La metodología que vamos a desarrollar a lo largo del proyecto va a ser Kanban y para ello vamos a usar la herramienta ProjectLibre.

1.3.2. Materiales.

Tras estudiar distintas herramientas y métodos de desarrollo, se ha estimado conveniente que la aplicación sea una aplicación nativa desarrollada para la plataforma Android,en el lenguaje de desarrollo Java, con el entorno de desarrollo Android Studio 2.2.2 con Sqlite para la base de datos local.

Para la aplicación externa, se realizará una aplicación web Python 3.7 como lenguaje de desarrollo y Django 1.11 como framework de desarrollo.

Para la API se realizará un servicio Rest con Django Rest framework 3.5.3 y postgre para la API RestFul.

El hosting se llevará a cabo mediante la página web Heroku con versión gratuita y base de datos postgre.

1.4. Recursos.

Para llevar a cabo el proyecto se va a usar un equipo MSI modelo MS-7817 con Windows 10 de 64 bits con un procesador Intel Core i5 de 3.2HGz, una memoria RAM de 16.0 GB con tarjeta gráfica Intel HD Graphics 4600 y versión de DirectX 12.

2. Estado del arte.

2.1. Breve resumen.

Con este estudio se desea conocer las potencias y limitaciones de las aplicaciones actualmente desarrolladas y comercializadas en la tienda para móviles de Google Play, con el objeto de descubrir las necesidades del sector que aún no están siendo suplidas o que se pudieran mejorar en cualquiera de sus aspectos y cómo se podría mejorar la experiencia de usuario tanto desde el lado del cliente del servicio como desde el lado del proveedor del servicio.

Muchas de las aplicaciones tienen funciones o desventajas similares por lo que se agruparán y solo se detallarán las que se han considerado más interesantes, más completas o en las que se pueda destacar alguna de sus funcionalidades.

2.2. Aplicaciones del mercado.

Nombre	Versión	Fecha de actualización	Tamaño	Valoración	Descargas
Alf@ Cloud	1.5	20/09/13	257 KB	3,0	Más de 100
Comanda 2000	1.0	30/11/15	86.90 KB	3,0	Más de 500
Comandera Plus	2.0.2	27/07/17	2.35 MB	5,0	Más de 50
DroidPOS	3.0.1	05/04/17	2.69 MB	5,0	Más de 1000
GPI Menu Phone	17.0628	28/06/17	11,98 MB	5,0	Más de 100
Netecnia Camarero	1.1	07/06/15	4.57 MB	N	Más de 100
Netecnia Cliente	1.1	07/06/15	4.57 MB	N	Más de 10
OfiComanda	1.0.34	27/06/17	11.53 MB	4,4	Más de 1000
OrderDroid	v7	1/06/17	2.77 MB	4.9	Más de 1000
Comandas Facile	2.1.10	05/0917	11.57 MB	3,9	Más de 5000
Comandas	1.0	04/0816	1.17 MB	3,4	Más de 5000
Comandero	0.88	27/0916	3.17 MB	2,5	Más de 100

EMC Comanda Electrônica	1.2.0	06/01/16	457 KB	3,0	Más de 500
Mygest	7.0	23/01/17	0.97 MB	5,0	Más de 1000
OrderLan Demo	1.1.1	17/11/15	24.20	4,7	Más de 100
Prima Comanda	2015	23/09/15		5,0	Más de 100
Restaurant Cafe Service	1.2	06/08/16	590 KB	3,0	Más de 50
Resulti - Comanda Electrônica	1.7.4	19/12/16	3.81 MB	3,4	Más de 100
Tablet Mr. Comanda	1.1	02/11/14	706 KB	4,2	Más de 500
Restages Mobile	1.47	19/07/16	5.09 MB	4.9	Más de 100

2.3. Aplicaciones relevantes.

A continuación se va a hacer un énfasis especial en aquellas aplicaciones que poseen alguna característica que las hace tener un valor por encima del resto de aplicaciones.

Alf@ Cloud - Comandas

Aplicación pensada para la gestión de comandas de mesas, con impresión interna de obrador y cocina, modificadores (“con tomate”,”sin cebolla”,”poco hecho”), emisión de ticket o factura y pedidos del cliente desde la propia mesa.

Contras.

- De pago
- Sin demo

Consideraciones.

- Los clientes pueden realizar pedidos desde su propia mesa.
- Video tutoriales para mejorar la experiencia de usuario.

Pruebas.

Accedemos a la aplicación y se nos muestra una pantalla (Figura 1) en la que debemos comprar el software para continuar.

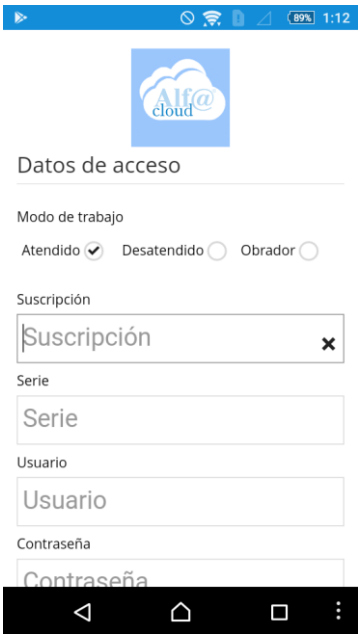


Figura 1. Inicio Alf@ Cloud

Comanda 2000

Aplicación pensada para tomar pedidos y enviarlos por Whatsapp y modificar la carta en precios y productos.

Contras.

- Sólo se pueden enviar comandas mediante Whatsapp.
- Muy simple, sin gestión de mesas.

Consideraciones.

- Las comandas se envían mediante whatsapp.

Pruebas.

Abrimos la aplicación y seleccionamos algunos platos para generar la comanda (Figura 2.a).

A continuación, pulsamos sobre “Pedir” y se nos abre la aplicación de Whatsapp para mandar las comandas (Figura 2.b).

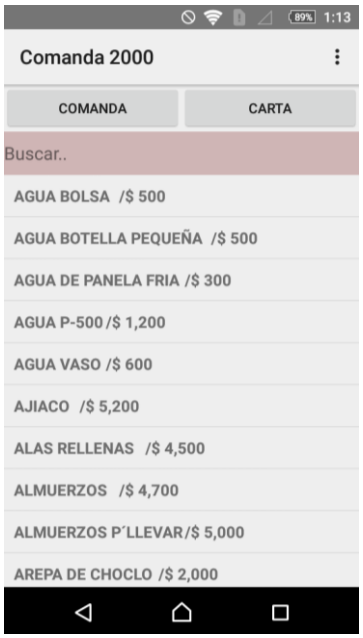


Figura 2.a: Inicio Comanda 2000.

Figura 2.b. Pedir comanda.

Comandera Plus

Permite tomar comandas en mesas y terraza, se pueden hacer pedidos, especificar variaciones de un artículo, mandar mensajes a cocina u obrador, hacer pre-ticket, hacer cobros parciales de cuentas y gestionar salones y terrazas.

Contras.

- Se necesita tener instalado un módulo de respaldo específico en el TPV.

Consideraciones.

- Permite hacer cobros parciales.

Pruebas.

Abrimos la aplicación y se nos pide la dirección de puerto asociada al módulo TPV (Figura 3), por lo que no podemos seguir probando la apliación.

Figura 3: Inicio Comandera Plus.

DroidPOS demo comandero

Permite visualizar evento de pedidos en tiempo real, atención centrada en el cliente, control de ventas por mesa, visualización de mesas ocupadas y libres, agregar comensales por mesa, ver lista de categoría y productos, agregar atributos por producto, visualizar el total a pagar por mesa o por comensal, múltiples destinos, cocinas o bares, agregar notas a destinos, agregar notas a productos, Idiomas Español, Ingles y Portugues.

Contras.

- De pago.
- Adición de pedidos a la comanda algo complejo.
- Al ser una demo no se sabe la capacidad real de la aplicación.

Consideraciones.

- Aplicación en varios idiomas.

Pruebas.

Abrimos la aplicación y se nos muestra un mensaje de error (Figura 4) por lo que no podemos seguir probando la aplicación.

Figura 4:Inicio DroidPOS

GPI Menu Phone

Permite realizar y gestionar comandas, tomar nota a los clientes sin necesidad de desplazarse hasta cocina, las comandas se imprimen de forma automática, respeta las ubicaciones reales del establecimiento.

Contras:

- Pesa demasiado.
- Diversos errores o posibles mejoras, como poder seleccionar cantidades, no poder seleccionar determinadas opciones a la vez.
- Interfaz cerrada y poco flexible.
- La aplicación se cierra inesperadamente, sin permitir su utilización.

Consideraciones:

- Permite personalizar la ubicación de las mesas para que se ajuste a las ubicaciones reales del establecimiento.

Pruebas.

Abrimos la aplicación y se nos muestra la pantalla principal (Figura 5.a) , al pulsar sobre cualquier botón se cierra la aplicación (Figura 5.b).

Figura 5.a: Inicio GPI Menu Phone

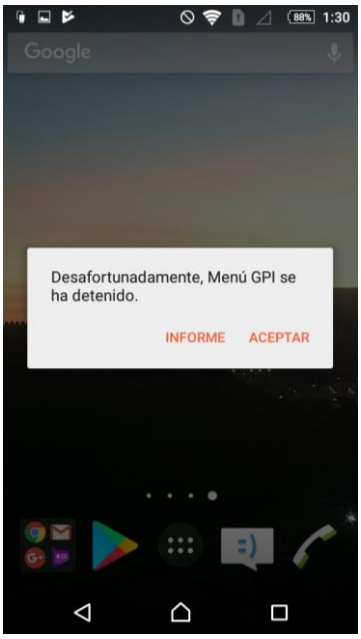


Figura 5.b: Error de aplicación.

Netecnia - Restaurante

Permite explorar la carta, crear pedidos, escanear los pedidos de los clientes y consultar pedidos anteriores.

Contras:

- Necesita dos aplicaciones distintas.
- Poco personalizable.
- Se necesitan aplicaciones de terceros.

Consideraciones:

- Tiene función de escáner para confirmar los pedidos de los clientes.

Pruebas.

Abrimos la aplicación y pulsamos sobre “Comenzar” (Figura 6.a).

A continuación pulsamos sobre alguna de las secciones (Figura 6.b), seleccionamos un producto (Figura 6.c) y la cantidad (Figura 6.d).

Figura 6.a: Inicio netecnia

Figura 6.b:Selección de categoría

Figura 6.c: Selección de producto

Figura 6.d:Selección de cantidad

Por último, pulsamos sobre “Pedido” (Figura 6.e) y en “Confirmar” (Figura 6.f) y pulsamos “Si” en la pantalla emergente.

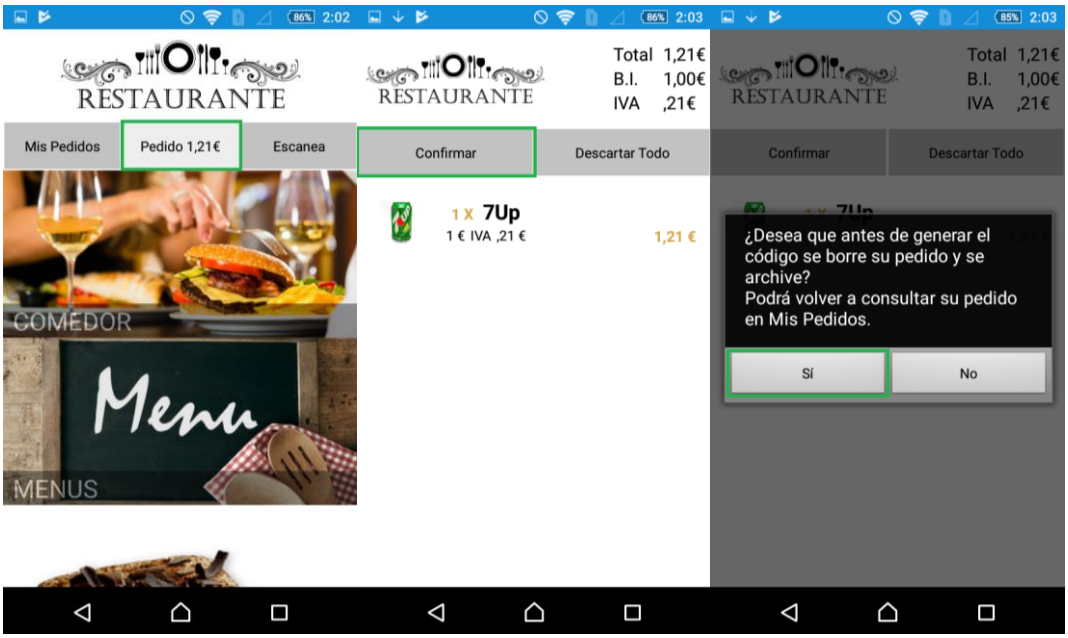


Figura 6.e: Pedido.

Figura 6.f: Confirmar

Figura 6.g: Aceptar.

Se creará un código QR con el pedido realizado (Figura 6.h).

Figura 6.h: Código QR

Ahora, abrimos la aplicación de camareros y pulsamos sobre comenzar (Figura 6.i).

Figura 6.i: Comenzar.

A continuación pulsamos sobre “Escanear” (Figura 6.j) y escaneamos el código QR creado por el cliente (Figura 6.k).

Figura 6.j: Escanear Figura 6.k: Escaner

Nos aparecerá el pedido del cliente, introducimos la mesa, pulsamos en “Confirmar” (Figura 6.l), confirmamos la acción (Figura 6.m) , y pulsamos “Si” a la ventana emergente (Figura 6.n).

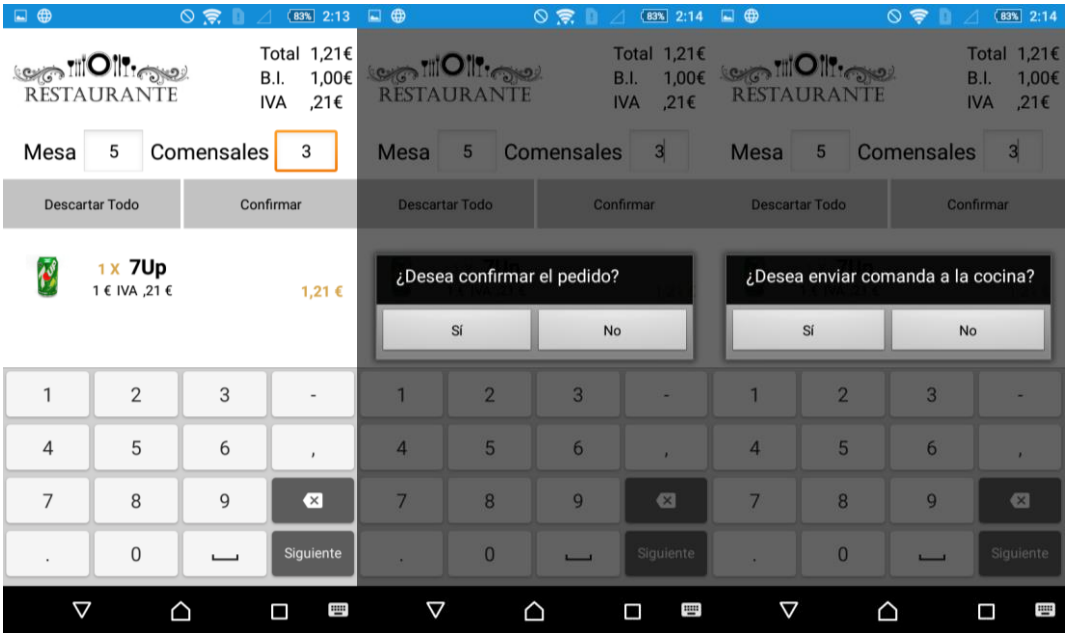


Figura 6.l:Confirmar. Figura 6.m: Pantalla emergente. Figura 6.n: Aceptar comanda.

Finalmente, se habrá enviado a cocina siempre y cuando estén los puertos configurados y haya otro dispositivo escuchando la petición.

OfiComanda

Permite gestionar comandas productos y comandas de forma simple y sencilla.

Contras:

- Muy pesada.
- Necesita complemento.
- Es de pago.
- Demasiado simple.

Consideraciones:

- Incluye sugerencias o plato/tapa/menú del día.

Pruebas.

Abrimos la aplicación (Figura 7.a) y pulsamos sobre “Importar”, seleccionamos “Si” (Figura 7.b) en la ventana emergente pero nos devuelve un error puesto que no tenemos instalado el complemento necesario (Figura 7.c).

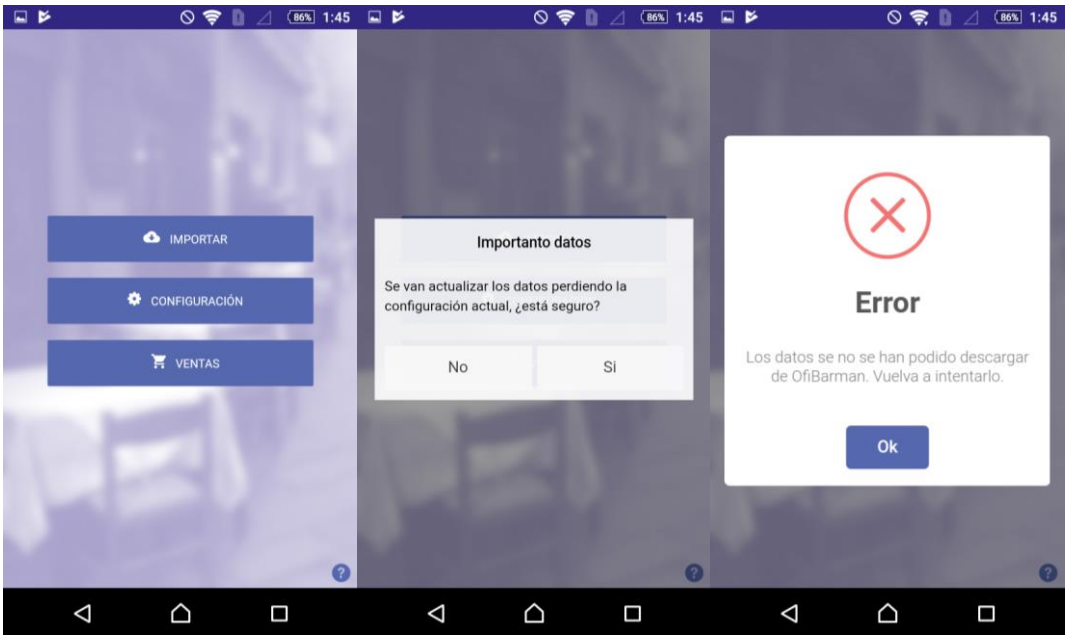


Figura 7.a:Inicio OfiComanda Figura 7.b: Importar datos Figura 7.c:Error

OrderDroid

Permite realizar comandas a los camareros, consultar órdenes en cada una de las mesas, imprimir recibos, realizar entradas de pedidos, realizar entradas de inventarios, realizar entradas de etiquetas, comprobar de precios, modo auto-venta o preventa para realizar la toma de pedidos, albaranes y facturas, generación de estadísticas.

Contras.

- Difícil de comprender.
- Interfaz mejorable
- Varios errores a la hora de acceder a mesas.
- Se queda pillado en bucle.

Consideraciones.

- Incluye estadísticas que permiten hacer estudios de mercado.

Pruebas.

Abrimos la aplicación y e introducimos la fecha actual (Figura 8.a).

A continuación se nos muestra la pantalla principal (Figura 8.b).

Figura 8.a: Configuración Figura 8.b: Pantalla principal

En el siguiente paso creamos un empleado (Figura 8.c), una mesa (Figura 8.d) y un producto de prueba (Figura 8.e).

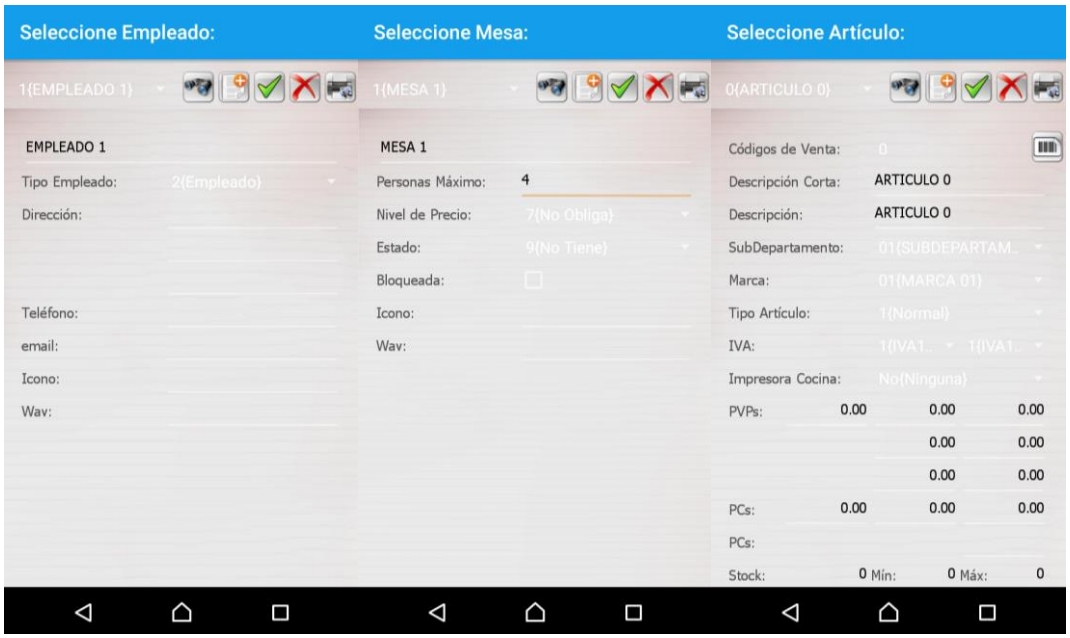


Figura 8.c:Crear empleado

Figura 8.d:Crear mesa

Figura 8.e:Crear articulo

Por último, intentamos crear una comanda y recibimos un error (Figura 8.f).

Figura 8.f: Error

Tabla comparativa.

A continuación se va a realizar una tabla comparativa para poder tener una visión más clara de en qué cualidades flojean las aplicaciones ya existentes y cómo se puede mejorar para facilitar la vida de los clientes

	Tiene versión demo	Es gratuito	No necesita complementos	Es fácil de manejar	Es completo	Es personalizable	Genera Estadísticas	Tiene auto-pedido	En español
Alf@ Cloud Comandas	X	X	V	V	V	V	X	V	V
Comanda 2000	X	V	V	V	X	X	X	X	V
Comandera Plus	X	X	X	X	X	V	X	X	V
DroidPOS Demo	V	X	X	X	V	V	X	X	V
GPI Menu Phone	X	X	V	X	V	X	X	X	V
Nectenia	V	V	V	V	V	X	X	V	V
OfiComanda	X	X	X	V	X	X	X	X	V
OrderDroid	V	V	V	X	X	X	V	X	V
Comandas Facile	X	V	X	V	X	X	X	X	V
Comandas	X	V	X	V	X	V	X	X	X
Comandero	X	X	V	V	X	V	X	X	V
EMC Comanda Electrônica	X	V	V	V	X	X	X	X	X
MyGest	X	V	X	V	X	X	X	X	V
OrderLanDemo	X	V	X	V	X	X	X	X	V
Prisma Comanda	X	V	X	V	X	X	X	X	V
Restaurant Cafe Service	X	V	V	V	X	X	X	X	X

Resulti - Comanda Electrônica	X	V	V	V	X	X	X	X	X
Table Mr. Comanda	X	X	V	V	X	X	X	X	V
Restages Mobile	X	V	X	V	X	X	X	X	V
ComandUAL	V	V	V	V	V	V	V	V	V

3. Materiales y métodos.

3.1. Introducción.

Tras estudiar distintas herramientas y métodos de desarrollo, se ha estimado conveniente la utilización del lenguaje de desarrollo Android en su entorno de desarrollo Android Studio para la aplicación móvil, Python y DJango para la API RestFul y la base de datos externa respectivamente y DJango como lenguaje de desarrollo para la aplicación web.

La metodología que se va a llevar a cabo para desarrollar el proyecto es Kanban

Para llevar a cabo el proyecto se va a usar un equipo MSI modelo MS-7817 con Windows 10 de 64 bits con un procesador Intel Core i5 de 3.2HGz, una memoria RAM de 16.0 GB con tarjeta gráfica Intel HD Graphics 4600 y versión de DirectX 12.

A continuación se va a llevar a cabo un exhaustivo estudio para determinar el mejor modelo o metodología de desarrollo a la hora de llevar a cabo el proyecto así como de los lenguajes y herramientas que se utilizarán para la realización del proyecto.

3.2. Metodologías.

Un modelo para el desarrollo de software es una representación abstracta de un proceso.

Cada modelo representa un proceso desde una perspectiva particular y así proporcione información parcial sobre el proceso [1].

3.2.1. En cascada.

Este modelo de desarrollo está planteado en bloques individuales que se van concatenando en el tiempo.

Este modelo se basa en cuatro fases distintas llamadas análisis, diseño, implementación e integración.

- El **análisis** de requerimientos consiste en reunir las necesidades del producto y casi siempre su salida es texto.
- El **diseño** describe la estructura interna del producto y suele representarse con diagramas y texto.
- La **implementación** significa programación. Producto de esta etapa es el código en cualquier nivel, incluido el producido por sistemas de generación automática.
- La **integración** es el proceso de integración es el proceso de ensamblar las partes para completar el producto.[2]

Ventajas

- Permite la departamentalización y control de gestión.

- El horario se establece con los plazos normalmente adecuados para cada etapa de desarrollo.
- Este proceso conduce a entregar el proyecto a tiempo.
- Es sencilla y facilita la gestión de proyectos.
- Permite tener bajo control el proyecto.
- Limita la cantidad de interacción entre equipos que se produce durante el desarrollo.[3]

Inconvenientes

- No refleja realmente el proceso de desarrollo del software. Ya que la mayoría de los que desarrollan proyectos no cumple con este lineamiento.
- Se tarda mucho tiempo en pasar por todo el ciclo
- La aplicación de la metodología en cascada se orienta mejor al desarrollo de proyectos de corto plazo, de poca innovación y proyectos definitivos y detallados.
- La metodología puede confundir al equipo profesional en las etapas tempranas del proyecto.
- No es frecuente que el cliente o usuario final explicite clara y completamente los requisitos.[3]

La siguiente figura es una ilustración de las fases del modelo en cascada

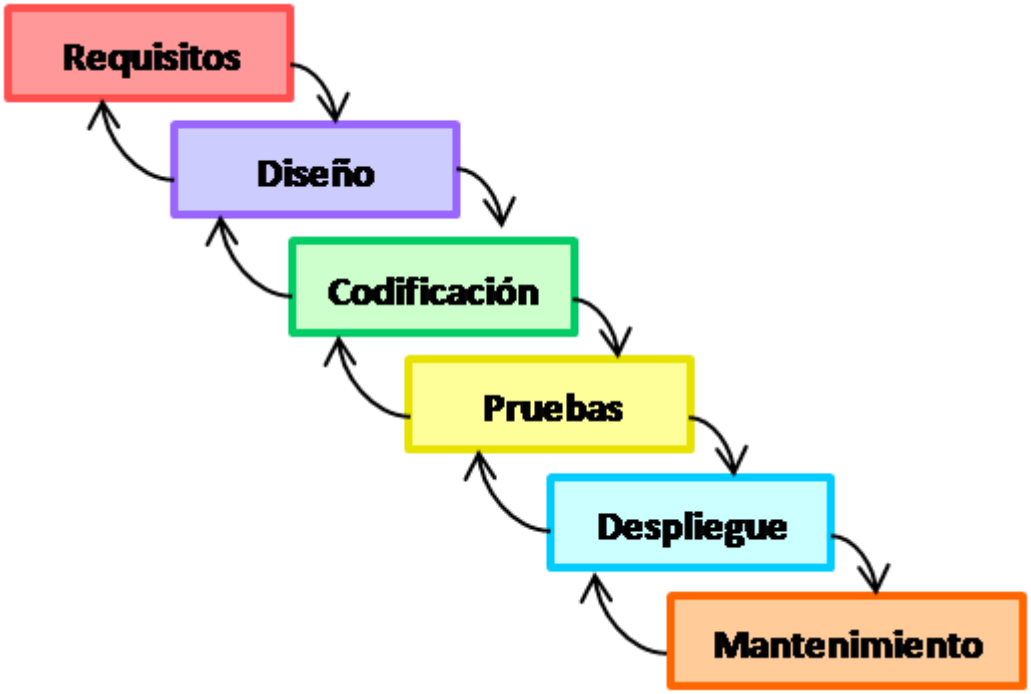


Figura 9. Fases del modelo en cascada.

3.2.2. Incremental.

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos.

Es usado cuando no se tienen totalmente claros los requisitos o se quiere realizar un proyecto en fases. Se basa en la filosofía de construir incrementando las funcionalidades del programa.[4]

Ventajas

- Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.
- Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.
- Es más fácil probar y depurar en una iteración más pequeña.
- Es más fácil gestionar riesgos.
- Cada iteración es un hito gestionado fácilmente.[5]

Inconvenientes

- Para el uso de este modelo se requiere de cierta experiencia para definir los incrementos y distribuir en ellos las tareas de forma proporcionada.
- Cada fase de una iteración es rígida y no se superponen con otras.
- Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio.[5]

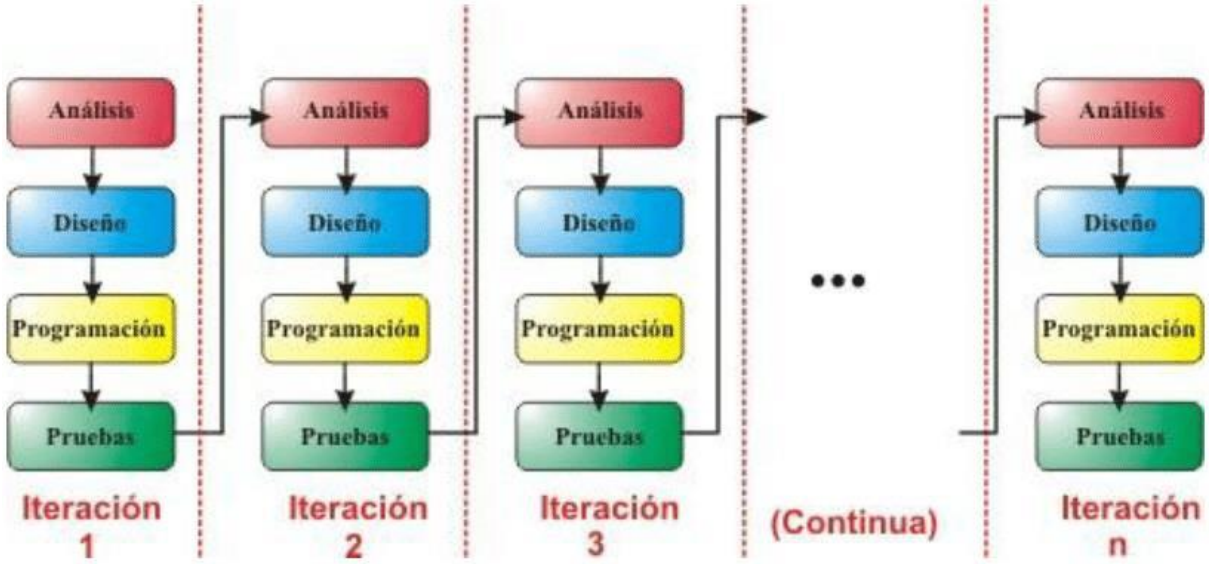


Figura 10. Fases del modelo incremental.

3.2.3. Espiral.

El modelo en espiral, propuesto originalmente por BOEHM en 1976, es un modelo de proceso de software evolutivo donde se conjuga la naturaleza de construcción de prototipos con los aspectos controlados y sistemáticos del modelo en cascada. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software que no se basa en fases claramente definidas y separadas para crear un sistema.[6]

Ventajas

- El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora.
- Como el software evoluciona a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los nivele evolutivos.
- El modelo en espiral permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
- El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas.
- En la utilización de grandes sistemas a doblado la productividad.[7]

Desventajas

- Resulta difícil convencer a grandes clientes de que el enfoque evolutivo es controlable.
- Debido a su elevada complejidad no se aconseja utilizarlo en pequeños sistemas.
- Genera mucho tiempo en el desarrollo del sistema
- Modelo costoso
- Requiere experiencia en la identificación de riesgos.[8]

Figura 11. Fases del modelo en espiral.

3.2.4. Metodologías ágiles.

Una metodología ágil es un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar nuevo software. [9]

Estas metodologías ágiles tienen como característica que cumplen con el manifiesto ágil:

- La principal prioridad es satisfacer al cliente a través de la entrega temprana y continua del software de valor.
- Se aceptan los requisitos cambiantes, aun llegando tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses.
- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
- La forma más eficiente y efectiva de comunicar información dentro de un equipo de desarrollo es mediante la conversación en persona.
- El software/producto/servicio que funcione es la principal medida de progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los desarrolladores, patrocinadores, y usuarios han de mantener un ritmo constante de forma indefinida.[10]

A continuación se realizará el estudio de tres tipos de metodologías ágiles.

3.2.5. Scrum.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.[11]

Ventajas

- Entregables en tiempo y forma, puedes ir enviando entregables al cliente mientras vas atacando los objetivos más sencillos, eso te hace ganar tiempo para atacar los objetivos más complejos.
- el ScrumMaster tiene el conocimiento necesario para lograr el objetivo primario y secundario por lo cual puede ir controlando el proyecto y delegando roles.
- Cada persona sabe que es lo que tiene que hacer y no es necesario estar reorganizando una y otra vez los Tracks de cada persona.
- Se involucra desde un principio y se da un rol a todos los stakeholders (personas que van a participar en el proyecto incluyendo cliente final, QA, Testers, etc.)[12]

Desventajas

- Algunos miembros del equipo pueden saltar pasos importantes en el camino rápido para llegar al “sprint” final.

- Debido a las reuniones diarias, a veces es muy cansador y estresante reunirse continuamente.
- Si un recurso falla o hay algún cambio es complicado reemplazar ese rol ya que es la persona que se lleva el conocimiento específico y afecta a todo el proyecto.[12]

3.2.6. Kanban.

La metodología ágil Kanban se basa en la idea de que el trabajo en curso (Work In Progress, WIP) debería limitarse y sólo deberíamos empezar con algo nuevo cuando un bloque de trabajo anterior haya sido entregado o ha pasado a otra función posterior de la cadena.

Esta técnica se creó en Toyota, y se utiliza para controlar el avance del trabajo, en el contexto de una línea de producción. Actualmente está siendo aplicado en la gestión de proyectos software.[13]

Principios básicos del método Kanban:

- Calidad garantizada: todo lo que se produce debe salir bien a la primera, sin margen de error. De esta forma en Kanban no se permite la rapidez, sino la calidad final de las tareas realizadas.
- Reducción del desperdicio: se basa en hacer solamente lo justo y necesario, pero hacerlo bien. Esto supone la reducción de todo lo superficial o secundario.
- Mejora continua: además de ser un método de gestión, es también un sistema de mejora en el desarrollo de proyectos, según los objetivos a alcanzar.
- Flexibilidad: lo siguiente a realizar se decide del backlog (tareas pendientes acumuladas), pudiéndose priorizar aquellas tareas entrantes según las necesidades del momento.[14]

Ventajas

- Reducción del trabajo en progreso.
- **Control del flujo de trabajo.** Fácil localización de cuellos de botella, y control sencillo del estado de desarrollo del producto.
- **Fácil de aplicar.** Apenas existen reglas y se puede llevar a cabo de forma artesanal, no requiere software adicional.
- **Promueve el trabajo en equipo.** Los desarrolladores se comunican entre sí y se ayudan cuando alguna actividad da problemas.
- **Reducción del tiempo perdido.** Todo el desarrollo se hace bajo demanda, por lo que no hay partes que sobren, además si alguien acaba su trabajo ayuda al resto del equipo.
- **Facilidad para añadir mejoras**. El tablero permite que todo el equipo de desarrollo pueda ver cómo se está haciendo el producto desde una visión global, lo que permite que puedan reflejar a sus compañeros diversas mejoras sobre el mismo.[15]

Desventajas

- **Dificultad de realizar las entregas a tiempo en grandes proyectos**. Dado que no hay un control específico del tiempo empleado en cada actividad, en grandes proyectos pueden acumularse un gran número de pequeños retrasos que provocan la demora en la entrega del producto final.

- **Falta de reglas:** Aunque la existencia de pocas reglas es una ventaja, puede convertirse en un problema cuando el desarrollador es inexperto y necesita una guía para realizar su trabajo. Por ello, se aconseja hacer uso de Kanban tras haber ganado experiencia con otras metodologías, ya que de este modo habrá reglas que se habrán interiorizado.
- **Dificultad a la hora de prever posibles problemas.** Aunque la localización y solución de problemas es sencilla en Kanban, su previsión antes de que ocurran no lo es.[15]

3.2.7. XP o Extreme Programming.

La metodología ágil XP está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo.

XP se basa en retroalimentación continua entre cliente y el equipo de desarrollo. XP es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes.[16]

Características específicas de XP

- Se valora al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software.
- Desarrollar software que funciona más que conseguir una buena documentación.
- La colaboración con el cliente. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo.
- Responder a los cambios. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto determina también el éxito o fracaso del mismo. La planificación no debe ser estricta sino flexible y abierta.[16]

Ventajas

- Da lugar a una programación sumamente organizada.
- Ocasiona eficiencias en el proceso de planificación y pruebas.
- Cuenta con una tasa de errores muy pequeña.
- Propicia la satisfacción del programador.
- Fomenta la comunicación entre los clientes y los desarrolladores.
- Facilita los cambios.
- Permite ahorrar mucho tiempo y dinero.
- Puede ser aplicada a cualquier lenguaje de programación.
- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el proyecto.
- La XP es mejor utilizada en la implementación de nuevas tecnologías.[17]

Desventajas:

- Es recomendable emplearla sólo en proyectos a corto plazo.
- En caso de fallar, las comisiones son muy altas.

- Requiere de un rígido ajuste a los principios de XP.
- Puede no siempre ser más fácil que el desarrollo tradicional.[17]

3.2.8. Conclusiones.

Debido a la naturaleza del proyecto y que no se van a marcar fases o hitos, además de ser un proyecto individual y no en equipo y que se tienen claros los requisitos desde el principio del proyecto, se ha considerado que los modelos que mejor se adaptan a las circunstancias son el modelo en cascada y la metodología ágil Kanban.

Debido a que el modelo en cascada en ocasiones puede ser insuficiente en comparación con la metodología Kanban, se ha decidido que Kanban será la metodología que se llevará a cabo a lo largo del proyecto.

3.3. Nativas VS Híbridas VS Web.

3.3.1. Introducción.

En esta sección se va a llevar a cabo un estudio sobre los distintos tipos de aplicaciones existentes para determinar qué tipo de aplicación se ajusta más a las necesidades del proyecto.

3.3.2. Desarrollo de apps nativas.

Estas aplicaciones son desarrolladas utilizando el lenguaje y el entorno proporcionado por el creador del sistema operativo (Android, Apple, etc.) y son por definición las más fluidas y estables al entenderse a la perfección con el sistema operativo para el que fueron desarrolladas.

Ventajas

- Su funcionamiento y rendimiento es muy bueno a nivel de animaciones, tiempo de respuesta, etc.
- Utilizan al máximo los recursos software y hardware del equipo donde se encuentran instaladas.
- No tienen prácticamente ninguna limitación técnica ya que podemos adaptar el app al 100%.

Inconvenientes

- Son exclusivas para el sistema operativo en el que fueron desarrolladas (Android, IOS, Windows Phone, Blackerry OS, etc.)
- Los desarrollos son más caros que con cualquier otra técnica: Si queremos desarrollar una app y publicarla en varios sistemas. No es posible la reutilización de código fuente para las diferentes plataformas.
- El proceso es más lento. Se necesita más tiempo para desarrollar la aplicación en ambas plataformas. [18]

3.3.3. Desarrollo de apps híbridas.

El desarrollo de aplicaciones híbridas es una técnica de desarrollo por la cual podemos desarrollar una app para varios sistemas operativos (iOS, Android, etc.) con el mismo código fuente.

Ventajas

- Los costes de desarrollo son menores que los de las apps nativas ya que pagamos una sola vez el desarrollo
- Con el desarrollo adecuado, son multiplataforma por lo que nos funcionan en todos los sistemas operativos.
- Permite distribución a través de las diferentes tiendas de aplicaciones por lo que nos podemos beneficiar de sus pasarelas de pago y de su difusión.
- Si quisiéramos publicar la misma aplicación en la www, lo podríamos realizar.

- Amplia comunidad de desarrolladores con lo que existen muchas librerías que se pueden reutilizar para cubrir distintos casos de uso.
- Permite combinar código nativo con código Phonegap de manera que si hubiese algo que no se puede hacer en Phonegap directamente, se podrá hacer desarrollando código nativo para cada plataforma.
- No requiere licencia expresa para desarrolladores y es fácil encontrar desarrolladores para la plataforma ya que para los desarrolladores con experiencia en programación web, la curva de aprendizaje es sencilla.

Inconvenientes

- Rendimiento: suelen tener bastante peor rendimiento que las apps nativas tanto al realizar animaciones de elementos de pantalla como en los tiempos de respuesta de la app… es algo que se puede depurar pero siempre es perceptible.
- Pueden llegar a tener un acceso limitado a la funcionalidad y servicios del dispositivo.
- Es muy difícil tener un buen diseño visual que se adapte a todos los tamaños de pantalla y distintos dispositivos.
- Pueden tener fallas de seguridad importantes debido a que usan el motor de renderizado web.
- Pueden tener bugs difíciles de depurar ya que no existen herramientas potentes para trazar errores. [18]

3.3.4. Desarrollo de Web Apps.

Otra de las posibilidades de las que disponemos son las aplicaciones web responsive o web apps que son desarrolladas utilizando el lenguaje habitual que tiene cualquier web de Internet. Estas “apps” se realizan utilizando javascript, html y css, combinado con herramientas o frameworks para web apps (jQuery Mobile, Sencha touch, etc.).

Ventajas

- Son aplicaciones multidispositivo y no importa su sistema operativo.
- El coste de desarrollo en proporción a las aplicaciones nativas, Phonegap o Xamarin es netamente inferior
- Podremos publicar el app en el momento en que queramos, sin tener que esperar a que el propietario de la tienda de aplicaciones (App Store, Google Play, etc.) nos dé su consentimiento para su publicación. Sin embargo para acceder al app, los usuarios deberán usar el navegador web de su dispositivo móvil y dirigirse a la URL que les indiquemos. El usuario siempre dispone de la última versión porque la podemos actualizar a demanda.
- Puede reutilizarse sitios responsive ya diseñados para su uso en móviles.

Inconvenientes

- No se puede publicar en las stores por lo que su difusión es más complicada y por tanto dificulta su éxito.

- No disponemos de todas las ventajas de uso de las aplicaciones nativas/Híbridas en cuanto al aprovechamiento de hardware o el software del dispositivo.
- El rendimiento y tiempo de carga es muy inferior a cualquier otra técnica de desarrollo de apps
- Hay funcionalidades que no son posibles de implementar al no estar permitidas desde el navegador web del usuario.
- Dependiendo del navegador y sistema operativo del usuario que acceda, es posible que no podamos hacer uso de muchos de los accesorios de un móvil como el GPS, la cámara, el micrófono, etc.
- Requiere conexión a Internet al no estar instalada en el dispositivo
- No podremos hacer uso de los sistemas de compras que habilitan los fabricantes para las apps: In app purchases (tendremos que habilitar los pagos por los medios tradicionales web: TPV, PayPal, etc.). [18]

3.3.5. Conclusiones.

Debido a las numerosas desventajas que presentan las webapps frente a sus ventajas, se ha descartado esta opción para el desarrollo de la aplicación.

Por otro lado, debido a la naturaleza del proyecto, un buen rendimiento de la aplicación y el menor tiempo de respuesta posible es esencial puesto que va a conectar e interaccionar con multitud de dispositivos paralelamente y en tiempo real.

La ventaja más importante de una aplicación híbrida es que puede ser multiplataforma pero, por otro lado, las aplicaciones híbridas empeoran el rendimiento de la aplicación y además son más difíciles de depurar, por lo que en conjunto es una mala alternativa para el desarrollo del software.

Por ello, hemos llegado a la conclusión de que la mejor solución es el desarrollo de la aplicación de forma nativa.

Una vez llegado a esta conclusión debemos preguntarnos, ¿En qué sistema nativo deberíamos desarrollar nuestra aplicación?

3.4. Android VS IOS.

3.4.1. Introducción.

En el siguiente apartado se ha a hacer análisis para determinar para qué plataforma móvil es más conveniente desarrollar la aplicación móvil.

3.4.2. Accesibilidad.

IOS

Para poder programar en iOS debemos de tener un Mac y una vez en posesión de uno tenemos que poseer la última versión de Mac Os para descargar Xcode o pagar en caso contrario por la última versión de Xcode.

Por otro lado, las aplicaciones iOS están ligadas exclusivamente a dispositivos Apple y debido el elevado precio de un dispositivo iOS, tenemos que tener en cuenta que muchos de los potenciales clientes de la aplicación no podrán adquirirla.

En el segundo trimestre de 2016, las ventas de Apple fueron del 12.9% del mercado total de dispositivos móviles.[19]

Android

Para poder programar en Android necesitaremos un entorno de desarrollo donde la mayoría son multiplataforma (disponibles para Mac, Linux y Windows) y la SDK que podremos instalar en cualquiera de estas plataformas.

Por otro lado Android está orientado a un público masivo, sirviendo para multitud de dispositivos y posibilitando adquirir un smartphone a un precio asequible para la mayoría, por lo que la cantidad de potenciales clientes que pueden adquirir la aplicación se eleva de forma considerable.

En el segundo trimestre de 2016, las ventas de Android fueron del 86.2% del mercado total de dispositivos móviles.[19]

3.4.3. Curva de aprendizaje.

IOS

Si desarrollamos en iOS, actualmente tenemos la opción de programar en Swift o en Objective-C, aunque lo ideal es aprender ambos ya que se integran a la perfección y cada uno tiene utilidades que no puede realizar el otro.

Por un lado, la curva de aprendizaje de Objective-C es mucho más elevada que Swift y Swift es 2,6 veces más rápido que Objective-C.

Por otro lado, Swift es un lenguaje nuevo por lo que tiene menos comunidad, menos librerías y seguramente deba ser actualizado por bugs y mejoras.

Objective-C tiene todas las características de bajo nivel de C.

Debido a su corto tiempo de vida, apenas existen cursos o libros sobre Swift de forma gratuita y aunque Objective-C es más antiguo, debido a su exclusividad la Comunidad de usuarios no es tan grande y no existen tantas librerías o esqueletos de aplicaciones como con otros lenguajes.[20]

Android

Por su lado, la curva de aprendizaje de Java es muy sencilla al principio, pero a medida que se va profundizando en sus utilidades se empieza a complicar de forma considerable.

La comunidad de Java es enorme, haciendo que casi cualquier problema o duda ya haya sido resuelta con anterioridad por otros usuarios de Java, lo que facilita el trabajo en gran medida.

Gracias a esta comunidad, existen multitud de libros, cursos y tutoriales gratuitos con los que se podrá llegar a tener un conocimiento muy elevado del lenguaje.[20]

3.4.4. Gestión de la memoria.

IOS

Hasta iOS 5, el programador debía hacerse cargo de hacer el RELEASE de las variables que ya no fuera a utilizar.
Cuando un objeto se utiliza desde varias partes del código, se produce lo que se llama Reference Counting, el cual es un contador de los sitios donde se está utilizando dicho objeto.
Así, cuando las partes de nuestra aplicación van liberando este objeto mediante su método RELEASE, el contador va disminuyendo hasta llegar a cero, cuando se destruye automáticamente liberando ese espacio en memoria.
Este proceso hace que la recolección de basura en iOS fuera tediosa.

A partir de iOS 5, se implementa lo que se llama “Automatic Reference Counting” (ARC) [21] que es una funcionalidad proporcionada por LLVM 3.0, el nuevo compilador incorporado por Apple como predeterminado en Xcode 4.2.
ARC realizará por nosotros todas las operativas de RETAIN y RELEASE sin que tengamos que preocuparnos por ello, haciendo la gestión de memoria más eficiente.[23]

Android

Por la parte de Android, la gestión de memoria se realiza mediante el Garbage Collector.[22]
El Garbage Collector es un proceso de baja prioridad que se ejecuta en la JVM y es el encargado de liberar la memoria que no se emplea. El ser de baja prioridad supone que no pueda estar todo el rato trabajando, y que solo se le asigne su tarea cuando el procesador no tiene un trabajo con mayor prioridad en ejecución. Sin embargo, al tratarse de un proceso de prioridad baja, es poco probable que se ejecute cuando se esté haciendo un uso intensivo de la CPU. [23]

3.4.5. Documentación.

IOS

Existe gran cantidad de documentación y ejemplos para iOS a nivel básico pero cuando se necesita hacer algo que se sale de los esquemas generales de una aplicación es difícil encontrar ayuda.
Esto se debe a que la comunidad no es tan grande como en otros lenguajes de programación.[24]

Android

La comunidad detrás de Android es enorme y es difícil necesitar hacer algo de lo que no haya ejemplos o documentación o bien del propio Google, o bien de la comunidad de usuarios que utiliza android.[24]

3.4.6. UIViewController vs Activity.

IOS

En el caso de iOS, el controlador UIViewController es un componente del núcleo de la aplicación, con el cual podemos gestionar los ciclos de vida de los eventos, subvistas

Los diseños están escritos en .Xib.

El editor de diseño de xCode es más suave que el editor de diseño de Android.[25]

Android

En Android se utiliza la clase Activity, que representa una pantalla en un dispositivo Android.

Los diseños están hechos en .XML.

Los diseños en .XML pueden ser fácilmente reutilizables.[25]

3.4.7. UIViewController vs Activity.

IOS

El simulador de iOS sólo imita el software del dispositivo y puede acceder a cualquier recurso del ordenador y es bastante más rápido comparado con el emulador de Android. Sin embargo, el emulador de iOS falla al dar una representación exacta de un dispositivo iOS.[26]

Android

El emulador de Android imita tanto el software como el hardware de un dispositivo, de forma que la simulación es mucho más realista que lo que ofrece el simulador de iOS. Se puede configurar el dispositivo de emulación con los parámetros específicos de hardware.

Sin embargo, es una mala idea que el rendimiento de la CPU sea el del dispositivo puesto que lo hace extremadamente lento. Si el objetivo del emulador es ahorrar tiempo evitando usar un dispositivo real, resulta molesto usar un emulador que tarde como un dispositivo real.[26]

3.4.8. Licencias.

IOS

Para poder ser desarrollador de Apple se debe pagar una licencia de 99$ y sus aplicaciones más baratas cuestan 0,99€, de los que sus desarrolladores en Europa se llevan en torno a un 70%, mientras que en USA obtienen un 60%.[27]

Android

Para poder ser desarrollador de Android, deberemos pagar una licencia de 25$. Una vez pagada, el desarrollador recaudará un 70% de los beneficios de la aplicación, en caso de que esta sea de pago.[27]

3.4.9. Conclusiones.

Una vez estudiadas las características, ventajas e inconvenientes de los dos principales sistemas nativos del mercado, iOS y Android, hemos llegado a la conclusión de que si bien ninguno sobresale por encima del otro, debido a la naturaleza y necesidades de nuestro proyecto, hemos decidido desarrollar la aplicación para la plataforma Android debido principalmente a la necesidad de llegar al máximo número de clientes posibles.

Una vez decidida la plataforma de desarrollo debemos plantearnos en qué IDE (entorno de desarrollo) llevaremos a cabo la aplicación.

3.5. IDE Android.

3.5.1. Introducción.

Una vez definida la plataforma móvil para la que se va a desarrollar la aplicación, se va a realizar un estudio para determinar el entorno de desarrollo sobre el que se va a desarrollar la aplicación.

3.5.2. Basic4Android.

Basic4Android es un entorno comercial que permite desarrollar aplicaciones para Android programando en Visual Basic.
Es una herramienta RAD (Rapid Application Development) para apps nativas de Android.

Ventajas

- No se requiere saber programar en XML.
- Depurador rápido.
- Editor visual WYSIWYG para Android.

Desventajas

- La versión estándar cuesta 60$.
- Versión de prueba de 30 días muy limitada.
- Escasez de librerías.[28]

3.5.3. App Inventor.

Es una plataforma desarrollada por Google Labs para que gente sin experiencia pueda desarrollar aplicaciones. Es un entorno totalmente visual en el que no hace falta código para desarrollar apps en Android.

Ventajas

- Su simplicidad hace que cualquier persona interesada pueda desarrollar aplicaciones.
- Es una herramienta gratuita.

Desventajas

- Su simplicidad hace que no se pueda desarrollar todo el potencial de Android.[29]

3.5.4. Eclipse.

Eclipse es un entorno de desarrollo integrado, de Código abierto y Multiplataforma.

Ventajas

- Es gratuito.
- Al ser multiplataforma, el mismo IDE sirve para desarrollar la aplicación android, de escritorio y web.

Desventajas

- Google ha dejado de dar soporte a Eclipse para desarrollos Android.[30]

3.5.5. Android Studio.

Android studio es un entorno de desarrollo impulsado por Google y es el IDE oficial para el desarrollo de aplicaciones para Android, reemplazando a Eclipse.

Ventajas

- Es el IDE recomendado por Google para el desarrollo de Aplicaciones Android.
- Renderización en tiempo Real.
- Soporte para construcción basado en Grandle.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, entre otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Herramienta de desarrollo gratuita.

Desventajas

- Requisitos algo elevados para que funcione correctamente.
- Curva de aprendizaje un tanto compleja.
- El sistema de construcción de proyectos Gradle puede resultar complicado inicialmente.[31]

3.5.6. Conclusiones.

Puesto que son de pago, hemos descartado la posibilidad de utilizar Basic4Android y App Inventor.

Por otro lado, dado que Google ha establecido Android Studio como IDE predeterminado para el desarrollo de apps en Android y ha dejado de dar soporte a Eclipse y las ventajas de Android Studio sobre el resto, hemos decidido utilizar el IDE Android Studio para desarrollar el proyecto.

3.6. Desktop App VS Web App.

3.6.1. Introducción.

En el siguiente apartado se van a estudiar las diferencias entre aplicaciones de escritorio y aplicaciones web para determinar qué tipo de aplicación es más beneficiosa para ser desarrollada.

3.6.2. Desktop App.

Una Desktop app es aquella que está instalada en el ordenador del Usuario, que es ejecutada directamente por el sistema operativo, ya sea Microsoft Windows, Mac OS X, Linux

Ventajas.

- Aprovecha los recursos de la máquina en la cual se ejecuta.
- Suelen ser más robustas y estables que las aplicaciones Web.
- El tiempo de respuesta es muy rápido.
- pueden ser más seguras que las webapp si se desarrollan bien.

Desventajas.

- El Software debe Instalarse siempre en el pc del usuario.
- Requieren instalación y actualización personalizada.
- Una aplicación de escritorio no puede ejecutarse en cualquier dispositivo debido a su arquitectura.[32]

3.6.3. Web App.

Una Web App es aquella que está instalada en un Servidor y para su ejecución sólo se requiere disponer de un ordenador con conexión a Internet y de un Navegador.

Ventajas.

- Su funcionalidad es independiente del sistema operativo instalado en el ordenador del usuario.
- No hay problemas de incompatibilidad entre versiones, porque todos los Usuarios trabajan con la misma
- Son fáciles de actualizar y mantener.

- Son aplicaciones muy ligeras debido a que el programa no está instalado en el pc, por lo que el Usuario no necesita tener un ordenador de grandes prestaciones para trabajar con ellas.

Desventajas.

- El aprovechamiento de los recursos de la máquina cliente se ve limitada debido a que se ejecuta en el navegador, es decir, la aplicación depende de una conexión a internet.[32]

3.6.4. Conclusiones.

Debido a las múltiples características de las webApp frente a las aplicaciones Desktop y las pocas desventajas sobre él, se ha decidido desarrollar la aplicación principal como una webApp.

3.7. Lenguaje de la WebApp.

3.7.1. Introducción.

Vamos a comparar las características y diferencias entre php, nodeJs, ruby y python para elegir en qué lenguaje de programación llevaremos a cabo la webApp y la Api Rest.

3.7.2. PHP.

PHP es un lenguaje de programación de uso general, open source, que funciona del lado de servidor, diseñado en principio para producir páginas web dinámicas.

Es ideal para aplicaciones donde hay páginas con mucho contenidos o datos complejos.

El framework más usado para PHP es laravel.

Ventajas.

- Lo soportan la mayoría de las plataformas de alojamiento web.
- Tiene ciertas características de los lenguajes orientados a objetos como la utilización de clases y herencias.
- Puede mezclarse con código HTML, aunque esto dificulta su lectura.
- Puede manejar ficheros y conectarse a distintas bases de datos.
- Existe numerosa documentación por lo que es relativamente sencillo resolver los problemas que nos puedan surgir durante el desarrollo de un sitio web.

Desventajas.

- Al ser interpretado en el servidor, es más fácil que se colapse cuando el número de peticiones de descarga de páginas aumenta.
- Parte del contenido de las páginas puede no ser accesible a los navegadores, dificultando el posicionamiento de las páginas.
- No es tan rápido como sus competidores.[33]

3.7.3. NodeJs.

Node.js es un framework, open source, que funciona del lado del servidor. Por lo tanto, permite que el servidor y las aplicaciones de escritorio se comuniquen por medio de Javascript.

Es ideal para aplicaciones que gran cantidad de llamadas asincrónicas por página para consultar pequeños dato, como por ejemplo mostrar datos en tiempo real.

El framework para node.js más usado es Express

Ventajas.

- Permite utilizar el mismo lenguaje (javascript), tanto en el cliente como en el servidor.

- Ofrece muy buena gestión de paquetes gracias a NPM (si quieres hacer algo, probablemente exista una librería/paquete que ya lo ofrezca). Detrás de Node hay una gran comunidad documentando, haciendo tutoriales y creando nuevos módulos.
- Con Node.js es posible hacer en el servidor, todo lo que necesitas – acceso a ficheros, a bases de datos, conexiones de clientes, entre otros.

Desventajas.

- La base de datos por defecto de NodeJs es mongodb, la cual no es una base de datos relacional.[34]

3.7.4. Ruby.

Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk.

El framework más usado para Ruby es RubyOnRails.

Ventajas.

- Claridad y simplicidad en el código fuente.
- Lenguaje fácil de aprender y curva de aprendizaje muy suave.
- Gracias a su framework RubyOnRails se pueden hacer cosas realmente potentes con él.
- Existe gran cantidad de módulos programados.
- Existe gran cantidad de documentación.

Desventajas.

- El tiempo de respuesta es lento en comparación con otros lenguajes de desarrollo.[35]

3.7.5. Python.

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web.

El framework más usado para desarrollar con python es Django.

Ventajas.

- Legibilidad del código.
- Abundancia de bibliotecas.
- Facilidad de uso.

Desventajas.

- Existen pocos Hostings compatibles.
- Resulta muy complejo desarrollar un proyecto sin un framework que lo apoye.[36]

3.7.6. Conclusiones.

Una vez conocidas las ventajas y desventajas de cada uno de los lenguajes de programación y sus frameworks, hemos decidido descartar node.js y ruby debido a que sus características no se ajustan a las necesidades del proyecto.

Por otro lado, tanto PHP como Python son buenas opciones y se posee conocimiento de ambos lenguajes, pero vamos a decantarnos por Python por la razón de que es menos usual en la comunidad de desarrolladores.

A continuación se va a pasar a especificar los detalles de la webbApp y de la api que conectará la webApp con las appPhone

3.8. Arquitectura Rest VS Soap

3.8.1. Introducción.

Para realizar un estudio para seleccionar la mejor forma de llevar a cabo la conexión entre los distintos elementos del proyecto.

3.8.2. Rest.

REST (Representational State Transfer)[37] Es un estilo de arquitectura de software para sistemas distribuidos que se centra en el uso de los estándares HTTP y XML para la transmisión de datos sin la necesidad de contar con una capa adicional.

Ventajas.

- Es muy ligero, sus respuestas contienen exactamente la información que necesitamos.
- Para sencillo de interpretar.
- Es sencillo de desarrollar.
- Es flexible en cuanto al tipo de respuesta que se necesita.

Desventajas.

- Su seguridad puede llegar a ser un problema y una tarea difícil de implementar.
- No hay un estándar en sus respuestas por lo que no se definen tipos de datos.[38]

3.8.3. Soap.

SOAP (Simple Object Access Protocol) es una infraestructura basada en XML donde cada objeto puede tener métodos definidos por el programador con los parámetros que este considere necesarios.[39]

Ventajas.

- El resultado que siempre es XML contiene una definición específica del tipo de dato, lo que hace del protocolo algo muy estricto.
- Es más seguro debido a que su implementación la mayor parte de las veces se hace del lado del servidor.

Desventajas.

- Una vez implementado, si se desea cambiar algo en el servidor impacta de forma negativa en los clientes ya que estos tienen que hacer muchas modificaciones al código.
- Las respuestas son demasiado complejas y difíciles de interpretar si no se tienen las herramientas correctas para hacerlo.[40]

3.8.4. Conclusiones.

Puesto que necesitamos un servicio que sea rápido y sencillo de usar, hemos estimado que vamos a utilizar los servicios RestFul.

3.9. Introducción a Api y WebApp

3.9.1. Introducción.

Vamos a comparar las características y diferencias entre php, nodeJs, ruby y python para elegir en qué lenguaje de programación llevaremos a cabo la webApp y la Api Rest.

3.9.2. API.

Para la api vamos a usar el framework Django Rest Framework.

Django Rest Framework es una aplicación Django que permite construir proyectos software bajo la arquitectura REST, incluye gran cantidad de código para reutilizar (Views, Resources, etc.) y una interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP como lo son: POST y GET.

Ventajas.

- La API navegable en la Web es una gran utilidad para sus desarrolladores.
- Serialización que admite fuentes de datos ORM y no ORM.[41]

3.9.3. WebApp.

Para la webApp se va a usar el lenguaje python con el framework Django.

Django es un framework web de alto nivel diseñado para Python que fomenta un desarrollo rápido y un diseño limpio y pragmático, siendo de código abierto y gratuito.

Ventajas.

- Django ha sido diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización tan pronto como sea posible..
- Django toma en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes.
- Algunos de los sitios más concurridos en la Web aprovechan la capacidad de Django para escalar rápida y flexiblemente.[42]

4. Análisis de requisitos.

4.1. Introducción.

A continuación se van a detallar los distintos requisitos a nivel de usuario de la aplicación móvil y web tanto para los clientes como para los empleados y los restaurantes.

"Un requisito es la capacidad que debe alcanzar o poseer un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal" IEEE, 1990. [43]

4.1.1. Tipos de requisitos.

Requisitos funcionales:

Los requisitos funcionales definen el comportamiento del sistema que se va a desarrollar, incluyendo los procesos fundamentales que el software llevará a cabo.[44]

Requisitos no funcionales:

Los requisitos no funcionales tienen que ver con restricciones y exigencias de calidad del sistema, cómo pueden ser los requisitos de rendimiento, mantenimiento, seguridad o fiabilidad.[45]

Actores:

Se denomina actor a un rol perfectamente definido que una persona puede desempeñar en el proceso de requisitos.[46]

Los actores involucrados en el proyecto son los siguiente:

- Cliente.
- Empleado.
- Restaurante

4.2. Requisitos funcionales de la aplicación móvil.

4.2.1. Requisitos de usuario comunes.

Estos requisitos están orientados a todos los usuarios de la aplicación, independientemente del rol que desempeñen en ella.

RF01. Versión Demo.

Los usuarios podrán utilizar una versión de prueba antes de registrarse en la aplicación.

RF02. Registrarse, autenticarse y cerrar sesión.

Cualquier usuario podrá registrarse como empresa o como cliente siempre y cuando el nombre de usuario y el correo electrónico no estén ya ocupados. Además, podrá autenticarse en la aplicación así como cerrar sesión.

RF03. Registrarse, autenticarse y cerrar sesión.

Cualquier usuario podrá registrarse como empresa o como cliente siempre y cuando el nombre de usuario y el correo electrónico no estén ya ocupados. Además, podrá autenticarse en la aplicación así como cerrar sesión.

RF04. Modificar los datos.

Cualquier usuario podrá modificar sus propios datos personales, pero no podrá cambiar su nombre de usuario ni modificar los datos de otros usuarios.

4.2.2. Requisitos de usuario de administrador.

Estos requisitos son exclusivos del administrador de un local, que coincide con la cuenta original del local.

RF05. Crear y eliminar usuarios.

El administrador del local podrá crear, leer y eliminar empleados de la base de datos, siempre y cuando el nombre de usuario del empleado no exista ya en la base de datos, pero no podrá modificarlos.

RF06. Asignar ubicación.

El administrador del local podrá actualizar la información del sistema para geolocalizarse en la aplicación y que usuarios clientes puedan encontrar el local en el mapa y el buscador de locales de la aplicación.

4.2.3. Requisitos de usuario de empleados.

Estos requisitos están orientados a los empleados de un local, así como a la cuenta original del local.

RF07. Crear, leer, modificar y eliminar mesas.

Los empleados del local podrán crear y modificar mesas siempre y cuando no exista una mesa con el mismo identificador, además de leer y eliminar mesas de la base de datos.

RF08. Crear, leer, modificar y eliminar espacios del local.

Los empleados del local podrán crear y modificar espacios del local siempre y cuando no exista un espacio con el mismo identificador, además de leer y eliminar espacios de la base de datos.

RF09. Crear, leer, modificar y eliminar productos.

Los empleados del local podrán crear y modificar productos siempre y cuando no exista un producto con el mismo identificador, además de leer y eliminar productos de la base de datos.

RF10. Crear, leer, modificar y eliminar ingredientes.

Los empleados del local podrán crear y modificar ingredientes siempre y cuando no exista un ingrediente con el mismo identificador, además de leer y eliminar ingredientes de la base de datos.

RF11. Crear, leer, modificar y eliminar categorías.

Los empleados del local podrán crear y modificar categorías siempre y cuando no exista un categoría con el mismo identificador, además de leer y eliminar categorías de la base de datos.

RF12. Crear, leer, modificar y eliminar sugerencias.

Los empleados del local podrán crear y modificar sugerencias siempre y cuando no exista un sugerencia con el mismo identificador, además de leer y eliminar sugerencias de la base de datos.

RF13. Modificar Stock.

Los empleados del local podrán consultar y modificar el stock de los productos y los ingredientes y será actualizado en tiempo real.

Los clientes pueden consultar en tiempo real el stock de los productos del local y de los ingredientes que dichos productos llevan.

RF14. Crear comandas de clientes.

Los empleados del local podrán crear comandas tomando nota a los clientes.

RF15. Consultar comandas.

Los empleados del local podrán consultar las comandas generadas en cada una de las mesas, pudiendo ver sus datos como cantidad productos, notas o el estado de la comanda.

RF16. Actualizar pedidos.

Los empleados podrán actualizar el estado de la comanda dependiendo del estado en el que se encuentre, siendo los valores posibles los siguientes:

- Sirviendo.
- Servido.
- Pagado.

RF17. Cancelar comanda.

Los empleados podrán cancelar una comanda en caso de que esta haya sido generada por equivocación.

RF18. Generar cuenta.

Los empleados del local podrán generar la cuenta de una mesa y también podrán actualizar una comanda individual al estado de pagada.

RF19. Limpiar mesa.

Los empleados del local podrán limpiar una mesa, pasando a cerradas todas las comandas y dejándola en estado de libre.

RF20. Generar código QR de mesas.

Los empleados del local podrán generar el código QR de una mesa individual desde la pantalla principal y generar de forma masiva los códigos QR de todas las mesas deseadas por el empleado.

RF21. Asignación de mesas.

Los empleados del local, mediante las opciones del sistema, podrán consultar la lista de mesas que tienen asignadas y modificarlas. Sólo podrán modificar esta lista para asignarse o quitarse mesas a ellos mismos, no pudiendo asignar las mesas a otros empleados.

RF22. Ver gráficas.

Los empleados del local podrán visualizar en gráficas el número de ventas generadas en su local agrupadas por meses en los últimos seis meses.

4.2.4. Requisitos de usuario de cliente.

Estos requisitos son exclusivos de los usuarios que actúen como clientes de locales.

RF23.Buscar locales.

Los clientes podrán, mediante el mapa de la aplicación, buscar un local específico y ubicarlo en el mapa para obtener su información o para, mediante google maps, establecer una ruta para llegar hasta dicho local.

Además, podrán ver todos los locales geolocalizados en la aplicación para elegir el que más se adapte a sus intereses.

RF24. Consultar el estado de las mesas de un local.

El cliente podrá, una vez que haya accedido a la información de un local, consultar el estado de las mesas con el fin de saber si dicho local dispone de mesas libres.

RF25. Consultar la carta de un local.

El cliente podrá, una vez que haya accedido a la información de un local, consultar la carta de la que dispone un local a fin de saber los productos que se ofrecen y el stock de estos antes de acudir al local.

RF26. Contactar con el local.

El cliente podrá, una vez que haya accedido a la información de un local, contactar con el local mediante un email o directamente mediante el número de teléfono.

RF27. Escanear un código QR.

El cliente podrá escanear el código QR de la mesa de un local para disfrutar de las ventajas de tener una mesa asignada.

RF28. Crear una nueva comanda.

El cliente podrá crear comandas siempre y cuando se encuentren logueados y hayan escaneado una mesa mediante el código QR correspondiente.

RF29. Consultar el estado de las comandas.

El cliente sólo podrá consultar los datos de las comandas que han sido generadas por o para ellos, siempre y cuando están logueados en la aplicación y tengan una mesa asignada.

RF30. Cancelar una comanda.

El cliente podrá realizar una petición de cancelación de comanda siempre y cuando estén logueados en la aplicación y tengan una mesa asignada, lo cual no garantiza la cancelación de la misma.

RF31. Generar cuenta.

Los clientes pueden realizar una petición de cuenta, sabiendo el importe exacto de esta y pudiendo elegir entre las distintas modalidades de pago para agilizar el proceso, siempre y cuando estén logueados en la aplicación y tengan una mesa asignada.

RF32. Valorar servicio.

Los clientes pueden valorar el servicio ofrecido por el local una vez que haya sido generada la cuenta mediante un sistema de valoración por estrellas donde cero estrellas es muy descontento y cinco estrellas es muy contento.

4.3. Requisitos funcionales de la aplicación web.

RF01. Registrarse, autenticarse y cerrar sesión.

Cualquier usuario podrá registrarse siempre y cuando el nombre de usuario y el correo electrónico no estén ya ocupados. Además, podrá autenticarse en la aplicación web y cerrar sesión.

RF02. Crear y eliminar usuarios.

El administrador del local podrá crear, leer y eliminar empleados de la base de datos, siempre y cuando el nombre de usuario del empleado no exista ya en la base de datos, pero no podrá modificarlos.

RF03. Crear, leer, modificar y eliminar mesas.

El administrador del local podrán crear y modificar mesas siempre y cuando no exista una mesa con el mismo identificador, además de leer y eliminar mesas de la base de datos.

RF04. Crear, leer, modificar y eliminar espacios del local.

El administrador del local podrán crear y modificar espacios del local siempre y cuando no exista un espacio con el mismo identificador, además de leer y eliminar espacios de la base de datos.

RF05. Crear, leer, modificar y eliminar productos.

El administrador del local podrán crear y modificar productos siempre y cuando no exista un producto con el mismo identificador, además de leer y eliminar productos de la base de datos.

RF06. Crear, leer, modificar y eliminar ingredientes.

El administrador del local podrán crear y modificar ingredientes siempre y cuando no exista un ingrediente con el mismo identificador, además de leer y eliminar ingredientes de la base de datos.

RF07. Crear, leer, modificar y eliminar categorías.

El administrador del local podrán crear y modificar categorías siempre y cuando no exista un categoría con el mismo identificador, además de leer y eliminar categorías de la base de datos.

RF08. Crear, leer, modificar y eliminar sugerencias.

El administrador del local podrán crear y modificar sugerencias siempre y cuando no exista un sugerencia con el mismo identificador, además de leer y eliminar sugerencias de la base de datos.

RF09. Modificar Stock.

El administrador del local podrán consultar y modificar el stock de los productos y los ingredientes y será actualizado en tiempo real.

Los clientes pueden consultar en tiempo real el stock de los productos del local y de los ingredientes que dichos productos llevan.

4.4. Requisitos no funcionales.

RNF01. Apariencia.

La aplicación móvil y web debe de ser legible, fácil de usar, interactiva y llamativa.

RNF02. Confidencialidad.

La información que se almacena en el dispositivo no incluye información de otros usuarios y en la nube no se almacena ni divulga a terceros datos sensibles de los usuarios.

RNF03. Duplicidad.

Los datos en la base de datos no podrán estar duplicados ni existir dos objetos con el mismo identificador o clave primaria.

RNF04. Disponibilidad.

Debido a su ámbito de uso, la aplicación móvil y web debe estar disponible para descargar y usar el 100% del tiempo.

RNF05. Eficiencia.

La aplicación móvil y web debe de funcionar lo mejor posible con los mínimos recursos posibles.

RNF06. Escalabilidad.

La aplicación móvil y web debe estar modulada de forma que se puedan añadir nuevas funcionalidades sin afectar al resto de la aplicación.RNF2. Integridad.Los datos almacenados en el sistema estarán libres de corrupción y serán consistentes.

RNF07. Mantenimiento.

La aplicación móvil y web debe estar modulada de forma que se puedan modificar funcionalidades sin afectar al resto de la aplicación.

RNF08. Resistencia.

La aplicación móvil y web debe de ser capaz de funcionar correctamente a pesar de la aparición de fallos.

RNF09. Robustez.

El sistema debe ser capaz de soportar situaciones excepcionales.

RNF10. Tiempo de respuesta.

Debido a que la aplicación móvil y web va a ser usada en tiempo real, es necesario que el tiempo de respuesta de la aplicación sea de como máximo dos segundos.

RNF11. Usabilidad.

La aplicación móvil y web debe de ser fácil de instalar y de utilizar, para asegurar la facilidad de uso en los locales y clientes con dificultades.

5. Diseño.

5.1. Introducción.

5.1.1. Diagramas de caso de uso

Un caso de uso representa una interacción entre un actor y la aplicación y se utiliza para capturar los requisitos funcionales de la aplicación.[47]

5.1.2. Diagramas de secuencia.

Un diagrama de secuencia muestra una interacción, que representa la secuencia de mensajes entre instancias de clases, componentes, subsistemas o actores.[48]

5.2. Diagramas de casos de uso de la aplicación móvil. (Figura 12.a)

A continuación se han modelado los casos de uso completos de la aplicación móvil, así como los escenarios basados en los requisitos funcionales anteriormente descritos.

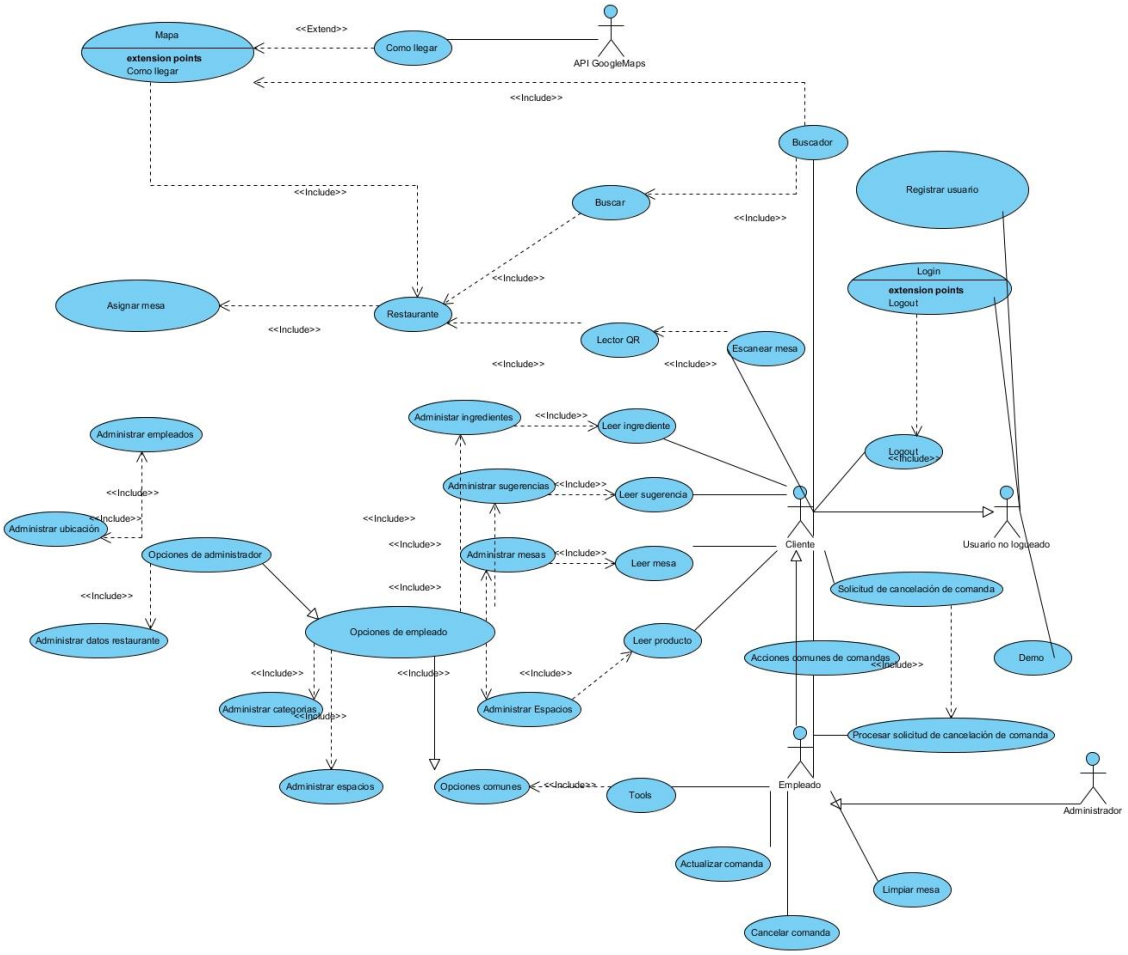


Figura 12.a: Diagrama de casos de uso general.

CU01. Funciones principales. (Figura 12)

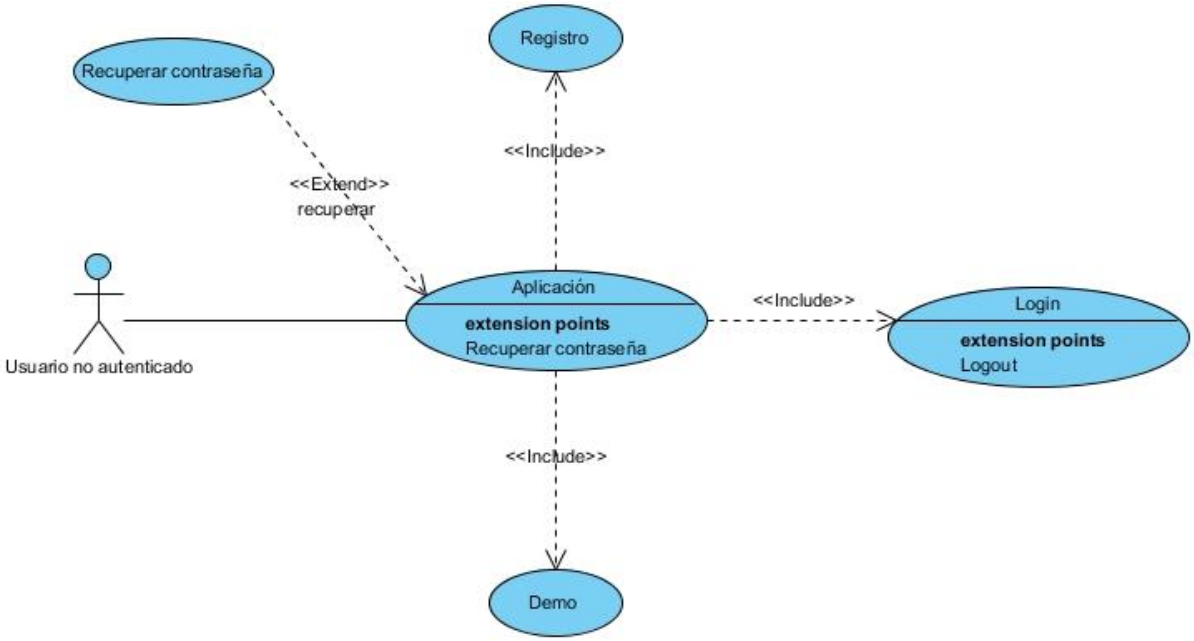


Figura 12: Diagrama de casos de uso de funciones principales.

Caso de uso:	Funciones principales	**CU01**
Descripción. El usuario puede identificarse, registrarse, usar la versión de prueba de la aplicación o recuperar contraseña.		
Actores. Usuario no identificado.		
Precondiciones. El usuario no debe de estar logueado.		
Flujo natural. 1. El usuario desde la pantalla de inicio introduce sus datos y pulsa sobre login. 2. El usuario desde la pantalla de inicio pulsa sobre “registrar”. 2.1. El usuario elige si desea registrarse como cliente o como empresa. 2.2. El usuario introduce los datos solicitados y confirma. 2.3. El sistema regresa al usuario a la pantalla de inicio. 3. El usuario desde la pantalla de inicio pulsa sobre “demo”. 3.1. El sistema muestra una versión limitada de la aplicación. 4. El usuario pulsa sobre “Recuperar contraseña”. 4.1. El sistema solicita que se introduzca el nombre de usuario o el email de la cuenta. 4.2. El usuario introduce uno de estos datos y el sistema genera y muestra una nueva contraseña.		

Flujo alternativo. 1.1. La aplicación mostrará un mensaje de error si el usuario ha introducido los campos incorrectamente. 2.2.1. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 2.2.2 La aplicación mostrará un mensaje de error si el usuario introduce un nombre de usuario o correo ya registrados en la aplicación. 4.1 La aplicación mostrará un mensaje de error si el nombre de usuario ni email existen.
Post-condiciones. El usuario es registrado o logueado en el sistema.

CU02. Registrarse. (Figura 13)

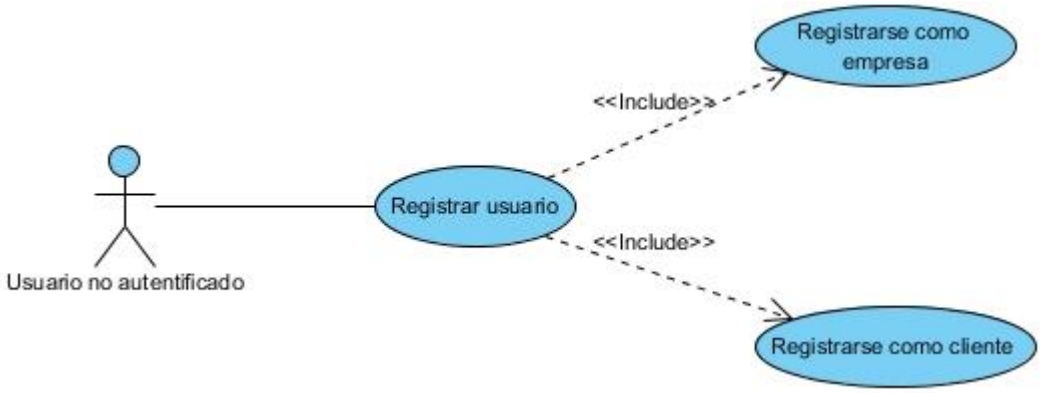


Figura 13: Diagrama de casos de uso de registrarse.

Caso de uso:	Registrar usuario	**CU2**
Descripción. Permite crear un usuario nuevo en la aplicación.		
Actores. Usuario no registrado.		
Precondiciones. El usuario no debe de estar logueado ni registrado.		
Flujo natural. 1. El usuario desde la pantalla de inicio pulsa sobre “registrar”. 2. El usuario elige si desea registrarse como cliente o como empresa. 3. El usuario introduce los datos solicitados y confirma. 4. El sistema regresa al usuario a la pantalla de inicio.		
Flujo alternativo. 3.1. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 3.2. La aplicación mostrará un mensaje de error si el usuario introduce un nombre		

de usuario o correo ya registrados en la aplicación.
Post-condiciones. El usuario es registrado en el sistema.

CU03. Login y Logout. (Figura 14)

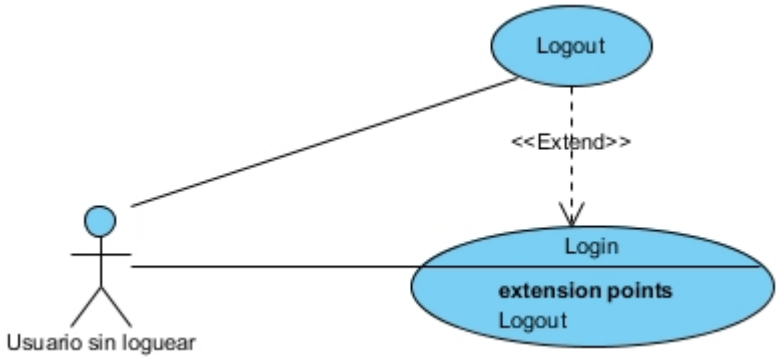


Figura 14: Diagrama de casos de uso de login y logout.

Caso de uso:	Login y Logout.	**CU3**
Descripción. Permite que un usuario inicie sesión en la aplicación y cierre sesión.		
Actores. Usuario no logueado.		
Precondiciones. El usuario no debe de estar logueado.		
Flujo natural. 1. El usuario introduce los datos solicitados en la pantalla de inicio. 2. El usuario pulsa sobre “login”. 3. El usuario desde el menú lateral pulsa sobre “cerrar sesión”. El sistema regresa al usuario a la pantalla de inicio.		
Flujo alternativo. 3.1. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 3.2. La aplicación mostrará un mensaje de error si el usuario introduce un nombre de usuario o correo ya registrados en la aplicación.		
Post-condiciones. El usuario es logueado o deslogueado del sistema.		

CU04. Opciones del sistema. (Figura 15)

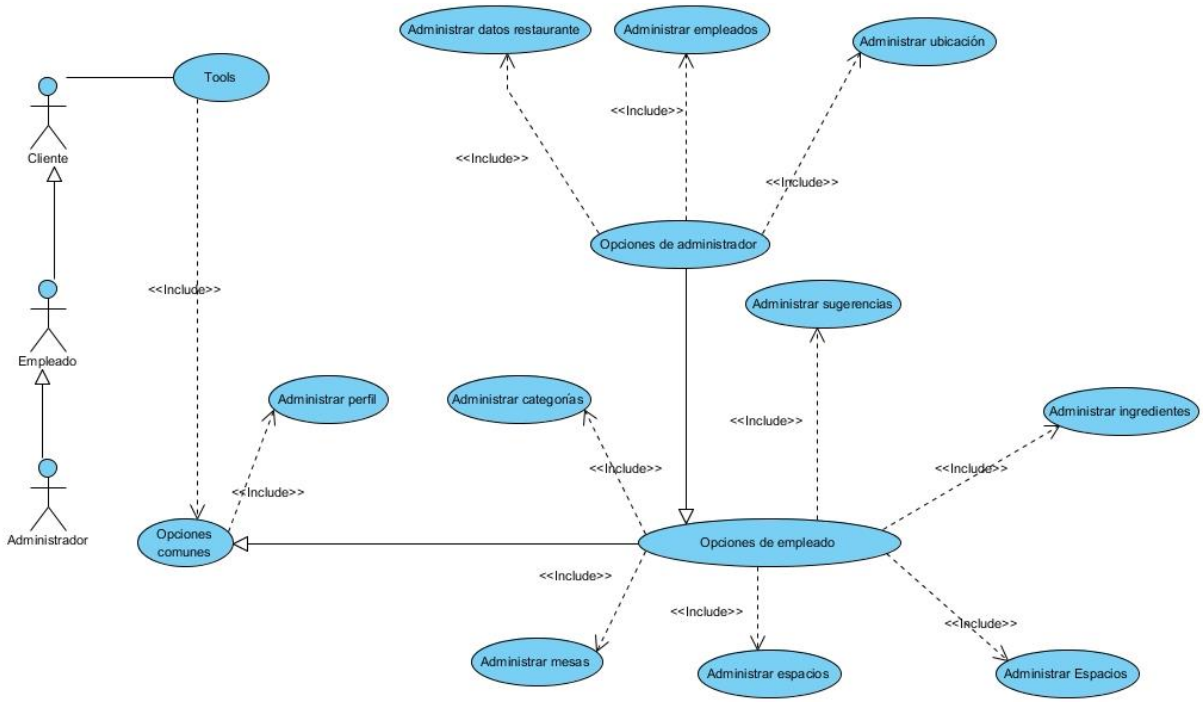


Figura 15: Diagrama de casos de uso de opciones del sistema.

Caso de uso:	Opciones del sistema.	**CU4**
Descripción. El cliente puede modificar sus datos personales. El empleado, además de poder hacer lo mismo que el cliente, puede administrar mesas, espacios, productos, ingredientes, categorías y sugerencias. El administrador puede, además de lo mismo que puede hacer el empleado, actualizar la ubicación del restaurante en el mapa y actualizar sus datos, así como administrar empleados.		
Actores. Cliente. Empleado. Administrador.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El usuario, desde cualquier pantalla, accede al menú lateral desde el botón situado en la esquina superior izquierda, y pulsa sobre “Tools”.		
Flujo alternativo.		
Post-condiciones. El usuario es registrado en el sistema.		

CU05. Administrar ubicación. (Figura 16)

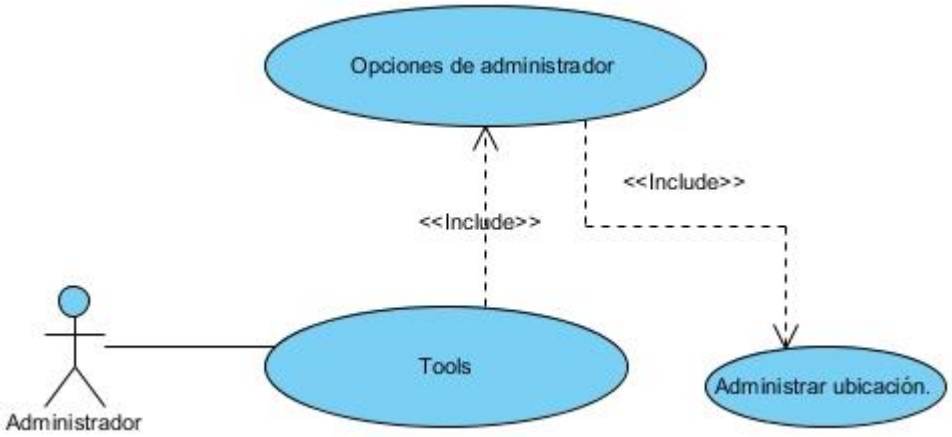


Figura 16: Diagrama de casos de uso de administrar ubicación.

Caso de uso:	Administrar ubicación.	**CU6**
Descripción. Permite que el administrador pueda actualizar la ubicación del local.		
Actores. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar ubicación”. 2. El usuario selecciona en el mapa la ubicación del local y confirma la elección.		
Flujo alternativo.		
Post-condiciones. La ubicación del local ha sido actualizada.		

CU6. Crear y eliminar usuarios. (Figura 17)

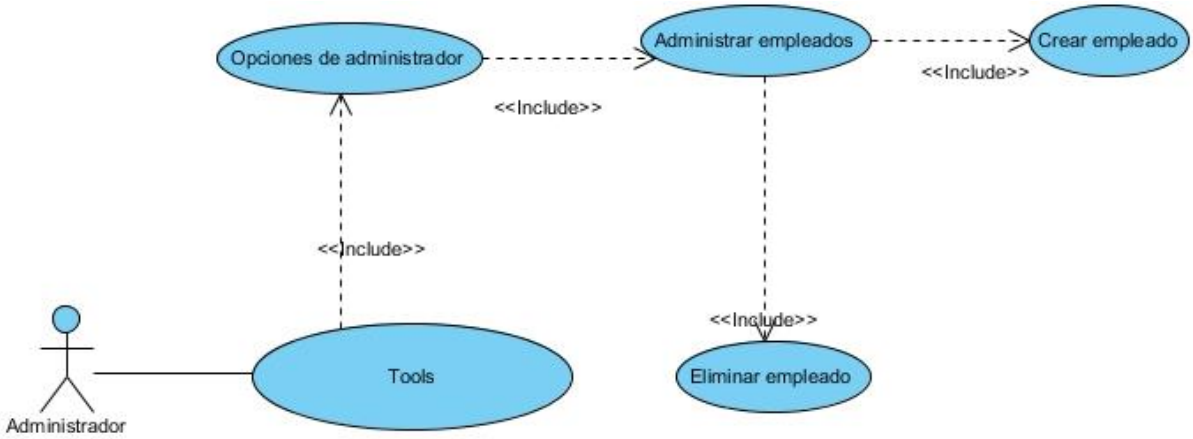


Figura 17: Diagrama de casos de uso de crear y eliminar usuarios.

Caso de uso:	Crear y eliminar empleados	**CU5**
Descripción. El administrador debe poder crear y eliminar empleados de su local en el sistema.		
Actores. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “administrar empleados”. 1.1. El administrador pulsa sobre el botón de “+” para insertar un nuevo empleado. 1.2. El administrador introduce los datos del nuevo empleado, excepto la contraseña que será introducida por el empleado la primera vez que inicie sesión. 2. El usuario accede a los ajustes del sistema y selecciona “administrar empleados”. 2.1. El administrador selecciona un empleado y pulsa sobre “Eliminar”. 2.2. El sistema muestra un mensaje de confirmación y el administrador confirma que desea realizar la acción.		
Flujo alternativo. 1.2. El nombre de usuario que pretende crear en el sistema ya existe en la base de datos.		
Post-condiciones. 1. El empleado es creado. 2. El empleado es eliminado.		

CU07. Crear, leer, modificar y eliminar mesas. (Figura 18)

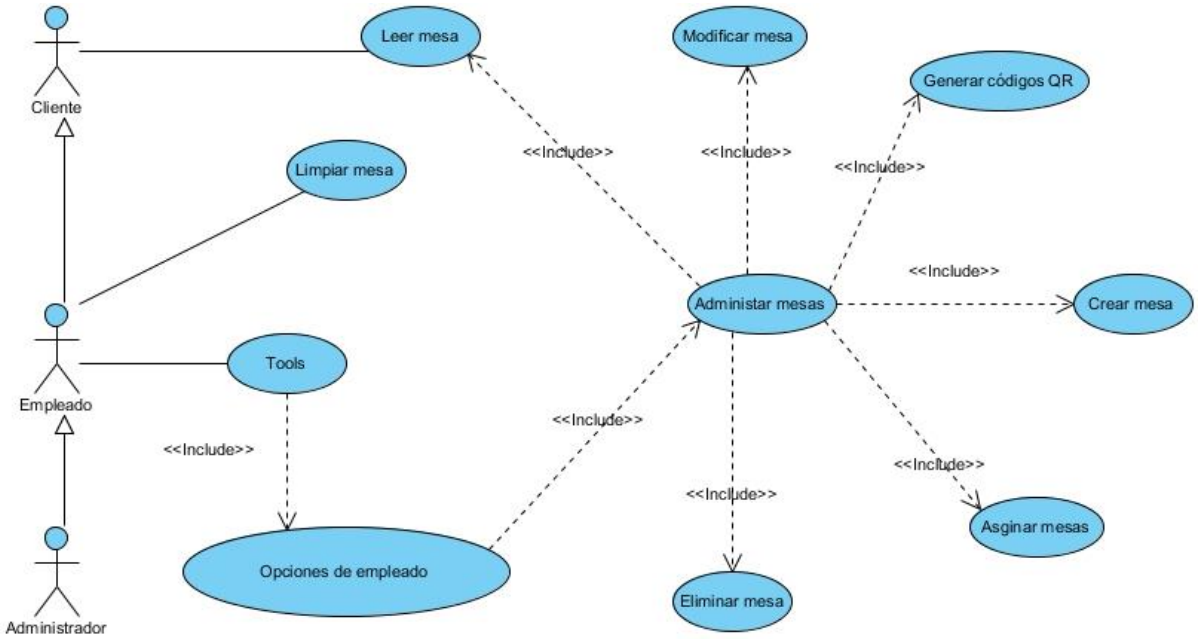


Figura 18: Diagrama de casos de uso de crear, leer, modificar y eliminar mesas.

Caso de uso:	CRUD Mesas.	**CU7**
Descripción. El empleado debe poder manipular mesas, pudiendo crear, leer, modificar y eliminar una mesa. El cliente puede ver las mesas de un local.		
Actores. Usuario cliente. Usuario empleado. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar mesas”. 1.1. El usuario pulsa sobre el botón de “+” para insertar una nueva mesa. 1.2. El usuario introduce los datos de la nueva mesa, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona administrar mesas. 2.1. El usuario selecciona una mesa para ver sus datos. 2.2. El usuario modifica los datos de la mesa y confirma la acción. 3. El usuario accede a los ajustes del sistema y selecciona “Administrar mesas”. 3.1. El administrador selecciona una mesa y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.		

4. El usuario cliente selecciona un local y el sistema muestra las mesas asociadas a dicho local.
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. La mesa es creada en la base de datos. 2. La mesa es actualizada en la base de datos. 3. La mesa es eliminada de la base de datos.

CU08. Crear, leer, modificar y eliminar espacios. (Figura 19)

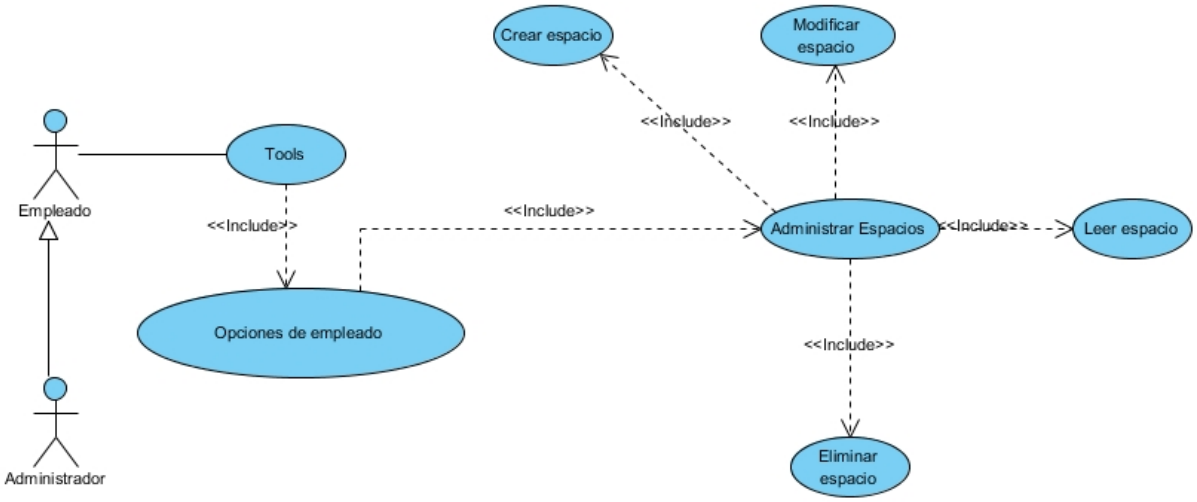


Figura 19: Diagrama de casos de uso de crear, leer, modificar y eliminar espacios.

Caso de uso:	CRUD Espacios.	**CU8**
Descripción. El empleado debe poder manipular espacios, pudiendo crear, leer, modificar y eliminar un espacio.		
Actores. Usuario empleado. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar espacios”.		

1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo espacio. 1.2. El usuario introduce los datos del nuevo espacio, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona “Administrar espacios”. 2.1. El usuario selecciona un espacio para ver sus datos. 2.2. El usuario modifica los datos del espacio y confirma la acción. 3. El usuario accede a los ajustes del sistema y selecciona “administrar espacios”. 3.1. El administrador selecciona un espacio y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. El espacio es creado en la base de datos. 2. El espacio es actualizado en la base de datos. 3. El espacio es eliminado de la base de datos.

CU09. Crear, leer, modificar y eliminar productos. (Figura 20)

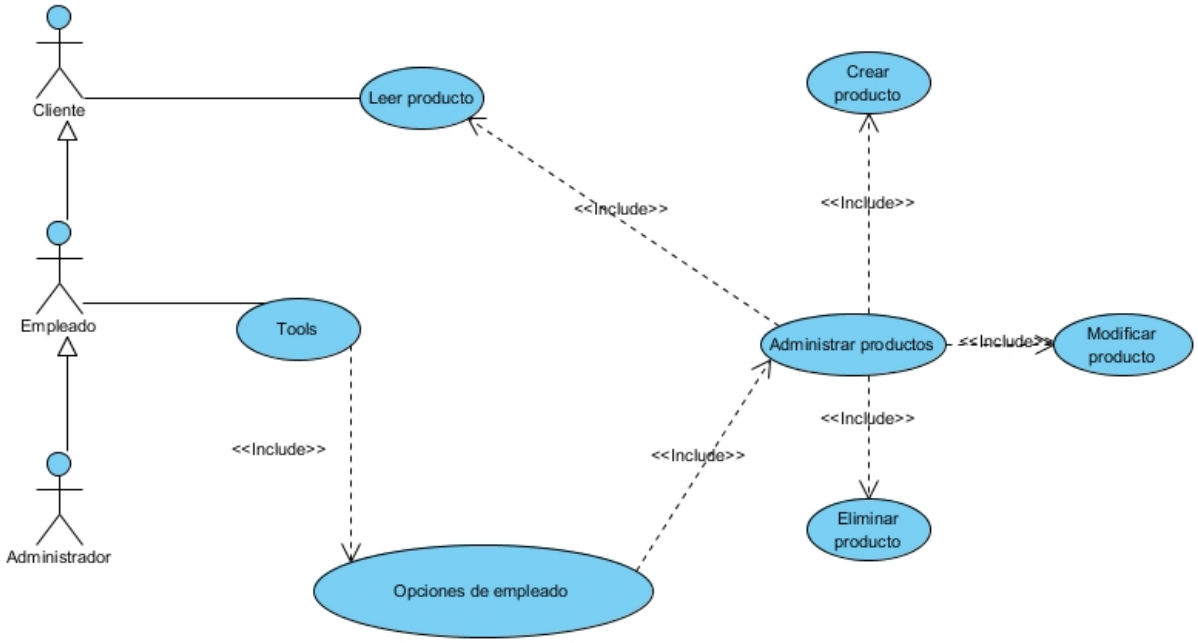


Figura 20: Diagrama de casos de uso de crear, leer, modificar y eliminar productos.

Caso de uso:	CRUD Productos.	**CU9**

Descripción.
El empleado debe poder manipular productos, pudiendo crear, leer, modificar y eliminar un producto.
El cliente puede ver los productos creados en un local.

Actores.
Usuario cliente.
Usuario empleado.
Usuario administrador.

Precondiciones.
El usuario debe de estar.

Flujo natural.

1. El usuario accede a los ajustes del sistema y selecciona “Administrar productos”.
   - 1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo producto.
   - 1.2. El usuario introduce los datos del nuevo producto, y confirma los datos.
2. El usuario accede a los ajustes del sistema y selecciona administrar producto.
   - 2.1. El usuario selecciona un producto para ver sus datos.
   - 2.2. El usuario modifica los datos del producto y confirma la acción.
3. El usuario accede a los ajustes del sistema y selecciona “Administrar productos”.
   - 3.1. El administrador selecciona un producto y pulsa sobre “Eliminar”.
   - 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.
4. El usuario cliente mediante el buscador de locales selecciona un local.
   - 4.1. El usuario accede al menú lateral “Stock”, donde puede ver los productos del local.

Flujo alternativo.
1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco.
1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.

Post-condiciones.

1. El producto es creado en la base de datos.
2. El producto es actualizado en la base de datos.
3. El producto es eliminado de la base de datos.

CU10. Crear, leer, modificar y eliminar ingredientes. (Figura 21)

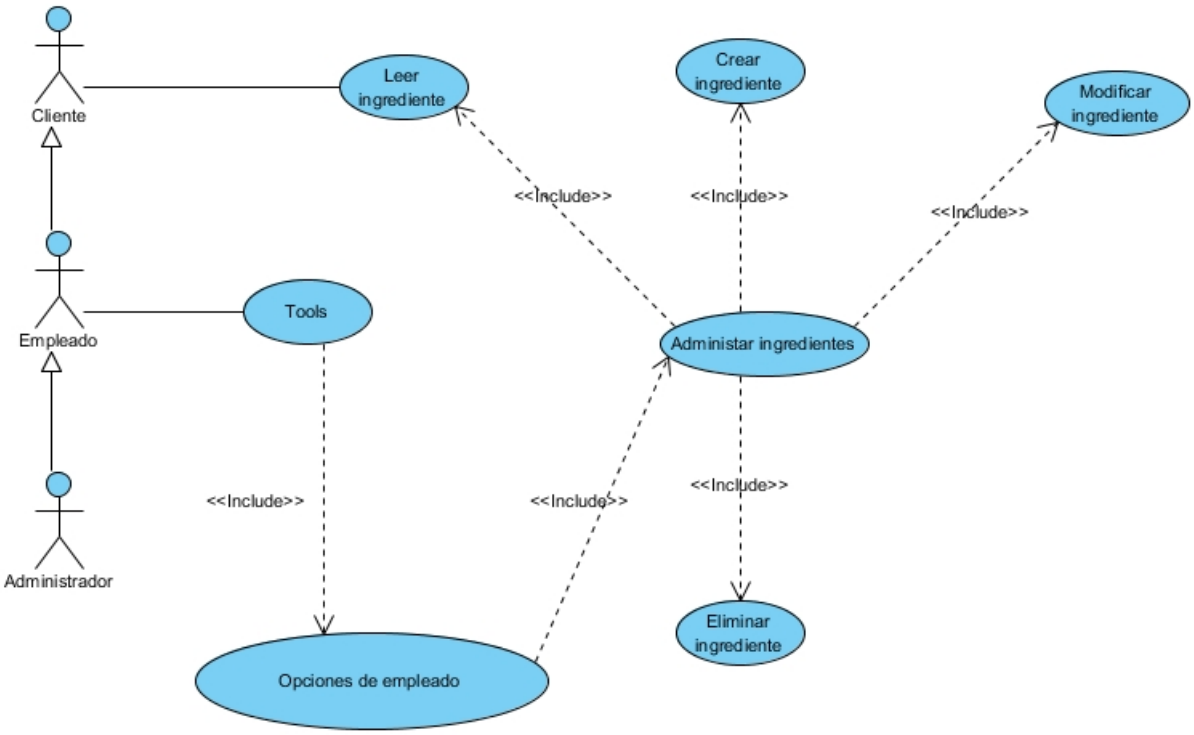


Figura 21: Diagrama de casos de uso de crear, leer, modificar y eliminar ingredientes.

Caso de uso:	CRUD Ingredientes.	**CU10**
Descripción. El empleado debe poder manipular ingredientes, pudiendo crear, leer, modificar y eliminar un ingrediente. El cliente puede ver los ingredientes creados en un local.		
Actores. Usuario cliente. Usuario empleado. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar ingredientes”. 1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo ingrediente. 1.2. El usuario introduce los datos del nuevo ingrediente, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona administrar ingrediente. 2.1. El usuario selecciona un ingrediente para ver sus datos. 2.2. El usuario modifica los datos del ingrediente y confirma la acción.		

3. El usuario accede a los ajustes del sistema y selecciona “Administrar ingredientes”. 3.1. El administrador selecciona un ingrediente y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción. 4. El usuario cliente mediante el buscador de locales selecciona un local. 4.1. El usuario accede al menú lateral “Stock”, donde puede ver los productos del local.
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. El ingrediente es creado en la base de datos. 2. El ingrediente es actualizado en la base de datos. 3. El ingrediente es eliminado de la base de datos.

CU11. Crear, leer, modificar y eliminar categorías. (Figura 21.b)

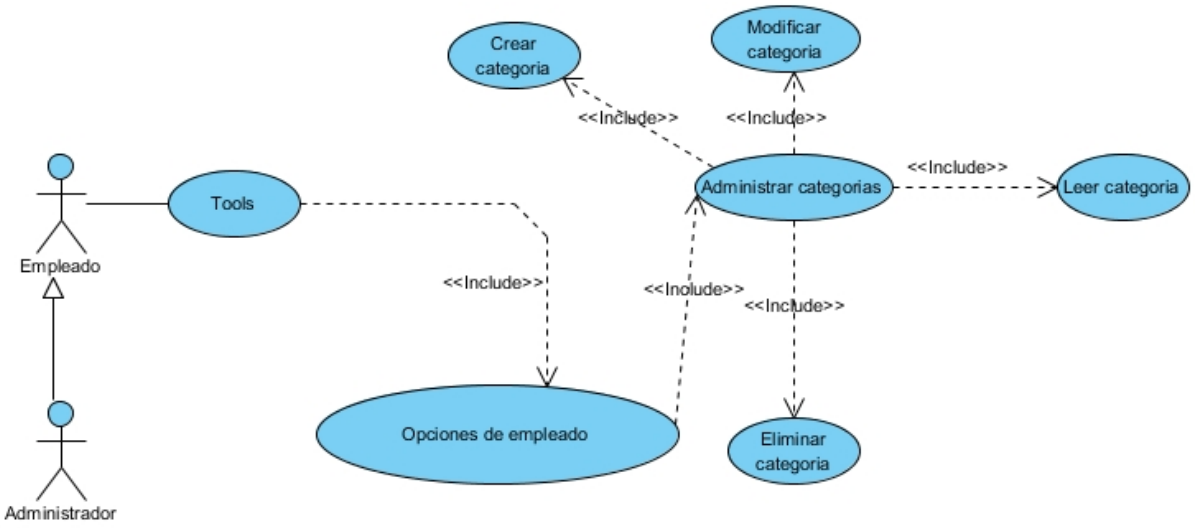


Figura 21.b: Diagrama de casos de uso de crear, leer, modificar y eliminar categorías.

Caso de uso:	CRUD Categorías.	**CU11**
Descripción. El empleado debe poder manipular categorías, pudiendo crear, leer, modificar y eliminar una categoría.		
Actores. Usuario empleado. Usuario administrador.		

Precondiciones.
El usuario debe de estar logueado como empleado o como empresa.

Flujo natural.

1. El usuario accede a los ajustes del sistema y selecciona “Administrar categorías”.
   1.1. El usuario pulsa sobre el botón de “+” para insertar una nueva categoría.
   1.2. El usuario introduce los datos de la nueva categoría, y confirma los datos.
2. El usuario accede a los ajustes del sistema y selecciona administrar categorías.
   2.1. El usuario selecciona una mesa para ver sus datos.
   2.2. El usuario modifica los datos de la categoría y confirma la acción.
3. El usuario accede a los ajustes del sistema y selecciona “administrar categorías”.
   3.1. El administrador selecciona una mesa y pulsa sobre “Eliminar”.
   3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.

Flujo alternativo.
1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco.
1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.

Post-condiciones.

1. La categoría es creada en la base de datos.
2. La categoría es actualizada en la base de datos.
3. La categoría es eliminada de la base de datos.

CU12. Crear, leer, modificar y eliminar sugerencias. (Figura 22)

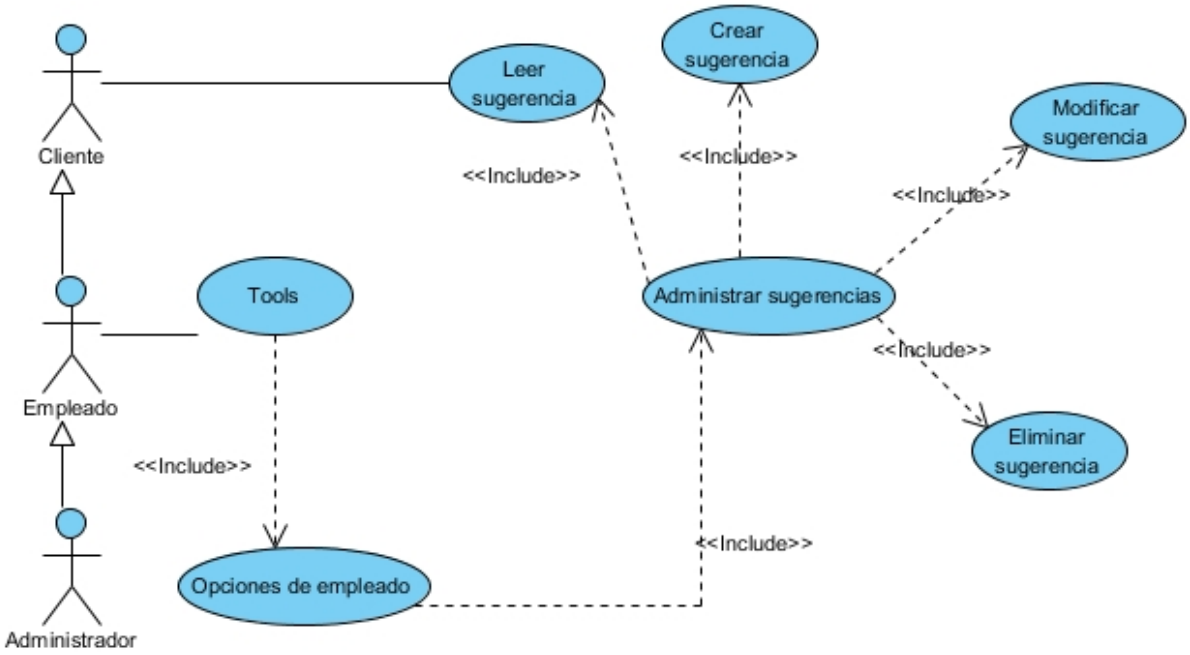


Figura 22: Diagrama de casos de uso de crear, leer, modificar y eliminar sugerencias.

Caso de uso:	CRUD Sugerencias.	**CU12**
Descripción. El empleado debe poder manipular sugerencias, pudiendo crear, leer, modificar y eliminar una sugerencia.		
Actores. Usuario empleado. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar sugerencias”. 1.1. El usuario pulsa sobre el botón de “+” para insertar una nueva sugerencia. 1.2. El usuario introduce los datos de la nueva sugerencia, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona “Administrar sugerencias. 2.1. El usuario selecciona una sugerencia para ver sus datos. 2.2. El usuario modifica los datos de la sugerencia y confirma la acción. 2.3. El usuario puede modificar si la sugerencia es visible a los clientes o no.		

3. El usuario accede a los ajustes del sistema y selecciona “administrar categorías”. 3.1. El administrador selecciona una sugerencia y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. La sugerencia es creada en la base de datos. 2. La sugerencia es actualizada en la base de datos. 2.1. La visualización de la sugerencia es actualizada. 3. La sugerencia es eliminada de la base de datos.

CU13. Crear, leer, modificar y eliminar comanda. (Figura 23)

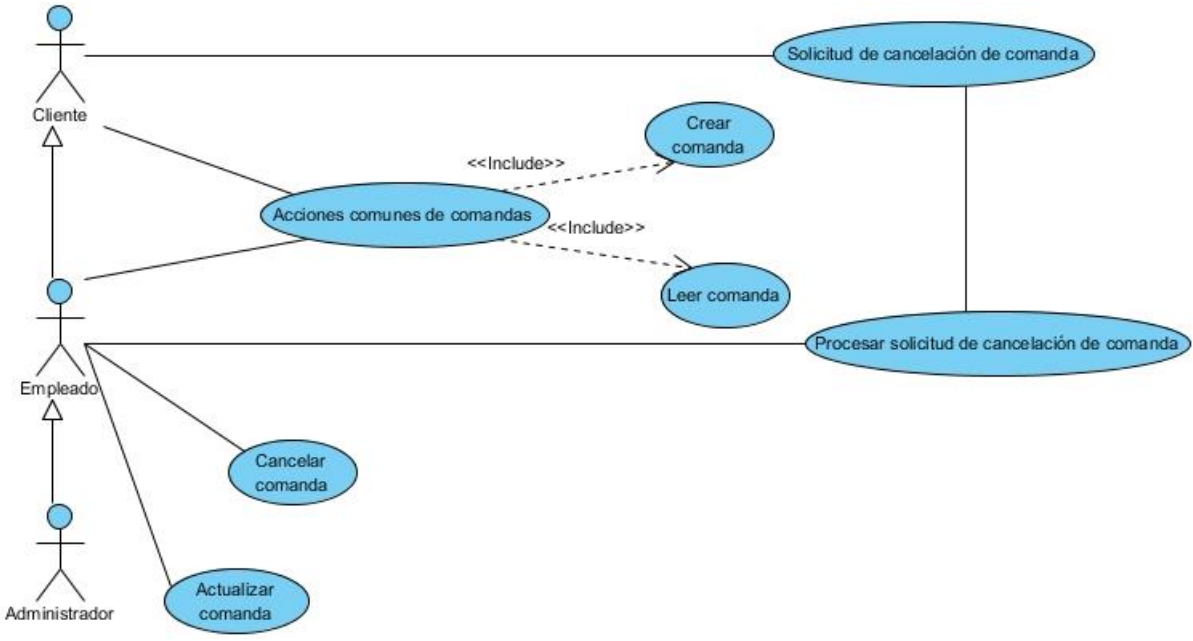


Figura 23: Diagrama de casos de uso de crear, leer, modificar y eliminar comandas.

Caso de uso:	CRUD Comandas.	**CU13**
Descripción. El empleado debe poder manipular comandas, pudiendo crear, leer, modificar y cancelar una comanda.		
Actores. Usuario empleado. Usuario administrador.		

Precondiciones. El usuario debe de estar logueado.
Flujo natural. 1. El usuario desde la pantalla principal selecciona una mesa y en la pestaña de “Nueva comanda” podrá crear una comanda nueva insertando los datos correspondientes. 2. El usuario desde la pantalla principal selecciona una mesa y en la pestaña de “Estado comanda” podrá ver las comandas generadas y mediante el botón “Cancelar” se podrá cancelar la comanda. 3. El usuario desde el menú lateral selecciona “Comandas”. 3.1. En la lista de comandas selecciona una comanda para visualizar. 3.2. Una vez abierta la comanda, podrá actualizar su estado o cancelarla.
Flujo alternativo.
Post-condiciones. 1. La comanda es creada en la base de datos. 2. La comanda es actualizada en la base de datos. 3. La comanda es cancelada en la base de datos.

CU14. Generar cuenta. (Figura 24)

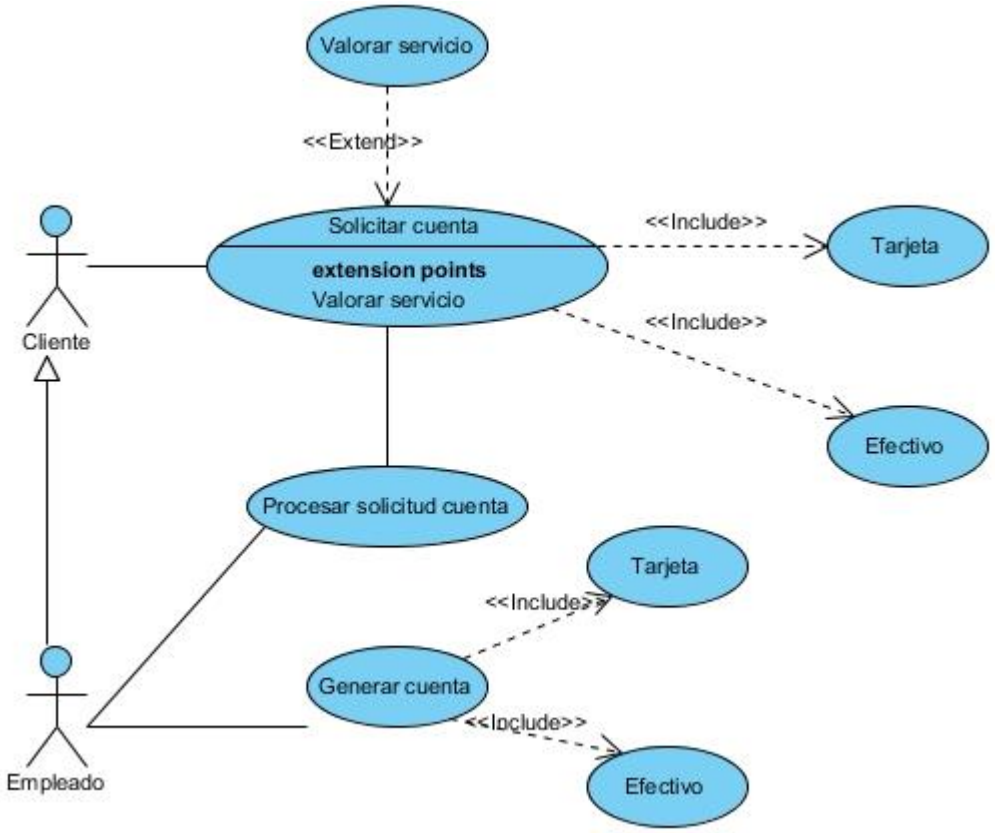


Figura 24: Diagrama de casos de uso de generar cuenta.

Caso de uso:	Generar cuenta.	**CU14**
Descripción. El cliente debe poder solicitar la cuenta, bien por tarjeta o bien en efectivo, a un empleado del local. El empleado debe poder generar la cuenta de una mesa.		
Actores. Usuario cliente. Usuario empleado. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El usuario cliente desde el estado de comandas de la mesa que tiene asignada, pulsa sobre ”Generar cuenta”. 1.1. El sistema le pregunta si desea pagar con tarjeta o si, por el contrario, prefiere pagar en efectivo. 1.2. El sistema envía una notificación al empleado con la mesa asignada		

para que prepare la cuenta. 2. El empleado, desde el estado de comandas de una mesa, pulsa sobre “Generar cuenta”. 2.1. El sistema le pregunta si desea pagar con tarjeta o si, por el contrario, prefiere pagar en efectivo. 2.2. El sistema guarda en la nube las comandas, pone la mesa en estado de “Libre” y elimina el cliente asociado a la mesa.
Flujo alternativo. 1.2. El usuario puede cancelar la generación de la cuenta.
Post-condiciones. 1. La cuenta es generada.

CU15. Buscar locales. (Figura 25)

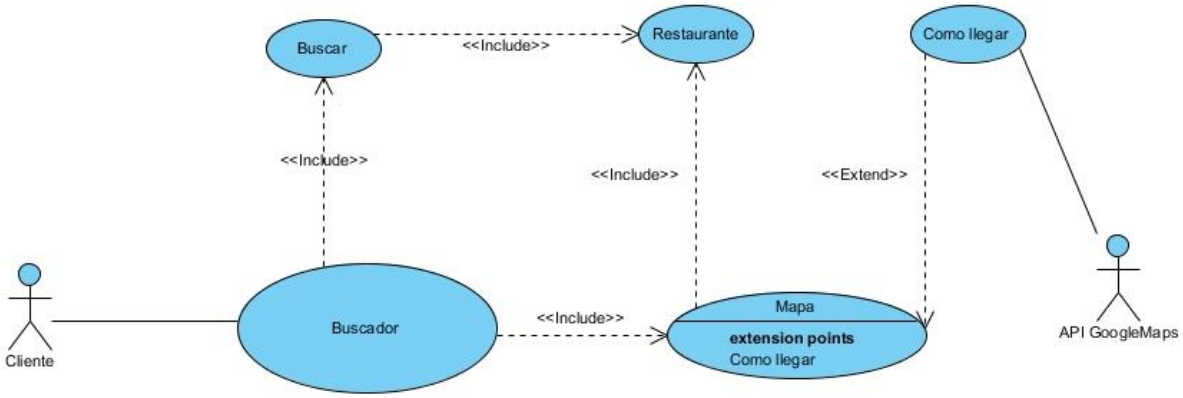


Figura 25: Diagrama de casos de uso de buscar locales.

Caso de uso:	Buscar locales.	**CU15**
Descripción. El cliente debe poder encontrar, mediante un buscador o el mapa, los locales localizados en la aplicación, así como acceder a su información y contenido.		
Actores. Usuario cliente.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El usuario desde el menú lateral, accede al buscador pulsando sobre “Mapa”. 2. El usuario introduce en el buscador el nombre del local. 2.1. El sistema redirige el mapa a la ubicación del local. 3. El cliente selecciona el local. 4. El sistema le pregunta si desea ir a la información del local 4.1. Si el cliente acepta, puede ver el contenido del local (mesas y		

productos) 4.2. Si el cliente cancela, puede acceder a google maps con la ubicación del local.
Flujo alternativo.
Post-condiciones. 1. El sistema muestra información del local.

CU16. Escanear mesa. (Figura 26)

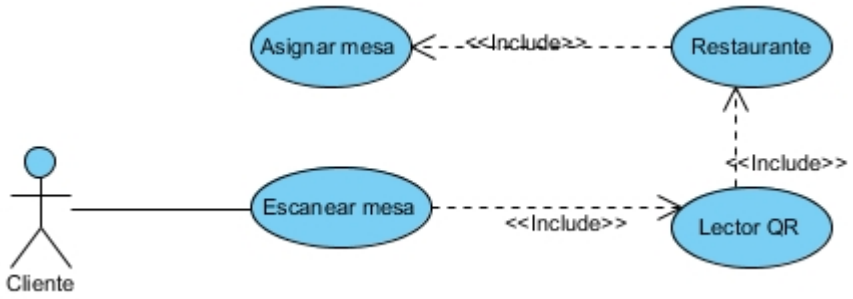


Figura 26: Diagrama de casos de uso de escanear mesa.

Caso de uso:	Escanear mesa.	**CU16**
Descripción. El cliente debe poder escanear una mesa mediante un código qr para ser asignada a dicho cliente y gozar de las funcionalidades de la aplicación.		
Actores. Usuario cliente.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El cliente escanea un código QR mediante el lector QR de la aplicación. 2. El sistema reconoce el código QR y le asigna la mesa correspondiente a dicho código.		
Flujo alternativo. 1.1. El sistema no reconoce el código QR y genera un mensaje de error.		
Post-condiciones. 1. El cliente puede empezar a crear comandas.		

CU17. Valorar servicios. (Figura 27)

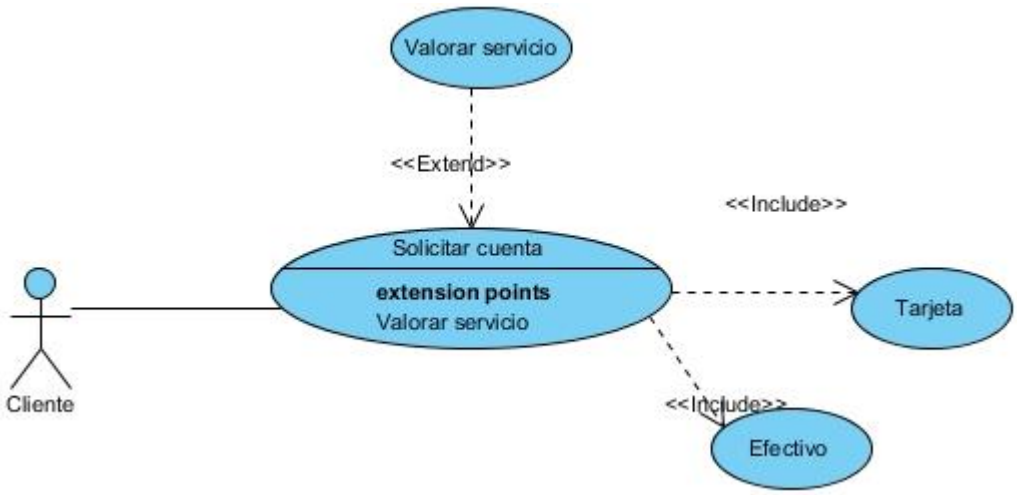


Figura 27: Diagrama de casos de uso de valorar servicios.

Caso de uso:	Valorar servicios.	**CU17**
Descripción. El cliente, al finalizar su estancia en el local, debe poder valorar el servicio dado por el local, mediante un sistema de estrellas y un comentario opcional.		
Actores. Usuario cliente.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El cliente solicita la cuenta. 2. El sistema abre un cuadro de diálogo que muestra al usuario. 3. El cliente introduce la puntuación al local y un comentario opcional. 4. El sistema guarda la valoración de forma anónima en la base de datos global.		
Flujo alternativo. 3.1. El cliente no realiza la valoración.		
Post-condiciones. 1. La valoración es almacenada en la base de datos global.		

5.3. Diagramas de casos de uso de la aplicación web.

A continuación se han modelado los casos de uso completos de la aplicación web, así como los escenarios basados en los requisitos funcionales anteriormente descritos.

CU01. Registrarse. (Figura 28)

Figura 28: Diagrama de casos de uso de registrarse.

Caso de uso:	Registrar usuario	**CU01**
Descripción. Permite crear un usuario nuevo en la aplicación.		
Actores. Usuario no registrado.		
Precondiciones. El usuario no debe de estar logueado ni registrado. El usuario que se crea debe estar registrado en la aplicación web.		
Flujo natural. 5. El usuario desde la pantalla de inicio pulsa sobre “registrar”. 6. El usuario elige si desea registrarse como cliente o como empresa. 7. El usuario introduce los datos solicitados y confirma. 8. El sistema regresa al usuario a la pantalla de inicio.		
Flujo alternativo. 3.1. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 3.2. La aplicación mostrará un mensaje de error si el usuario introduce un nombre de usuario o correo ya registrados en la aplicación.		
Post-condiciones. El usuario es registrado en el sistema.		

CU02. Login y Logout. (Figura 29)

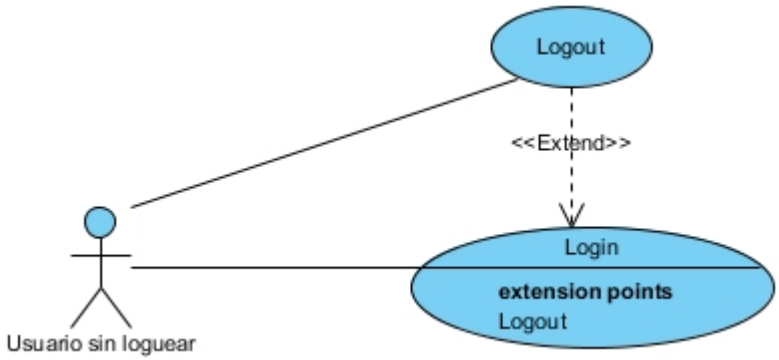


Figura 29: Diagrama de casos de uso de login y logout.

Caso de uso:	Login y Logout.	**CU02**
Descripción. Permite que un usuario inicie sesión en la aplicación y cierre sesión.		
Actores. Usuario no logueado.		
Precondiciones. El usuario no debe de estar logueado.		
Flujo natural. 1. El usuario introduce los datos solicitados en la pantalla de inicio. 2. El usuario pulsa sobre “login”. 3. El usuario desde el menú lateral pulsa sobre “cerrar sesión”. 4. El sistema regresa al usuario a la pantalla de inicio.		
Flujo alternativo. 3.1. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 3.2. La aplicación mostrará un mensaje de error si el usuario introduce un nombre de usuario o correo ya registrados en la aplicación.		
Post-condiciones. El usuario es logueado o deslogueado del sistema.		

CU03. Crear y eliminar usuarios. (Figura 3)

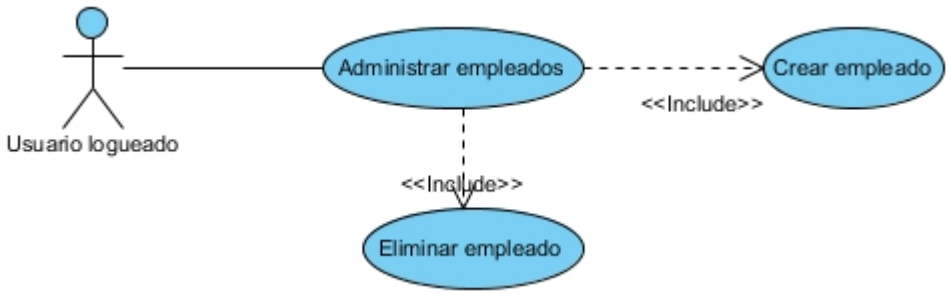


Figura 30: Diagrama de casos de uso de crear y eliminar usuarios.

Caso de uso:	Crear y eliminar empleados	**CU03**
Descripción. El usuario debe poder crear y eliminar empleados de su local en el sistema.		
Actores. Usuario logueado.		
Precondiciones. El usuario debe de estar logueado como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “administrar empleados”. 1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo empleado. 1.2. El usuario introduce los datos del nuevo empleado, excepto la contraseña que será introducida por el empleado la primera vez que inicie sesión. 2. El usuario accede a los ajustes del sistema y selecciona “administrar empleados”. 2.1. El usuario selecciona un empleado y pulsa sobre “Eliminar”. 2.2. El sistema muestra un mensaje de confirmación y el administrador confirma que desea realizar la acción.		
Flujo alternativo. 1.2. El nombre de usuario que pretende crear en el sistema ya existe en la base de datos.		
Post-condiciones. 3. El empleado es creado. 4. El empleado es eliminado.		

CU04. Crear, leer, modificar y eliminar mesas. (Figura 30)

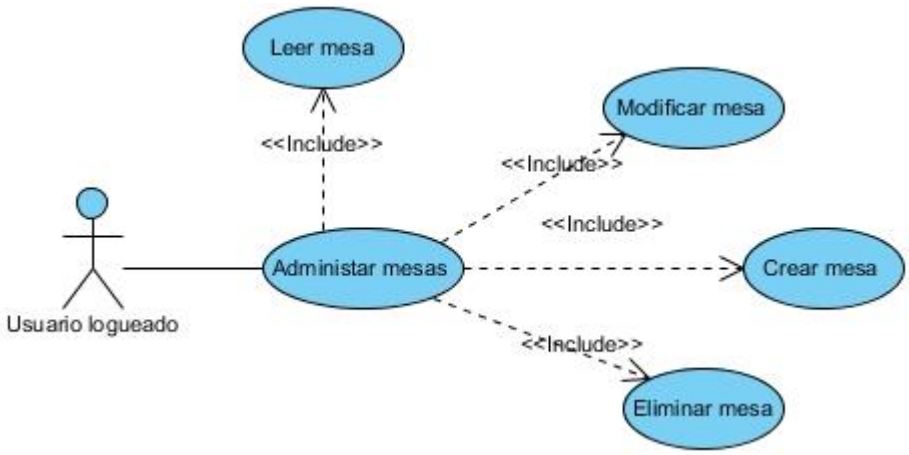


Figura 31: Diagrama de casos de uso de crear, leer, modificar y eliminar mesas.

Caso de uso:	CRUD Mesas.	**CU04**
Descripción. El usuario debe poder manipular mesas, pudiendo crear, leer, modificar y eliminar una mesa.		
Actores. Usuario logueado.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar mesas”. 1.1. El usuario pulsa sobre el botón de “+” para insertar una nueva mesa. 1.2. El usuario introduce los datos de la nueva mesa, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona administrar mesas. 2.1. El usuario selecciona una mesa y pulsa “Editar”. 2.2. El usuario modifica los datos de la mesa y confirma la acción. 3. El usuario accede a los ajustes del sistema y selecciona “Administrar mesas”. 3.1. El usuario selecciona una mesa y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.		
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.		
Post-condiciones.		

1. La mesa es creada en la base de datos. 2. La mesa es actualizada en la base de datos. 3. La mesa es eliminada de la base de datos.

CU05. Crear, leer, modificar y eliminar espacios. (Figura 31)

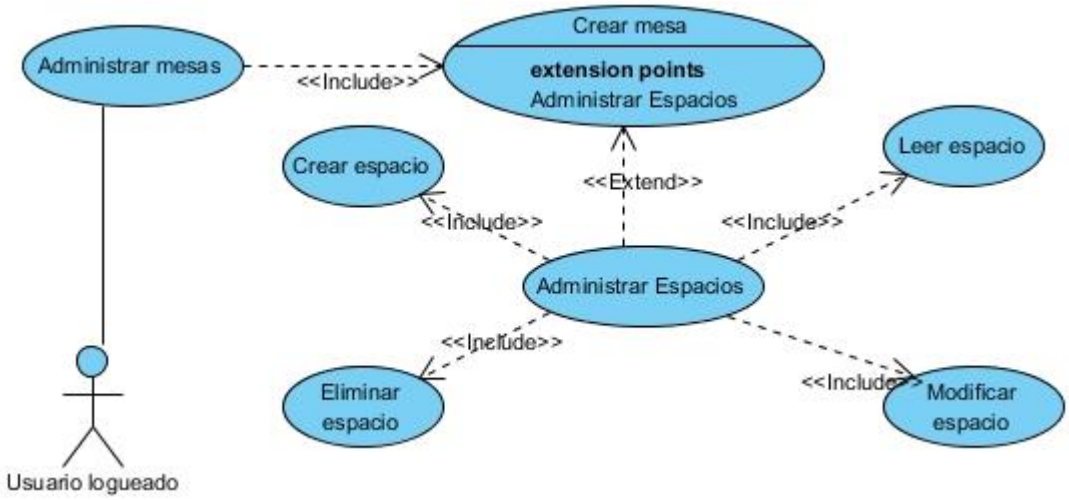


Figura 32: Diagrama de casos de uso de crear, leer, modificar y eliminar espacios.

Caso de uso:	CRUD Espacios.	**CU05**
Descripción. El usuario debe poder manipular espacios, pudiendo crear, leer, modificar y eliminar un espacio.		
Actores. Usuario.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar mesas”. 1.1. El usuario pulsa sobre el botón de “+” para insertar una nueva mesa. 1.2. El usuario pulsa sobre “+” en la sección de “Espacio”. 1.3. Pulsa sobre el botón de “+” para insertar un nuevo espacio. 1.4. El usuario introduce los datos del nuevo espacio, y confirma los datos. 2. De la misma forma que para crear un espacio, el usuario accede a administrar espacios. 2.1. El usuario selecciona un espacio y pulsa “Editar”. 2.2. El usuario modifica los datos del espacio y confirma la acción. 3. De la misma forma que para crear un espacio, el usuario accede a administrar espacios.		

3.1. El usuario selecciona un espacio y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. El espacio es creado en la base de datos. 2. El espacio es actualizado en la base de datos. 3. El espacio es eliminado de la base de datos.

CU06. Crear, leer, modificar y eliminar productos. (Figura 32)

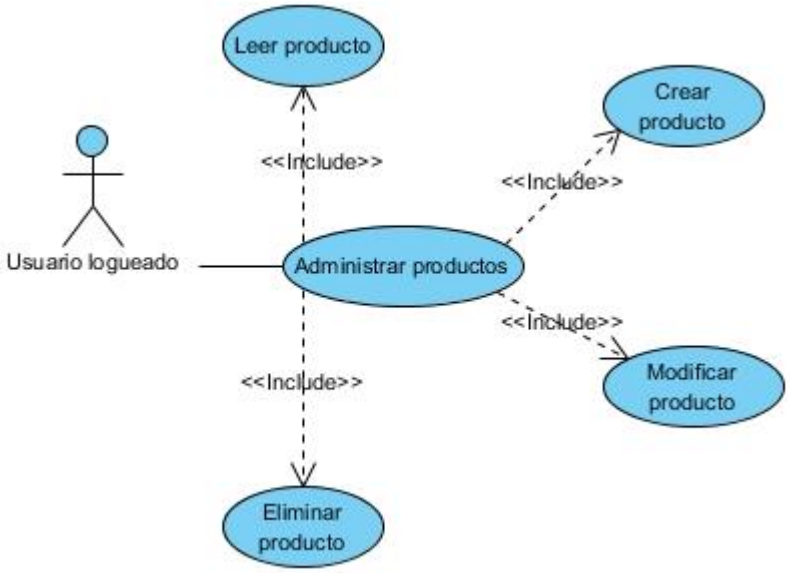


Figura 33: Diagrama de casos de uso de crear, leer, modificar y eliminar productos.

Caso de uso:	CRUD Productos.	**CU06**
Descripción. El usuario debe poder manipular productos, pudiendo crear, leer, modificar y eliminar un producto.		
Actores. Usuario.		
Precondiciones. El usuario debe de estar logueado.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar productos”.		

1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo producto. 1.2. El usuario introduce los datos del nuevo producto, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona administrar producto. 2.1. El usuario selecciona un producto y pulsa “Editar”. 2.2. El usuario modifica los datos del producto y confirma la acción. 3. El usuario accede a los ajustes del sistema y selecciona “Administrar productos”. 3.1. El usuario selecciona un producto y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. El producto es creado en la base de datos. 2. El producto es actualizado en la base de datos. 3. El producto es eliminado de la base de datos.

CU7. Crear, leer, modificar y eliminar ingredientes. (Figura 34)

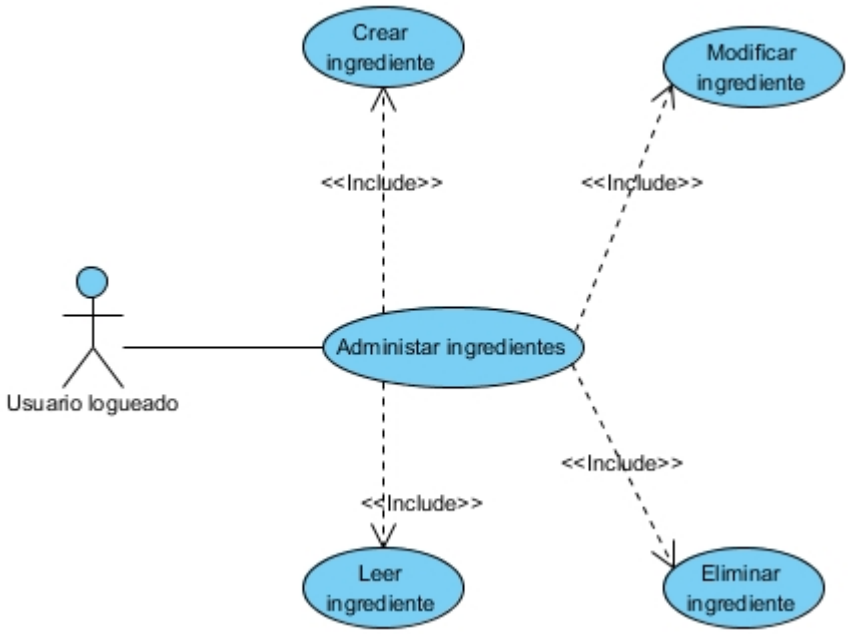


Figura 34: Diagrama de casos de uso de crear, leer, modificar y eliminar ingredientes.

Caso de uso:	CRUD Ingredientes.	**CU07**

Descripción.
El usuario debe poder manipular ingredientes, pudiendo crear, leer, modificar y eliminar un ingrediente.

Actores.
Usuario.

Precondiciones.
El usuario debe de estar logueado.

Flujo natural.

1. El usuario accede a los ajustes del sistema y selecciona “Administrar ingredientes”.
   - 1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo ingrediente.
   - 1.2. El usuario introduce los datos del nuevo ingrediente, y confirma los datos.
2. El usuario accede a los ajustes del sistema y selecciona administrar ingrediente.
   - 2.1. El usuario selecciona un ingrediente y pulsa “Editar”.
   - 2.2. El usuario modifica los datos del ingrediente y confirma la acción.
3. El usuario accede a los ajustes del sistema y selecciona “Administrar ingredientes”.
   - 3.1. El usuario selecciona un ingrediente y pulsa sobre “Eliminar”.
   - 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.
4. El usuario cliente mediante el buscador de locales selecciona un local.
   - 4.1. El usuario accede al menú lateral “Stock”, donde puede ver los productos del local.

Flujo alternativo.
1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco.
1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.

Post-condiciones.

1. El ingrediente es creado en la base de datos.
2. El ingrediente es actualizado en la base de datos.
3. El ingrediente es eliminado de la base de datos.

CU8. Crear, leer, modificar y eliminar categorías. (Figura 35)

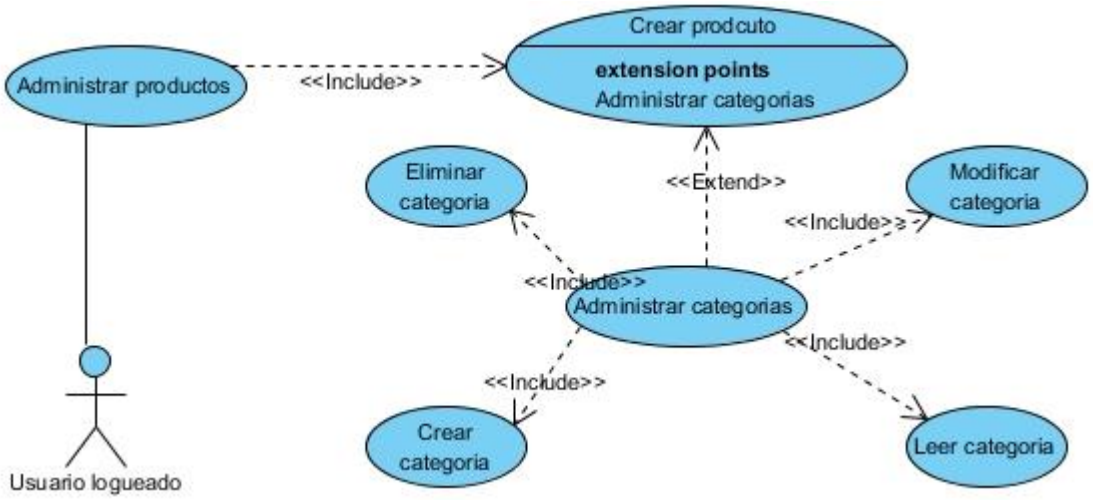


Figura 35: Diagrama de casos de uso de crear, leer, modificar y eliminar categorías.

Caso de uso:	CRUD Categorías.	**CU08**
Descripción. El empleado debe poder manipular categorías, pudiendo crear, leer, modificar y eliminar una categoría.		
Actores. Usuario empleado. Usuario administrador.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar productos”. 1.1. El usuario pulsa sobre el botón de “+” para insertar un nuevo producto. 1.2. El usuario pulsa sobre “+” en la sección de “Categoría”. 1.3. Pulsa sobre el botón de “+” para insertar una nueva categoría. 1.4. El usuario introduce los datos de la nueva categoría, y confirma los datos. 2. De la misma forma que para crear una categoría, el usuario accede a administrar categorías. 2.1. El usuario selecciona una categoría y pulsa “Editar”. 2.2. El usuario modifica los datos de la categoría y confirma la acción. 3. De la misma forma que para crear una categoría, el usuario accede a administrar categorías. 3.1. El usuario selecciona una categoría y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.		

Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.
Post-condiciones. 1. La categoría es creada en la base de datos. 2. La categoría es actualizada en la base de datos. 3. La categoría es eliminada de la base de datos.

CU9. Crear, leer, modificar y eliminar sugerencias. (Figura 36)

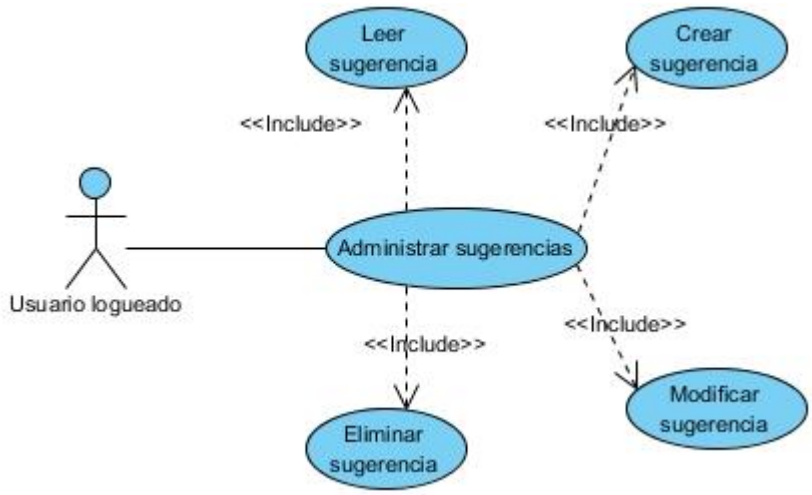


Figura 36: Diagrama de casos de uso de crear, leer, modificar y eliminar sugerencias.

Caso de uso:	CRUD Sugerencias.	**CU09**
Descripción. El usuario debe poder manipular sugerencias, pudiendo crear, leer, modificar y eliminar una sugerencia.		
Actores. Usuario.		
Precondiciones. El usuario debe de estar logueado como empleado o como empresa.		
Flujo natural. 1. El usuario accede a los ajustes del sistema y selecciona “Administrar sugerencias”. 1.1. El usuario pulsa sobre el botón de “+” para insertar una nueva sugerencia. 1.2. El usuario introduce los datos de la nueva sugerencia, y confirma los datos. 2. El usuario accede a los ajustes del sistema y selecciona “Administrar sugerencias. 2.1. El usuario selecciona una sugerencia y pulsa “Editar”. 2.2. El usuario modifica los datos de la sugerencia y confirma la acción. 2.3. El usuario puede modificar si la sugerencia es visible a los clientes o no. 3. El usuario accede a los ajustes del sistema y selecciona “administrar categorías”. 3.1. El administrador selecciona una sugerencia y pulsa sobre “Eliminar”. 3.2. El sistema muestra un mensaje de confirmación y el usuario confirma que desea realizar la acción.		
Flujo alternativo. 1.2. La aplicación mostrará un mensaje de error si el usuario ha dejado campos en blanco. 1.2. La aplicación mostrará un mensaje de error si el usuario introduce un identificador que ya está introducido en la base de datos.		
Post-condiciones. 1. La sugerencia es creada en la base de datos.		

2. La sugerencia es actualizada en la base de datos.
   2.1. La visualización de la sugerencia es actualizada.
3. La sugerencia es eliminada de la base de datos.

5.4. Diagramas de secuencias de la aplicación móvil.

DI01. Registrar usuario. (Figura 37)

Flujo del diagrama.

1. Un usuario no registrado en el sistema pulsa sobre registrarse.
2. LoginActivity pregunta al usuario si pretende registrarse como cliente o compo empresa.
3. El usuario elige el modo de registro en el sistema.
4. LoginActivity llama a RegistroCliente y muestra los dos en pantalla.
5. El usuario rellena los datos y confirma el registro.
6. RegistroCliente manda los datos DataBaseManager y esta clase comprueba que los datos estén correctos.
7. Una vez comprobado, crea el usuario en la api y devuelve un mensaje al usuario confirmando que el usuario ha sido creado.

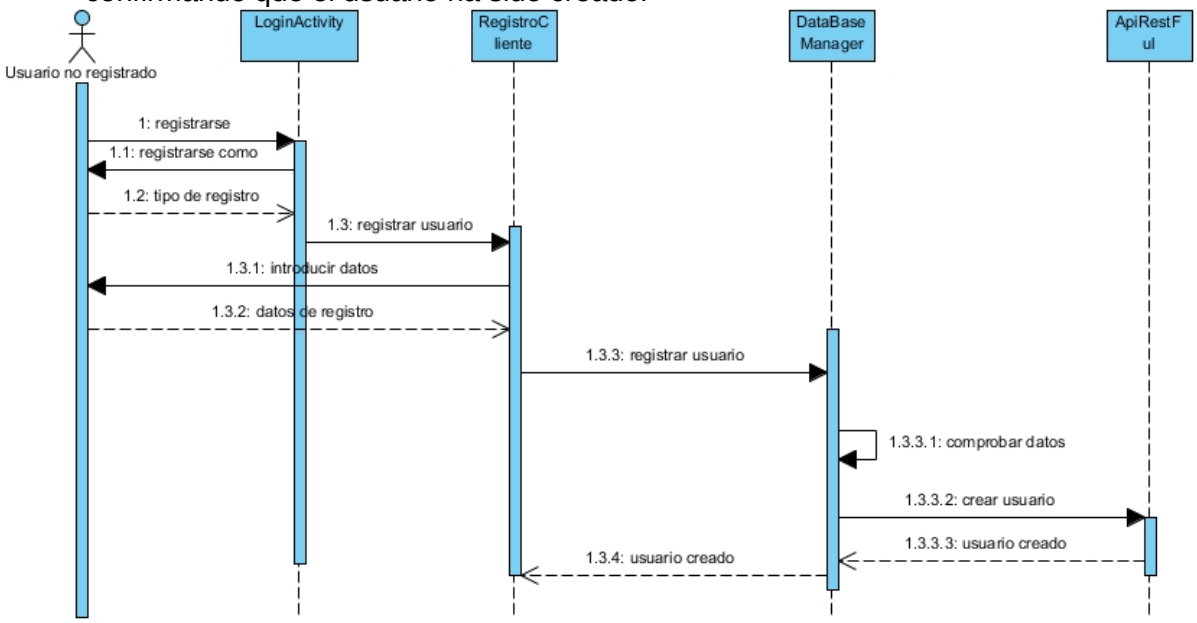


Figura 37: Diagrama de secuencias de registrar usuario.

DI02. Login y Logout. (Figura 38)

Flujo del diagrama.

1. Un usuario rellena los datos de la pantalla principal e inicia sesión.
2. LoginActivity manda un mensaje a DataBaseManager y esta clase hace una petición a la Api para obtener los datos del sistema.
3. La api devuelve los usuarios del sistema y DataBaseManager comprueba si los datos introducidos coinciden con los datos del sistema.
4. Una vez que ha obtenido la información, se envía un mensaje al usuario confirmando el inicio de sesión.

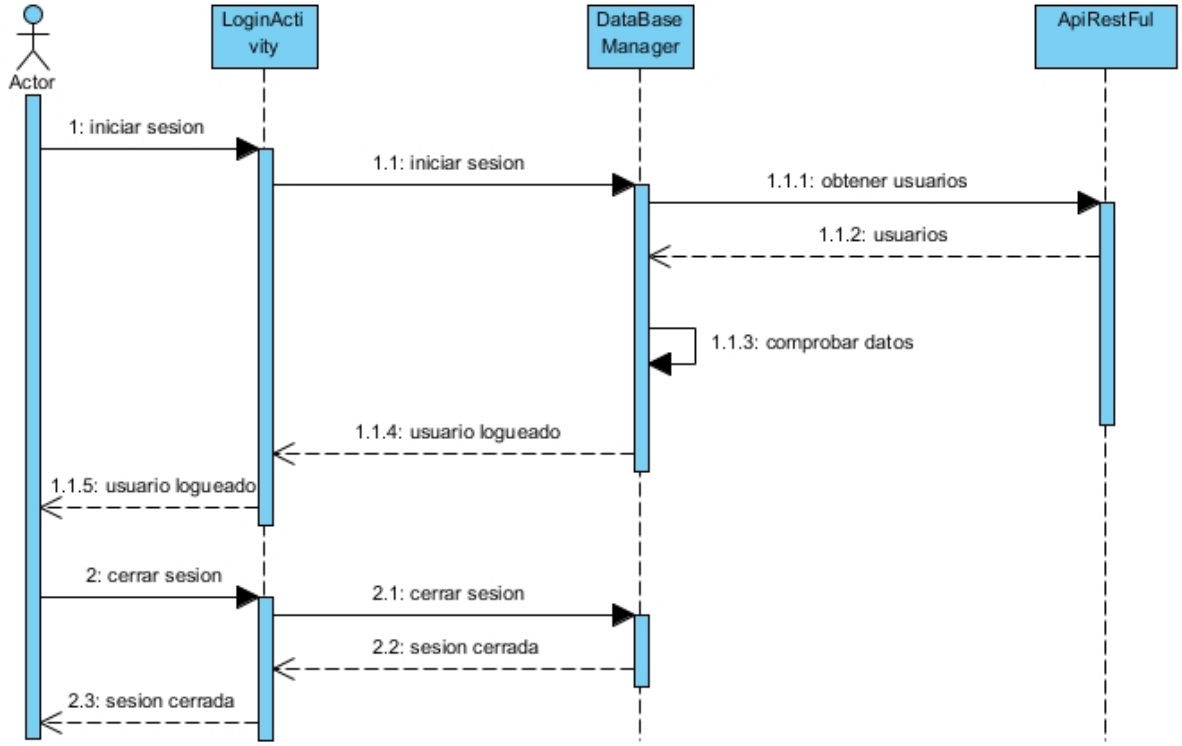


Figura 38: Diagrama de secuencias de login y logout.

DI03. Recuperar contraseña. (Figura 39)

Flujo del diagrama.

1. Un usuario desde la pantalla inicial pulsa sobre “Recuperar contraseña”.
2. LoginActivity muestra una ventana emergente en la que se solicita el nombre de usuario o email
3. LoginActivity realiza una búsqueda en la ApiRestFul para verificar que los datos existen
4. Una vez se ha obtenido la información, LoginActivity genera una nueva contraseña y se envía un mensaje al usuario con su nueva credencial.

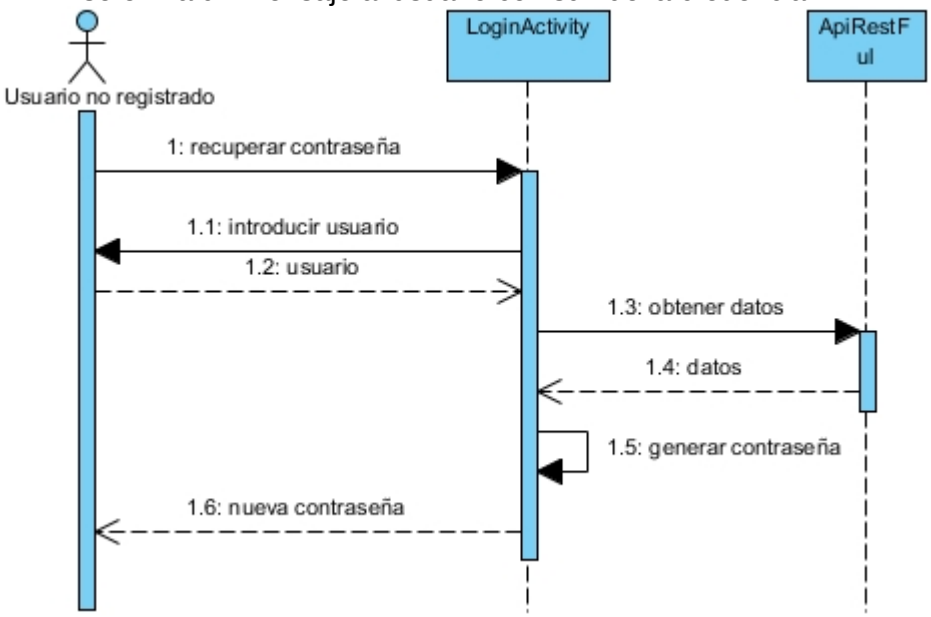


Figura 39: Diagrama de secuencias de recuperar contraseña..

DI04. Modificar datos. (Figura 40)

Flujo del diagrama.

1. Un usuario desde SettingActivity, pulsa sobre “Administrar perfil”.
2. El sistema le lleva a AdministrarPerfilActivity y muestra al usuario sus datos.
3. El usuario modifica sus datos y los envía a AdministrarPerfilActivity.
4. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
5. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando la modificación.

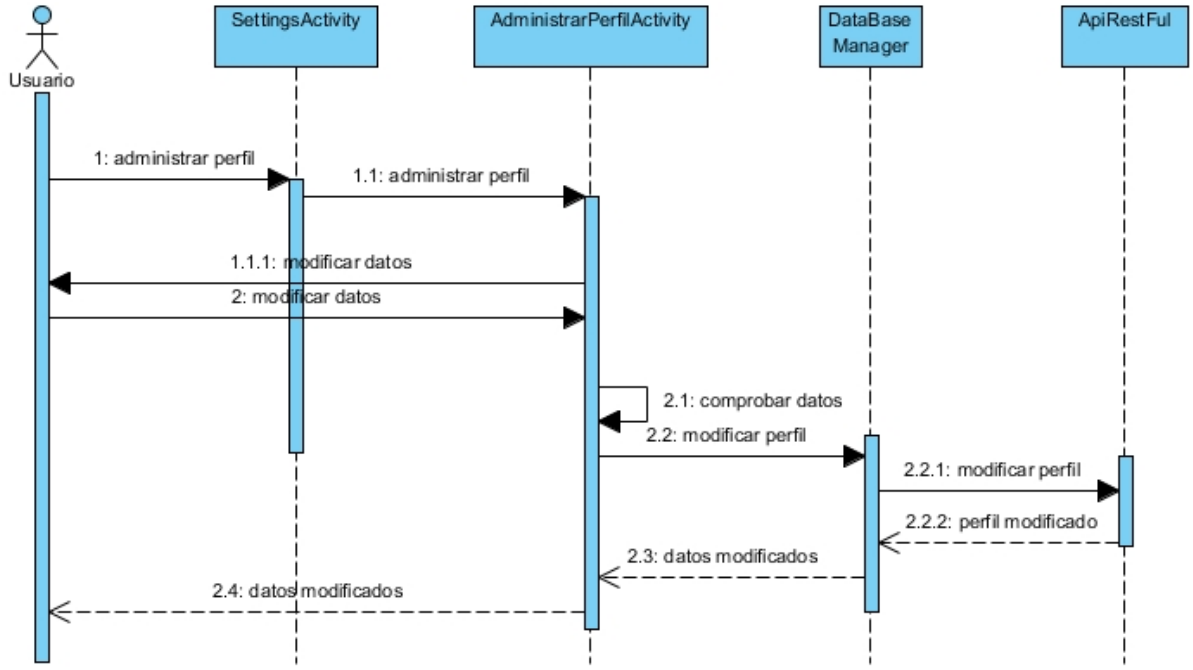


Figura 40: Diagrama de secuencias de modificar datos.

DI05. Crear empleado. (Figura 41)

Flujo del diagrama.

1. Un administrador de un local desde SettingActivity, pulsa sobre “Administrar empleados”.
2. El sistema le lleva a AdministrarEmpleadosActivity y el administrador selecciona crear un empleado.
3. La clase muestra una pantalla en la que introducir los datos del empleado y el administrador los rellena y los envía.
4. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
5. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando la creación.

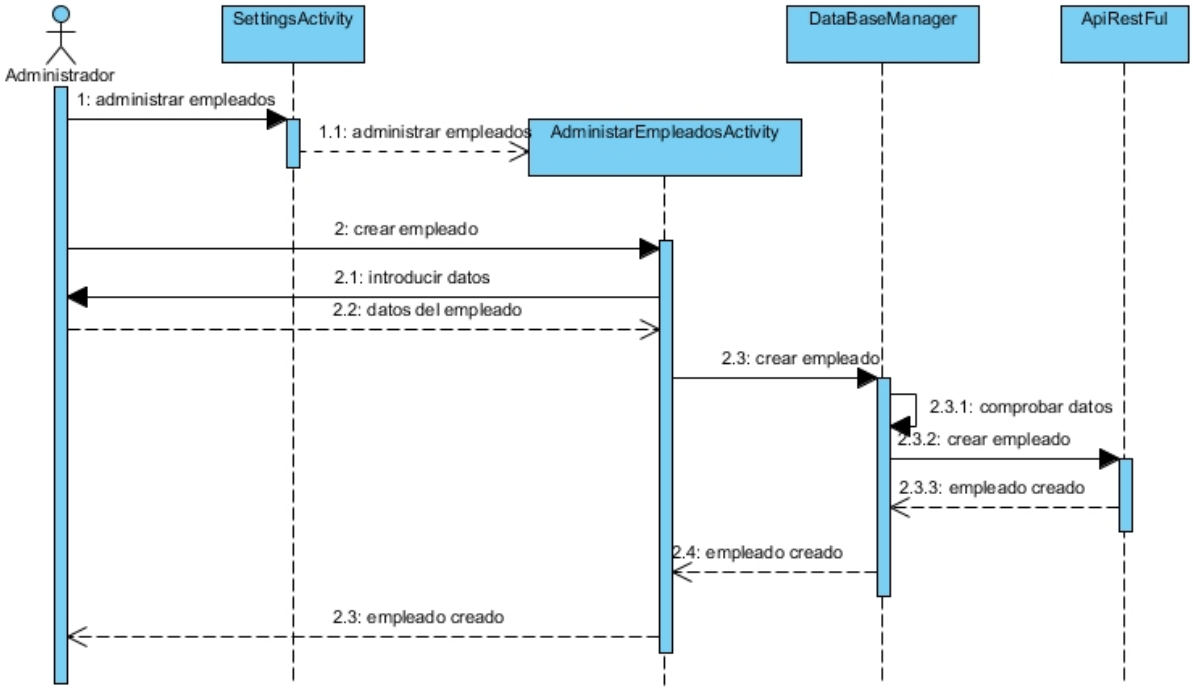


Figura 41: Diagrama de secuencias de crear empleado.

DI06. Eliminar empleado. (Figura 42)

Flujo del diagrama.

1. El administrador de un local desde SettingActivity, pulsa sobre “Administrar empleado”.
2. El administrador desde selecciona un empleado y la clase le muestra sus datos.
3. El empleado pulsa sobre eliminar.
4. El Sistema le pregunta si quiere realizar la acción y el empleado confirma.
5. La clase le manda los datos a DataBaseManager y desde aquí se elimina el empleado.
6. El sistema devuelve un mensaje confirmando que el empleado ha sido eliminado.

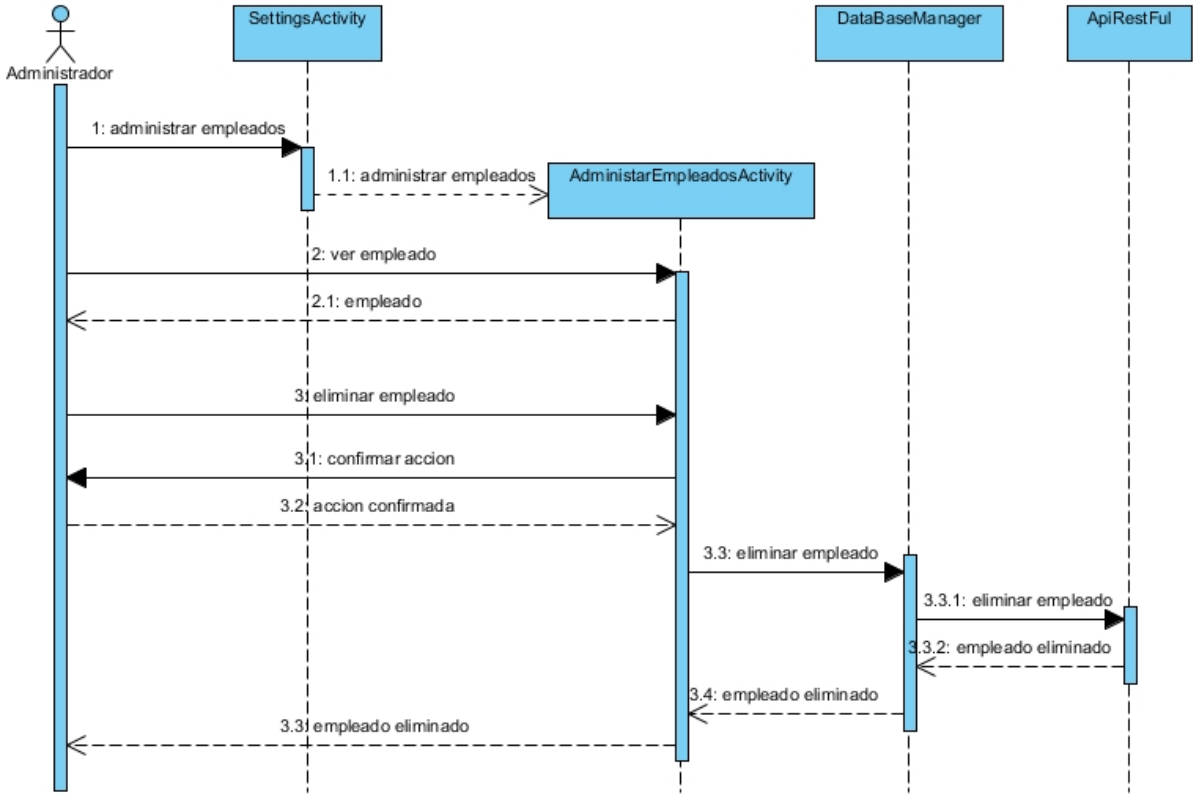


Figura 42: Diagrama de secuencias de eliminar empleado.

DI07. Asignar ubicación. (Figura 43)

Flujo del diagrama.

1. Un administrador desde SettingActivity, pulsa sobre “Mapa”.
2. El sistema le muestra un mapa de la zona en la que se encuentra.
3. El administrador selecciona en el mapa la ubicación del restaurante.
4. MapaActivity llama a DataBaseManager que se encarga de modificar la ubicación en la api.

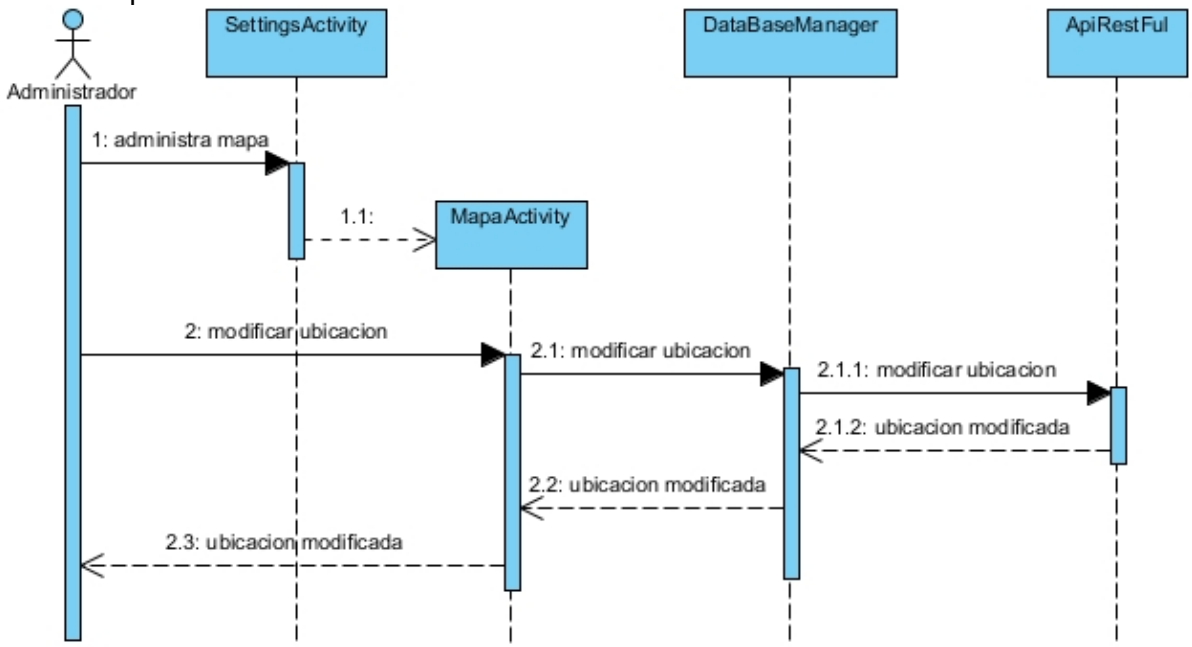


Figura 43: Diagrama de secuencias de asignar ubicación.

DI08. Crear, leer, modificar y eliminar mesas. (Figura 44)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar mesas”.
2. SettingsActivity abre una nueva instancia de AdministrarMesasActivity donde se encuentran todas las acciones a realizar.
3. Para crear una mesa, el empleado selecciona crear una mesa mediante el botón.
   - 3.1. La clase muestra una pantalla en la que introducir los datos de la mesa y el empleado los rellena y los envía.
   - 3.2. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
   - 3.3. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando que la creación se ha llevado a cabo correctamente
4. Para modificar una mesa, el empleado pulsa sobre una de las mesas de la lista y el sistema le muestra sus datos.
   - 4.1. A continuación, el empleado modifica los datos y los envía a la clase.
   - 4.2. La clase comprueba que los datos están bien introducidos y manda un mensaje para modificar la mesa a DataBaseManager.
   - 4.3. DataBaseManager modifica los datos en la api y devuelve al empleado un mensaje confirmando que la modificación se ha llevado a cabo correctamente.

5. Para eliminar una mesa, el empleado pulsa sobre una de las mesas de la lista y selecciona el botón “Eliminar”.
    5.1. AdministrarMesasActivity pide al empleado que confirme que desea eliminar la mesa.
    5.2. Cuando el empleado ha confirmado que desea realizar la acción, la clase manda un mensaje a DataBaseManager para que elimine la mesa.
    5.3. DataBaseManager elimina la mesa de la api y devuelve un mensaje confirmando que la eliminación se ha llevado a cabo correctamente.

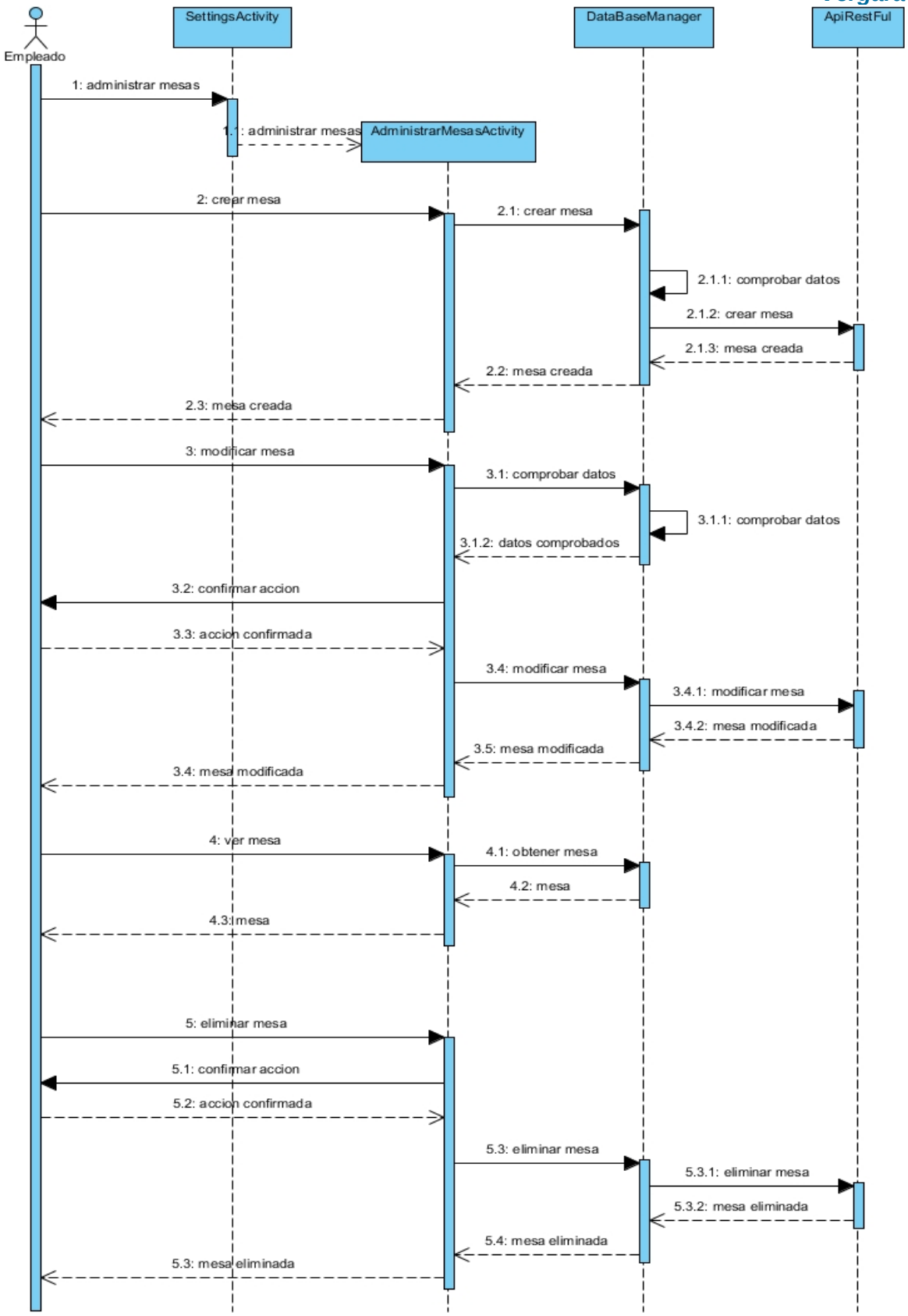


Figura 44: Diagrama de secuencias de crear, leer, modificar y eliminar mesas.

DI09. Crear, leer, modificar y eliminar espacios del local. (Figura 45)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar espacios”.
2. SettingsActivity abre una nueva instancia de AdministrarEspaciosActivity donde se encuentran todas las acciones a realizar.
3. Para crear un espacio, el empleado selecciona crear un espacio mediante el botón.
   - 3.1. La clase muestra una pantalla en la que introducir los datos del espacio y el empleado los rellena y los envía.
   - 3.2. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
   - 3.3. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando que la creación se ha llevado a cabo correctamente.
4. Para modificar un espacio, el empleado pulsa sobre uno de los espacios de la lista y el sistema le muestra sus datos.
   - 4.1. A continuación, el empleado modifica los datos y los envía a la clase.
   - 4.2. La clase comprueba que los datos están bien introducidos y manda un mensaje para modificar el espacio a DataBaseManager.
   - 4.3. DataBaseManager modifica los datos en la api y devuelve al empleado un mensaje confirmando que la modificación se ha llevado a cabo correctamente.
5. Para eliminar un espacio, el empleado pulsa sobre uno de los espacios de la lista y selecciona el botón “Eliminar”.
   - 5.1. AdministrarEspaciosActivity pide al empleado que confirme que desea eliminar el espacio.
   - 5.2. Cuando el empleado ha confirmado que desea realizar la acción, la clase manda un mensaje a DataBaseManager para que elimine el espacio.
   - 5.3. DataBaseManager elimina el espacio de la api y devuelve un mensaje confirmando que la eliminación se ha llevado a cabo correctamente.

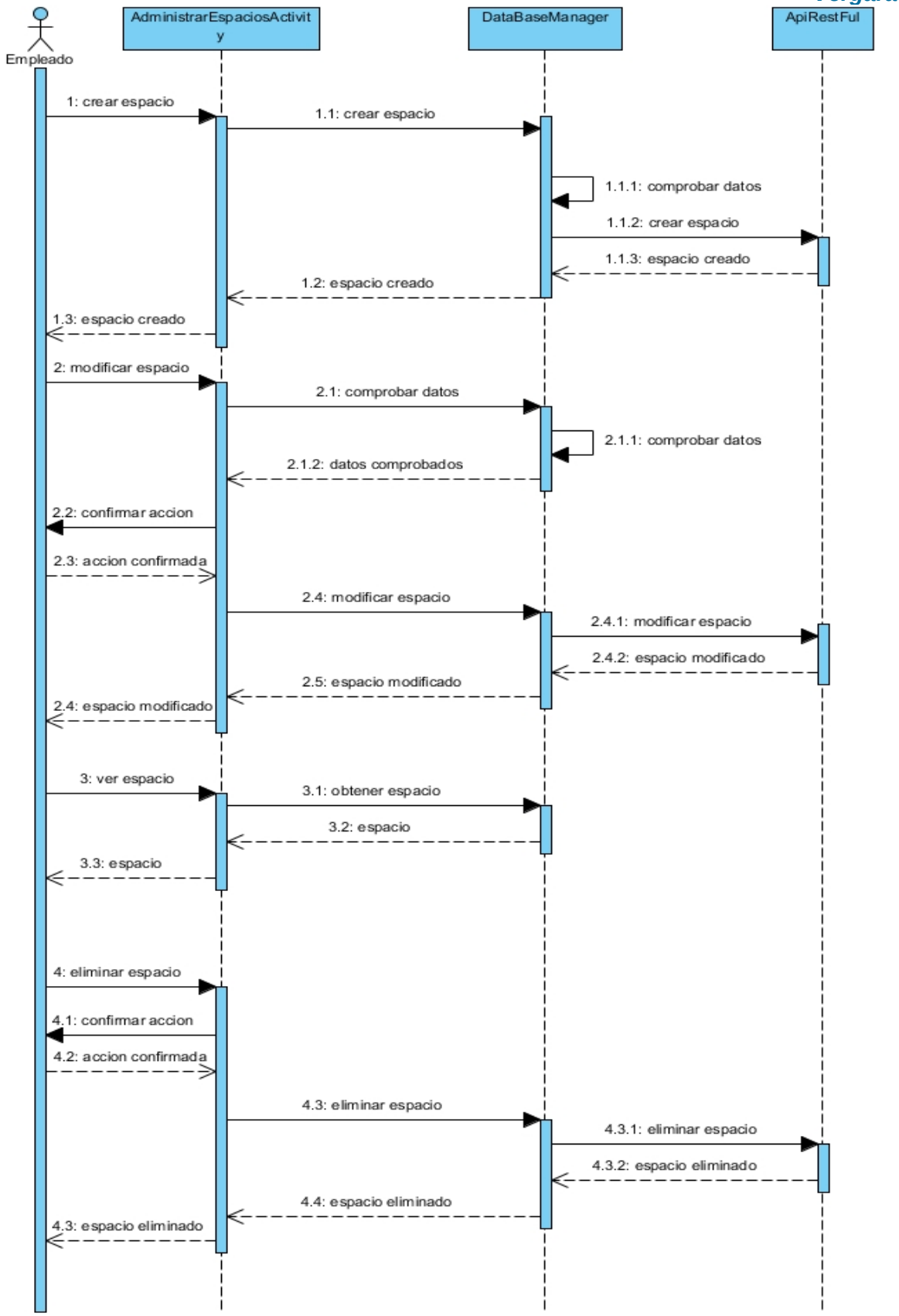


Figura 45: Diagrama de secuencias de crear, leer, modificar y eliminar espacios.

DI10. Crear, leer, modificar y eliminar productos. (Figura 46)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar productos”.
2. SettingsActivity abre una nueva instancia de StockActivity donde se encuentran todas las acciones a realizar.
3. Para crear un producto, el empleado selecciona crear un producto mediante el botón.
   - 3.1. La clase muestra una pantalla en la que introducir los datos del producto y el empleado los rellena y los envía.
   - 3.2. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
   - 3.3. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando que la creación se ha llevado a cabo correctamente.
4. Para modificar un producto, el empleado pulsa sobre uno de los productos de la lista y el sistema le muestra sus datos.
   - 4.1. A continuación, el empleado modifica los datos y los envía a la clase.
   - 4.2. La clase comprueba que los datos están bien introducidos y manda un mensaje para modificar el producto a DataBaseManager.
   - 4.3. DataBaseManager modifica los datos en la api y devuelve al empleado un mensaje confirmando que la modificación se ha llevado a cabo correctamente.
5. Para eliminar un producto, el empleado pulsa sobre uno de los productos de la lista y selecciona el botón “Eliminar”.
   - 5.1. StockActivity pide al empleado que confirme que desea eliminar el producto.
   - 5.2. Cuando el empleado ha confirmado que desea realizar la acción, la clase manda un mensaje a DataBaseManager para que elimine el producto.
   - 5.3. DataBaseManager elimina el producto de la api y devuelve un mensaje confirmando que la eliminación se ha llevado a cabo correctamente.

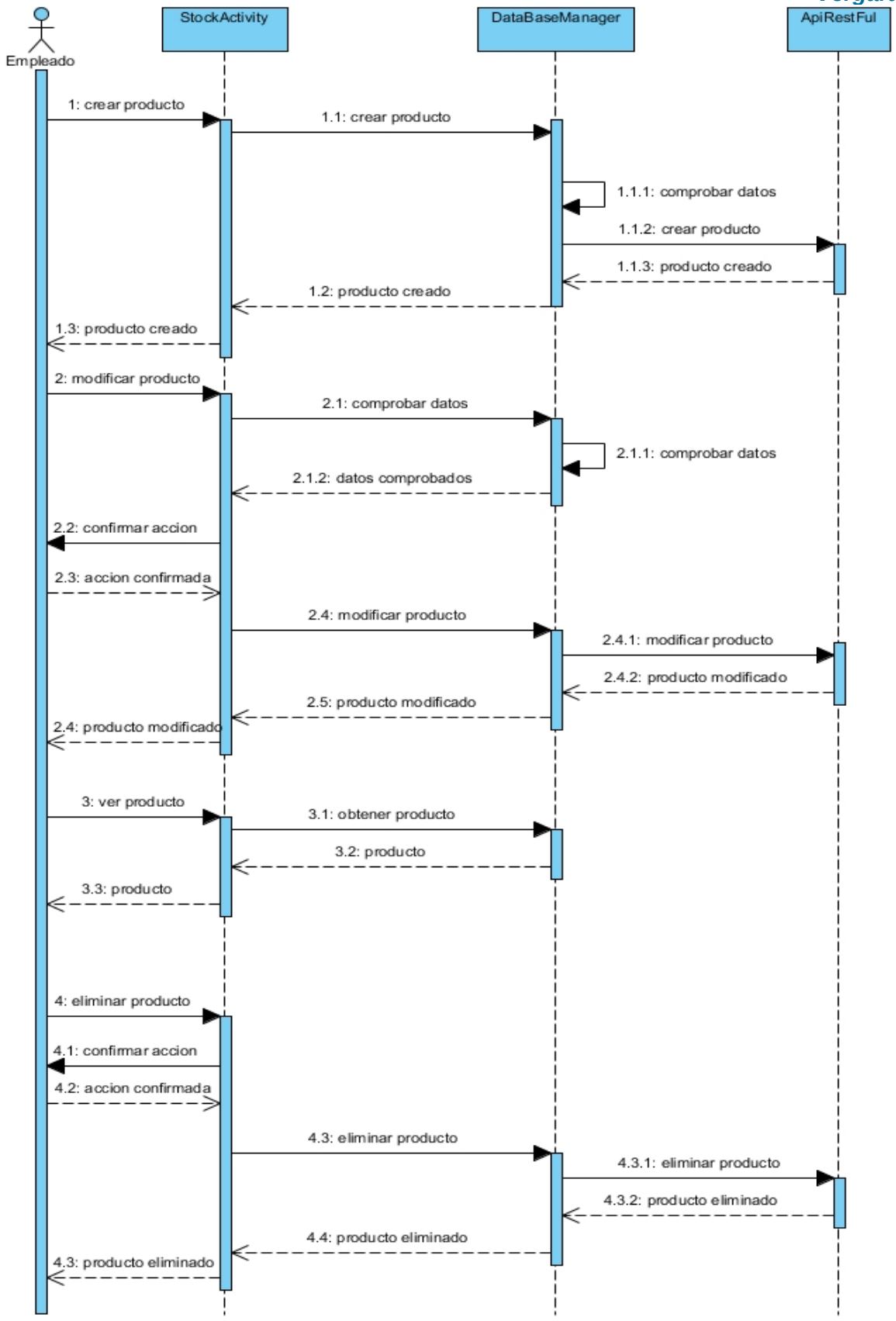


Figura 46: Diagrama de secuencias de crear, leer, modificar y eliminar productos.

DI11. Crear, leer, modificar y eliminar ingredientes. (Figura 47)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar ingredientes”.
2. SettingsActivity abre una nueva instancia de StockActivity donde se encuentran todas las acciones a realizar.
3. Para crear un ingrediente, el empleado selecciona crear un ingrediente mediante el botón.
   - 3.1. La clase muestra una pantalla en la que introducir los datos del ingrediente y el empleado los rellena y los envía.
   - 3.2. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
   - 3.3. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando que la creación se ha llevado a cabo correctamente.
4. Para modificar un ingrediente, el empleado pulsa sobre uno de los ingredientes de la lista y el sistema le muestra sus datos.
   - 4.1. A continuación, el empleado modifica los datos y los envía a la clase.
   - 4.2. La clase comprueba que los datos están bien introducidos y manda un mensaje para modificar el producto a DataBaseManager.
   - 4.3. DataBaseManager modifica los datos en la api y devuelve al empleado un mensaje confirmando que la modificación se ha llevado a cabo correctamente.
5. Para eliminar un ingrediente, el empleado pulsa sobre uno de los ingredientes de la lista y selecciona el botón “Eliminar”.
   - 5.1. StockActivity pide al empleado que confirme que desea eliminar el ingrediente.
   - 5.2. Cuando el empleado ha confirmado que desea realizar la acción, la clase manda un mensaje a DataBaseManager para que elimine el producto.
   - 5.3. DataBaseManager elimina el producto de la api y devuelve un mensaje confirmando que la eliminación se ha llevado a cabo correctamente.

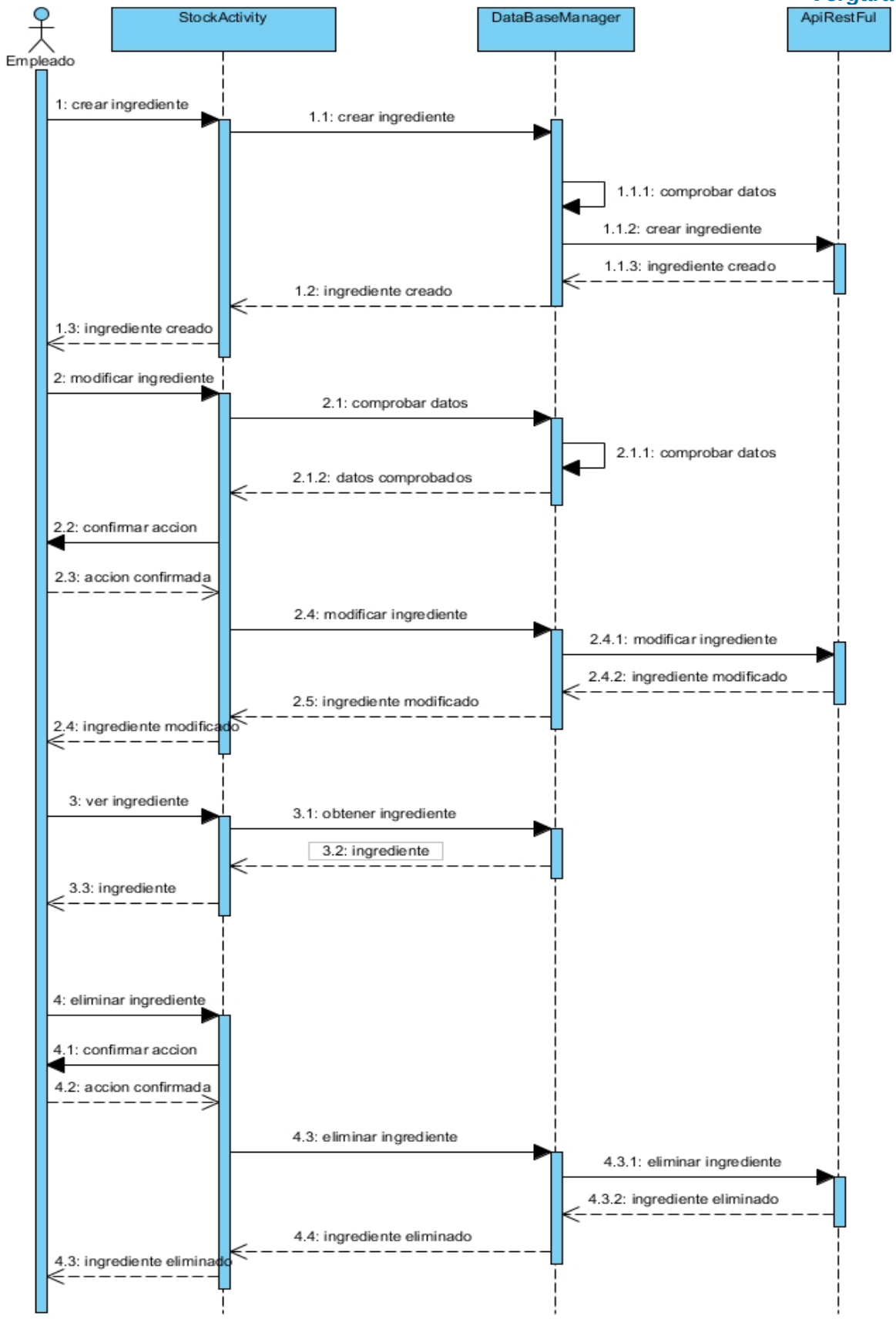


Figura 47: Diagrama de secuencias de crear, leer, modificar y eliminar ingredientes.

DI12. Crear, leer, modificar y eliminar categorías. (Figura 48)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar categorias”.
2. SettingsActivity abre una nueva instancia de AdministrarCategoriasActivity donde se encuentran todas las acciones a realizar.
3. Para crear una categoría, el empleado selecciona crear una categoría mediante el botón.
    3.1. La clase muestra una pantalla en la que introducir los datos de la categoría y el empleado los rellena y los envía.
    3.2. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
    3.3. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando que la creación se ha llevado a cabo correctamente
4. Para modificar una categoría, el empleado pulsa sobre una de las categorías de la lista y el sistema le muestra sus datos.
    4.1. A continuación, el empleado modifica los datos y los envía a la clase.
    4.2. La clase comprueba que los datos están bien introducidos y manda un mensaje para modificar la mesa a DataBaseManager.
    4.3. DataBaseManager modifica los datos en la api y devuelve al empleado un mensaje confirmando que la modificación se ha llevado a cabo correctamente.
5. Para eliminar una categoría, el empleado pulsa sobre una de las categorías de la lista y selecciona el botón “Eliminar”.
    5.1. AdministrarCategoriasActivity pide al empleado que confirme que desea eliminar la categoría.
    5.2. Cuando el empleado ha confirmado que desea realizar la acción, la clase manda un mensaje a DataBaseManager para que elimine la categoría.
    5.3. DataBaseManager elimina la mesa de la api y devuelve un mensaje confirmando que la eliminación se ha llevado a cabo correctamente.

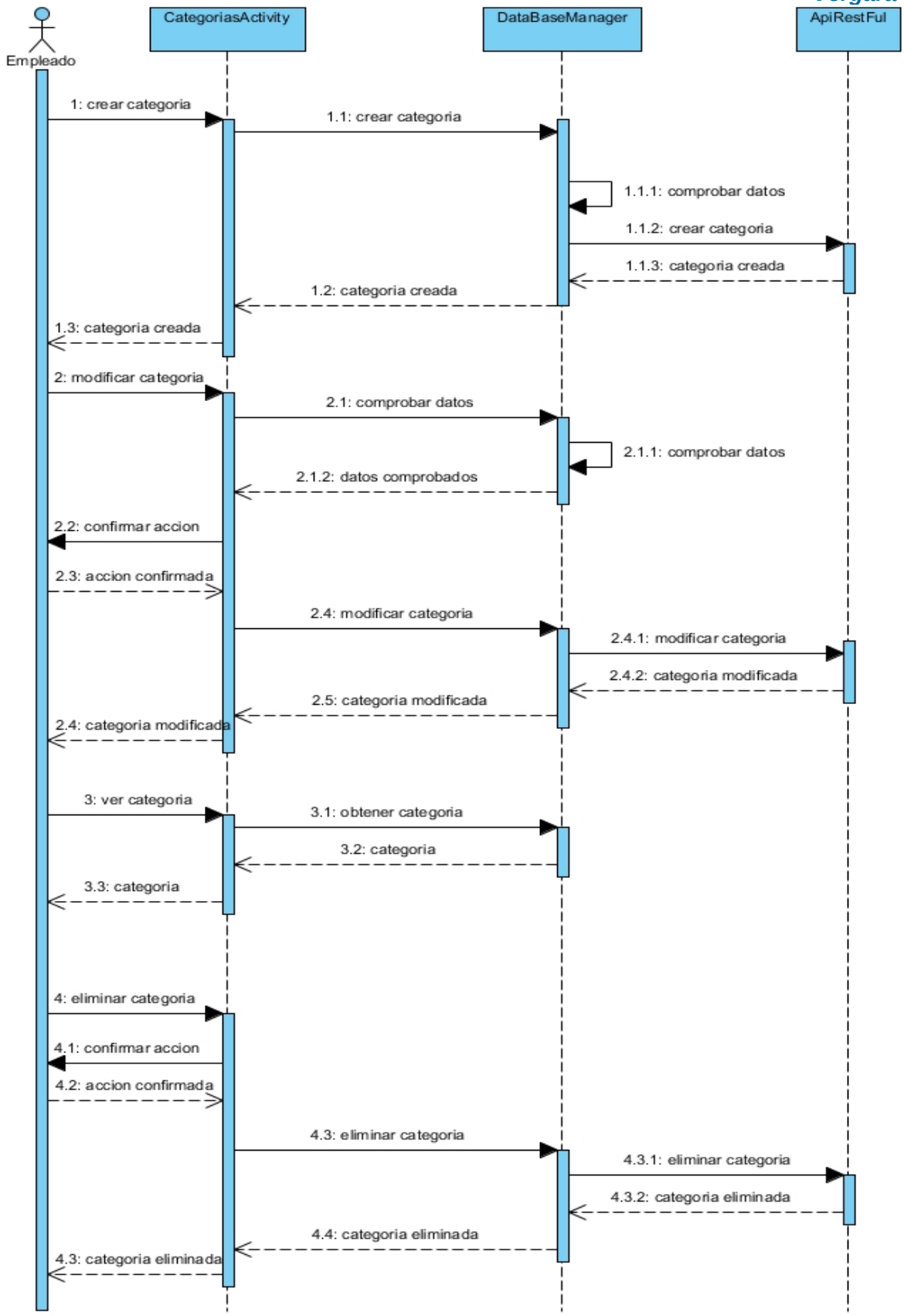


Figura 48: Diagrama de secuencias de crear, leer, modificar y eliminar categorías.

DI13. Crear, leer, modificar y eliminar sugerencias. (Figura 49)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar sugerencias”.
2. SettingsActivity abre una nueva instancia de SugerenciasActivity donde se encuentran todas las acciones a realizar.
3. Para crear una sugerencia, el empleado selecciona crear una sugerencia mediante el botón.
   - 3.1. La clase muestra una pantalla en la que introducir los datos de la sugerencia y el empleado los rellena y los envía.
   - 3.2. La clase comprueba que los datos estén bien introducidos y después los manda a DataBaseManager.
   - 3.3. DataBaseManager manda los datos a la Api y la api devuelve un mensaje confirmando que la creación se ha llevado a cabo correctamente
4. Para modificar una sugerencia, el empleado pulsa sobre una de las sugerencias de la lista y el sistema le muestra sus datos.
   - 4.1. A continuación, el empleado modifica los datos y los envía a la clase.
   - 4.2. La clase comprueba que los datos están bien introducidos y manda un mensaje para modificar la mesa a DataBaseManager.
   - 4.3. DataBaseManager modifica los datos en la api y devuelve al empleado un mensaje confirmando que la modificación se ha llevado a cabo correctamente.
5. Para eliminar una sugerencia, el empleado pulsa sobre una de las sugerencias de la lista y selecciona el botón “Eliminar”.
   - 5.1. SugerenciasActivity pide al empleado que confirme que desea eliminar la sugerencia.
   - 5.2. Cuando el empleado ha confirmado que desea realizar la acción, la clase manda un mensaje a DataBaseManager para que elimine la sugerencia.
   - 5.3. DataBaseManager elimina la mesa de la api y devuelve un mensaje confirmando que la eliminación se ha llevado a cabo correctamente.

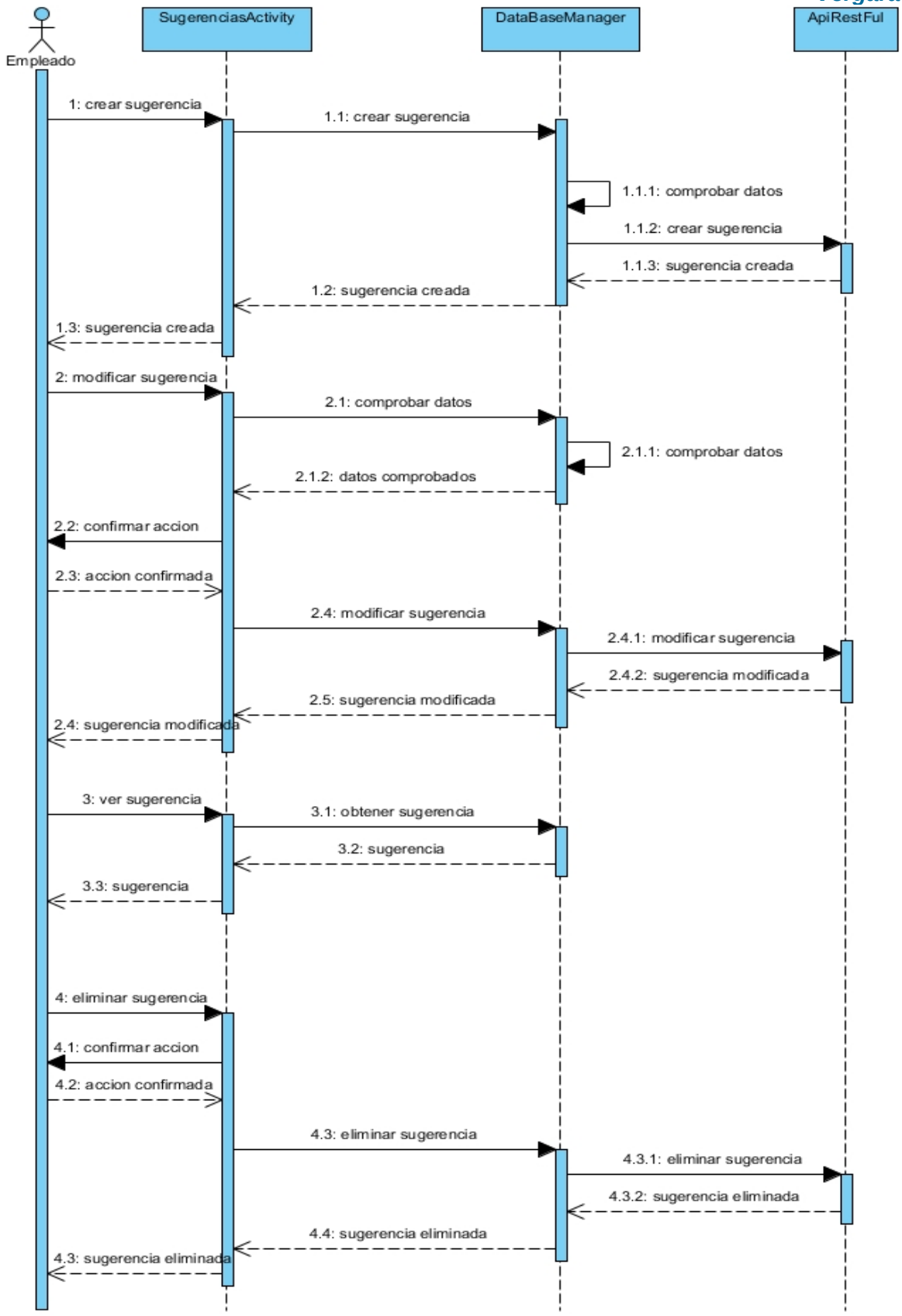


Figura 49: Diagrama de secuencias de crear, leer, modificar y eliminar sugerencias.

DI14. Asignación de mesas. (Figura 50)

Flujo del diagrama.

1. Un empleado desde SettingsActivity, pulsa sobre “Administrar mesas”.
2. Dentro de AdministrarMesasActicity, el empleado pulsa sobre “Asignar mesas” y selecciona las mesas que quiere asignarse.
3. El sistema le muestra un mensaje para confirmar la acción.
4. El empleado confirma la acción y la clase lanza un mensaje a DataBaseManager para que actualice las mesas.
5. DataBaseManager actualiza las mesas en la api y devuelve un mensaje confirmando que la acción se ha realizado correctamente.

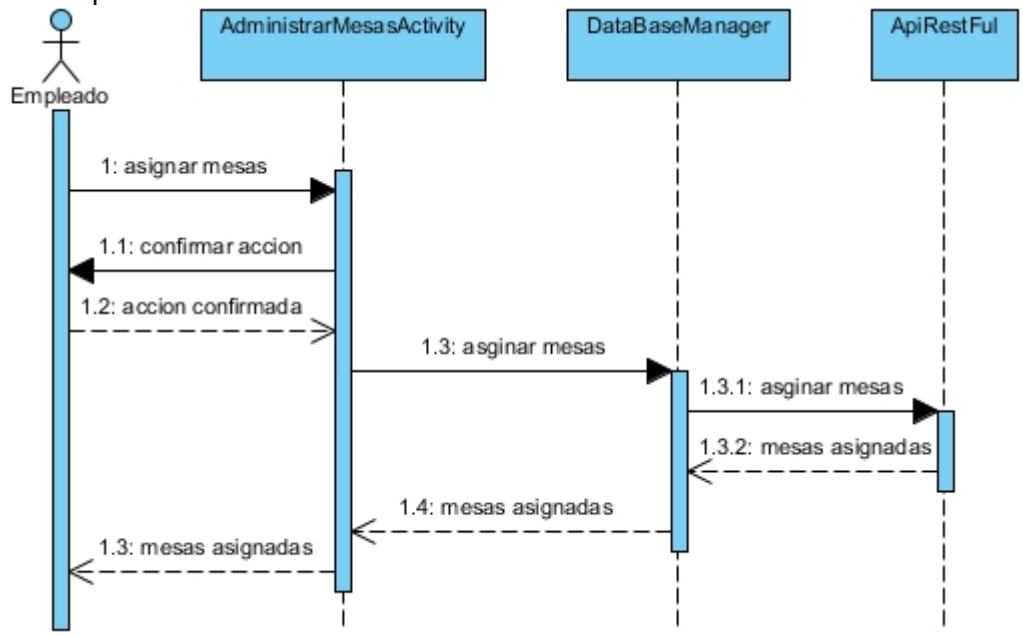


Figura 50: Diagrama de secuencias de asignación de mesas.

DI15. Generar código QR de mesas.(Figura 51)

Flujo del diagrama.

1. Un empleado desde MainActivity, pulsa sobre una mesa libre.
   - 1.1. El sistema le envía un mensaje confirmando si desea generar el código QR de la mesa y el empleado confirma la acción.
   - 1.2. MainActivity llama a la clase PDFCreator y esta genera el PDF.
   - 1.3. El PDF es descargado en el dispositivo móvil y se envía un mensaje confirmando que la acción se ha llevado a cabo correctamente.
2. Un empleado, desde AdministrarMesasActivity, selecciona generar código QR y elige las mesas cuyos códigos QR desea generar.
   - 2.1. El sistema muestra un mensaje para confirmar la acción y el empleado confirma que desea realizar la acción.
   - 2.2. AdministrarMesasActivity lanza un mensaje a la clase PDFCreator y esta clase genera el PDF con el código QR de todas las mesas seleccionadas.
   - 2.3. El sistema descarga el PDF en el dispositivo móvil y devuelve un mensaje confirmando que la acción se ha llevado a cabo correctamente.

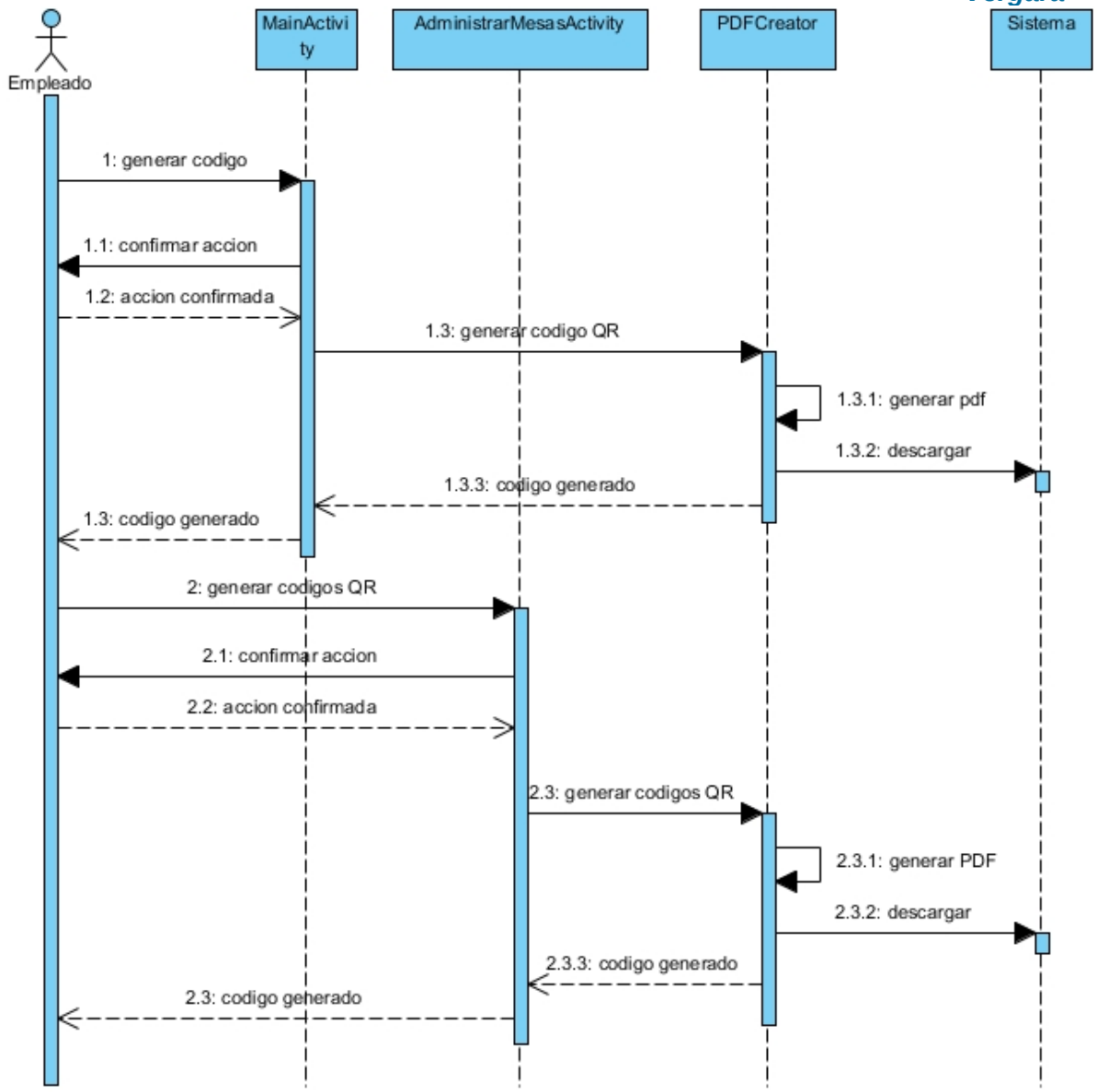


Figura 51: Diagrama de secuencias de generar código QR de mesas..

DI16. Modificar Stock producto. (Figura 52)

Flujo del diagrama.

1. Un empleado desde StockActivity selecciona un producto para ver sus datos y el sistema le muestra los detalles del producto.
2. El empleado selecciona modificar stock del producto.
3. La clase StockActivity lanza un mensaje a DataBaseManager cambiando el stock del producto al estado contrario y DataBaseManager actualiza el stock en la api.

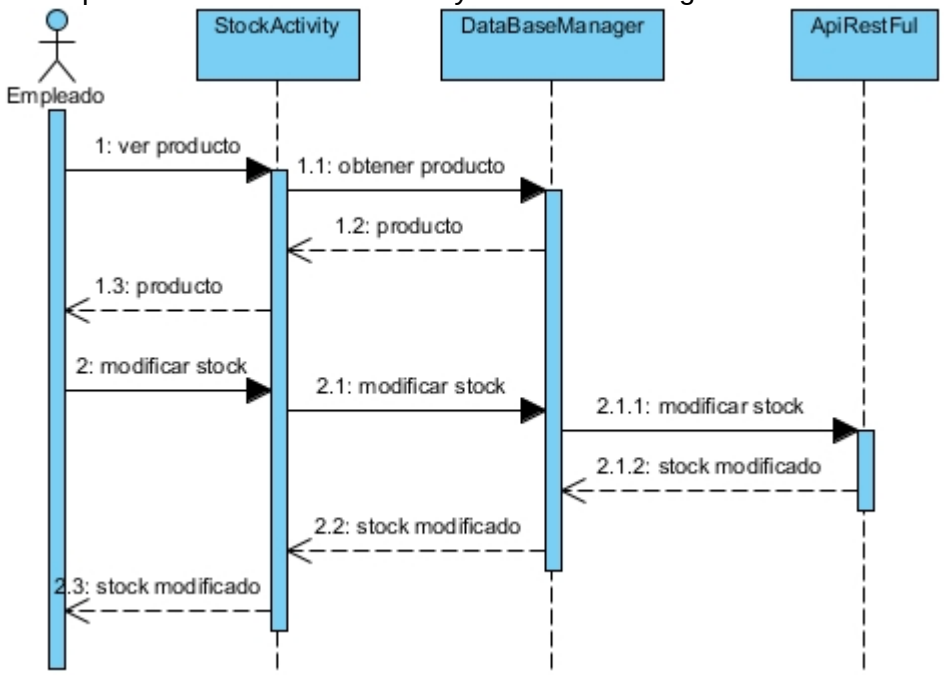


Figura 52: Diagrama de secuencias de modificar stock producto.

DI17. Modificar Stock ingrediente. (Figura 53)

Flujo del diagrama.

4. Un empleado desde StockActivity selecciona un producto para ver sus datos y el sistema le muestra los detalles del ingrediente.
5. El empleado selecciona modificar stock del ingrediente.
6. La clase StockActivity lanza un mensaje a DataBaseManager cambiando el stock del ingrediente al estado contrario y DataBaseManager actualiza el stock en la api.

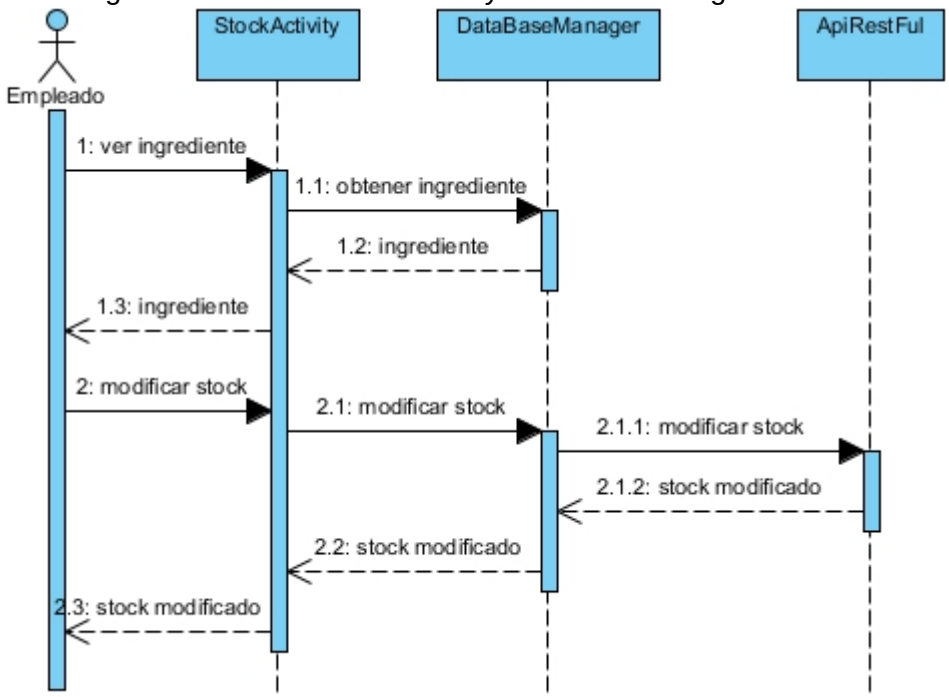


Figura 53: Diagrama de secuencias de modificar stock ingrediente.

DI18. Crear comandas. (Figura 54)

Flujo del diagrama.

1. Un usuario de la aplicación, desde una mesa en la clase MesaActivity, introduce la comanda que desea realizar.
2. Una vez completada, el sistema muestra un mensaje para confirmar que la comanda está correcta y el usuario confirma la acción.
3. La clase MesaActivity lanza un mensaje a DataBaseManager para crear la comanda y esta crea la comanad en la Api.
4. Una vez que la comanda está creada en la api, se devuelve un mensaje al usuario informando de que la acción se ha llevado a cabo correctamente.

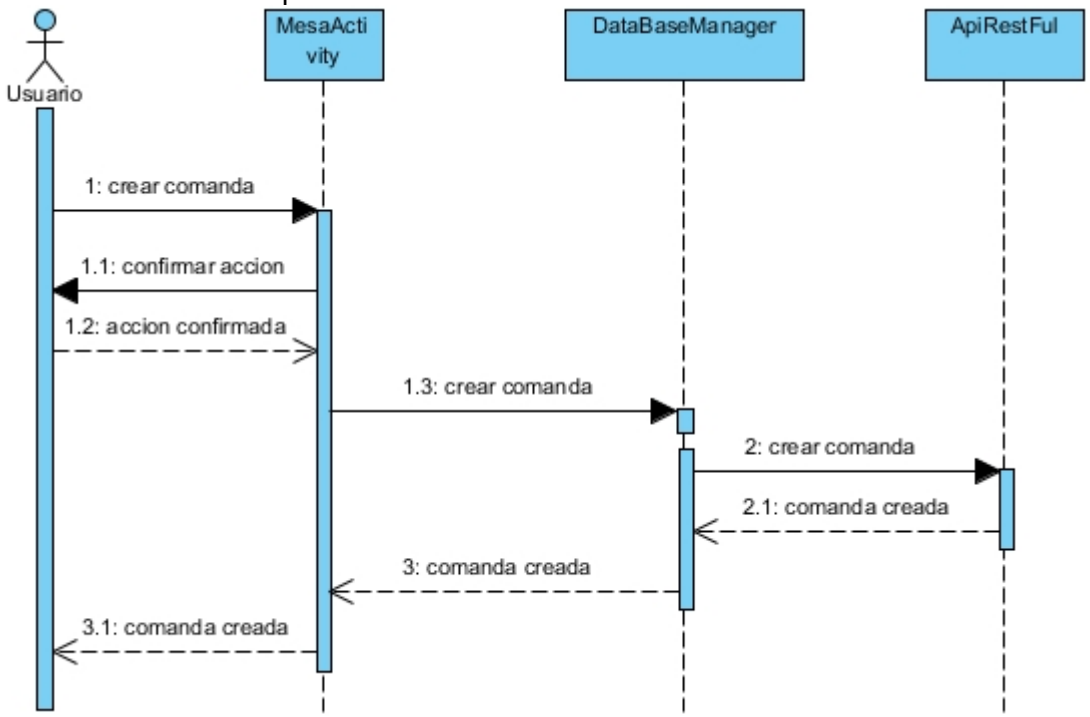


Figura 54: Diagrama de secuencias de crear comandas.

DI19. Consultar comandas como cliente. (Figura 55)

Flujo del diagrama.

1. Un cliente desde la clase MesaActivity pulsa sobre “Estado Comandas” para ver la lista de comandas de su mesa.
2. MesaActivity llama a EstadoComandasActivity que obtiene la lista de comandas de DataBaseManager y las muestra al usuario.
3. El usuario pulsa sobre una de las comandas y EstadoComandasActivity lanza una llamada a DetalleComandaActivity, que obtiene los datos de DataBaseManager y los muestra al cliente.

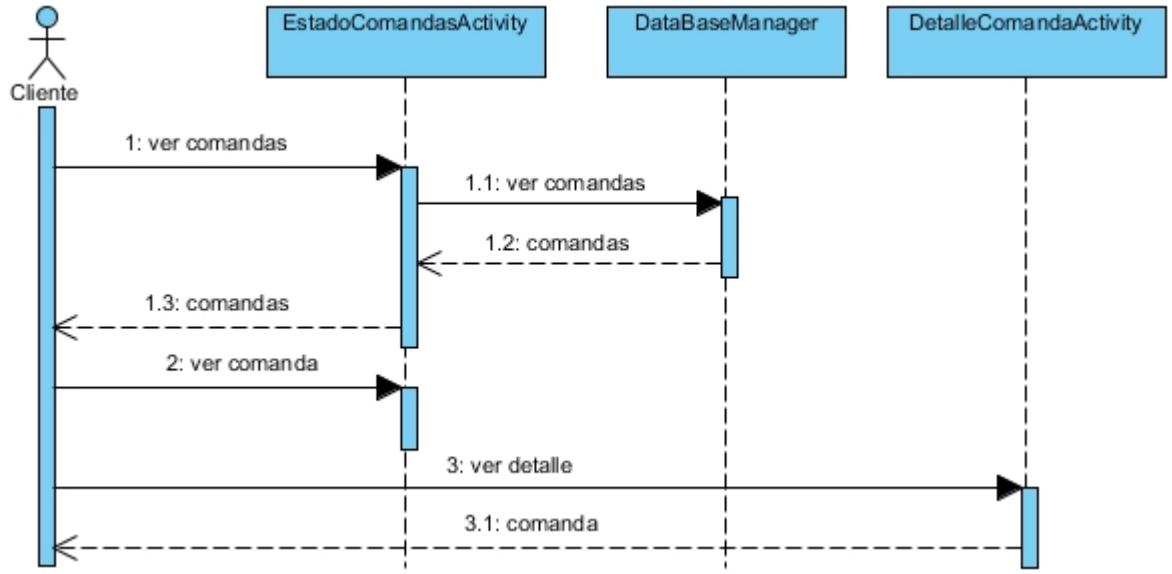


Figura 55: Diagrama de secuencias de consultar como cliente.

DI20. Consultar comandas como empleado. (Figura 56)

Flujo del diagrama.

1. Un empleado desde la clase MesaActivity pulsa sobre “Estado Comandas” para ver la lista de comandas de su mesa.
   1.1. MesaActivity llama a EstadoComandasActivity que obtiene la lista de comandas de DataBaseManager y las muestra al usuario.
   1.2. El usuario pulsa sobre una de las comandas y EstadoComandasActivity lanza una llamada a DetalleComandaActivity, que obtiene los datos de DataBaseManager y los muestra al cliente.
2. Un empleado desde el menú lateral accede a “Comandas” o desde SettingsActivity “Gesionar comandas” y el sistema llama a la clase ComandasActivity para ver la lista de comandas del local.
   2.1. ComandasActivity obtiene la lista de comandas de DataBaseManager y las muestra al empleado.
   2.2. El empleadopulsa sobre una de las comandas y ComandasActivity lanza una llamada a DetalleComandaActivity, que obtiene los datos de DataBaseManager y los muestra al empleado.

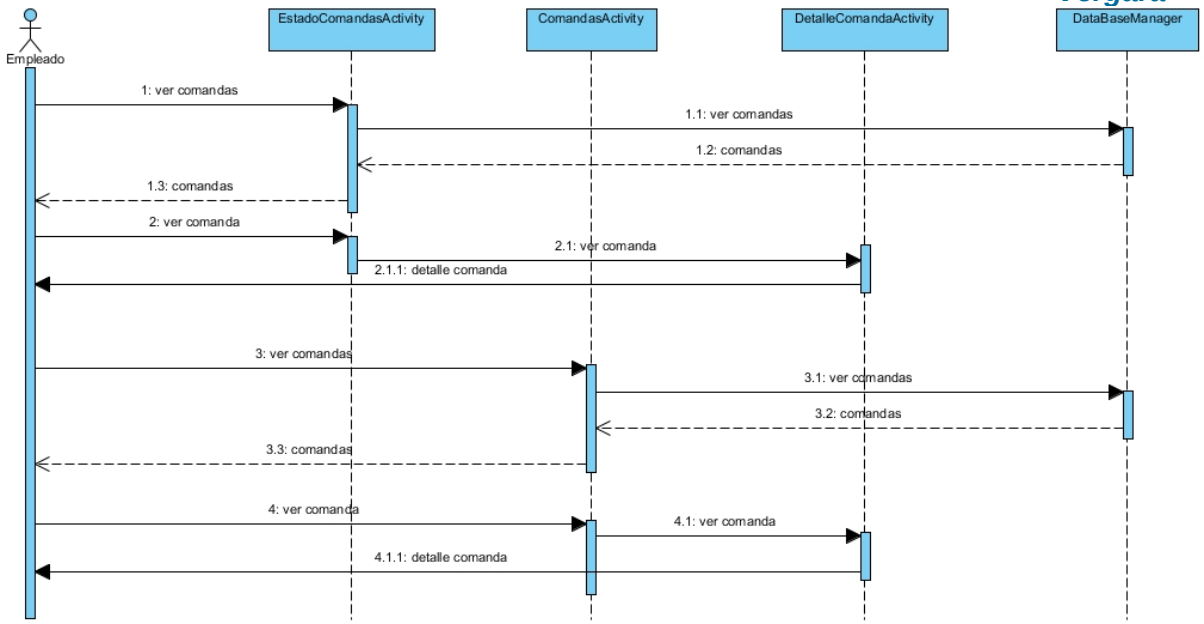


Figura 56: Diagrama de secuencias de consultar como empleado.

DI21. Actualizar pedidos. (Figura 57)

Flujo del diagrama.

1. Un empleado desde DetalleComandaActivity, pulsa sobre el botón para actualizar la comanda al siguiente estado.
2. DetalleComandaActivity llama a DataBaseManager y actualice la comanda en la Api.

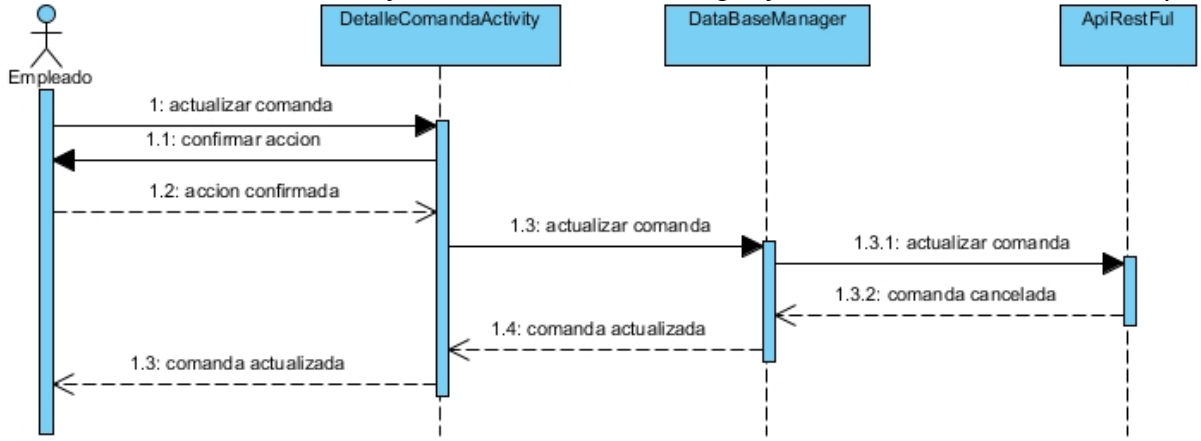


Figura 57: Diagrama de secuencias de actualizar pedidos.

DI22. Cancelar comanda como empleado.

Flujo del diagrama.

1. Un empleado desde DetalleComandaActivity, pulsa sobre cancelar.
2. DetalleComanadActivity envía un mensaje al empleado para confirmar si se desea cancelar la comanda y el empleado confirma la acción.
3. DetalleComandaActivity lanza un mensaje a DataBaseManager que cancela la comanda en la api.

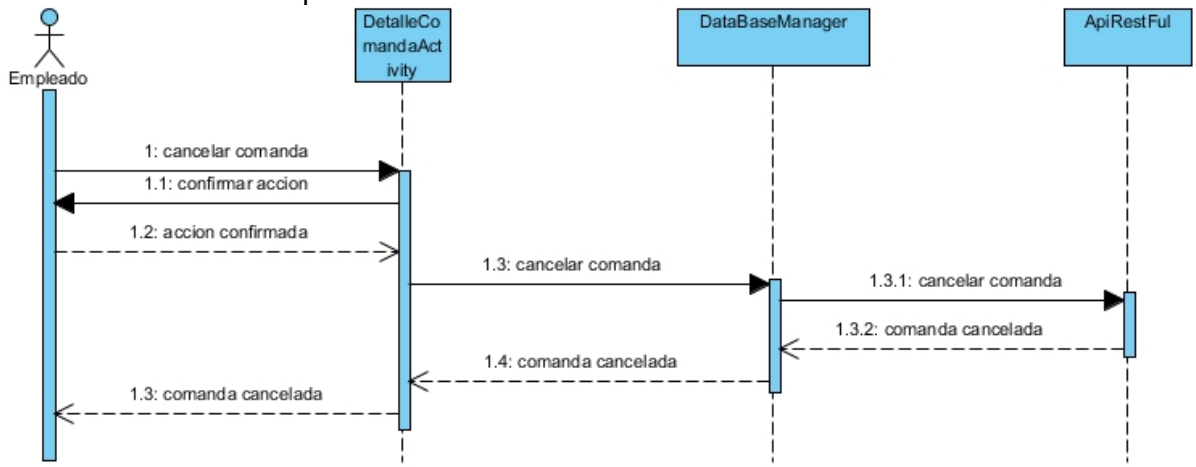


Figura 58.a: Cancelar comanda como empleado.

DI23. Cancelar comanda como cliente. (Figura 58)

Flujo del diagrama.

1. Un cliente, desde DetalleComandaActivity, pulsa sobre “Cancelar.”
2. A continuación, DetalleComandaActivity lanza una petición al empleado asignado a la mesa para cancelar la comanda.

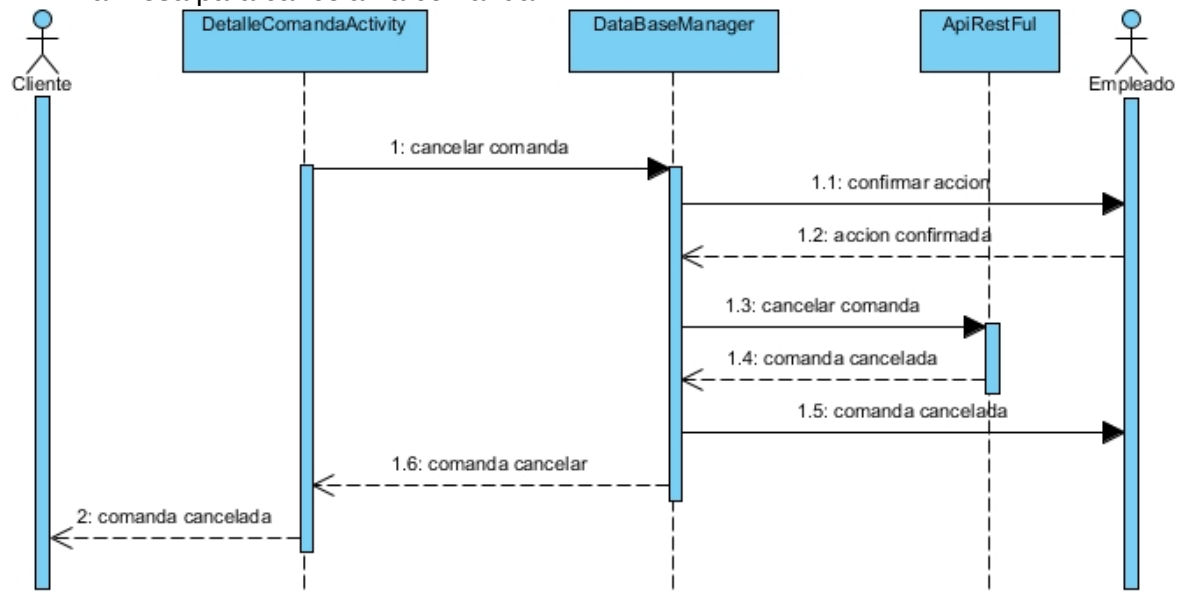


Figura 58: Diagrama de secuencias de actualizar comanda como cliente.

DI24. Ver gráficas. (Figura 59)

Flujo del diagrama.

1. Un empleado desde el menú lateral, accede a “Gráficas”.
2. EstadisticasActivity llama a DataBaseManager y obtiene los datos de todas las comandas del local.
3. EstadisticasActivity procesa los datos obtenidos y muestra una gráfica con las ventas de los últimos 6 meses.

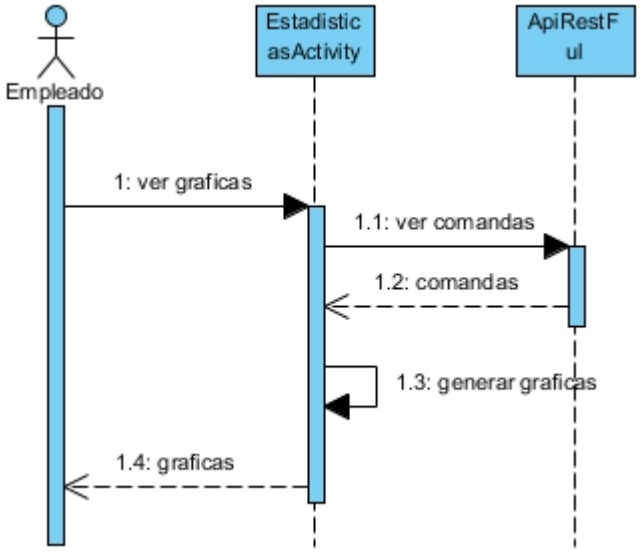


Figura 59: Diagrama de secuencias de actualizar comanda como empleado.

DI25. Buscar locales. (Figura 60)

Flujo del diagrama.

1. El cliente desde MapaActivity selecciona un local desde el mapa o buscador.
2. MapaActivity llama a DataBaseManager y esta a su vez llama a ApiRestFul para obtener los datos desde la api.
3. ApiRestFul devuelve los datos a DataBaseManager y este, a su vez, al cliente.
4. El cliente pulsa sobre “como llegar” y el sistema llama a la api de GoogleMaps para iniciar el cómo llegar.

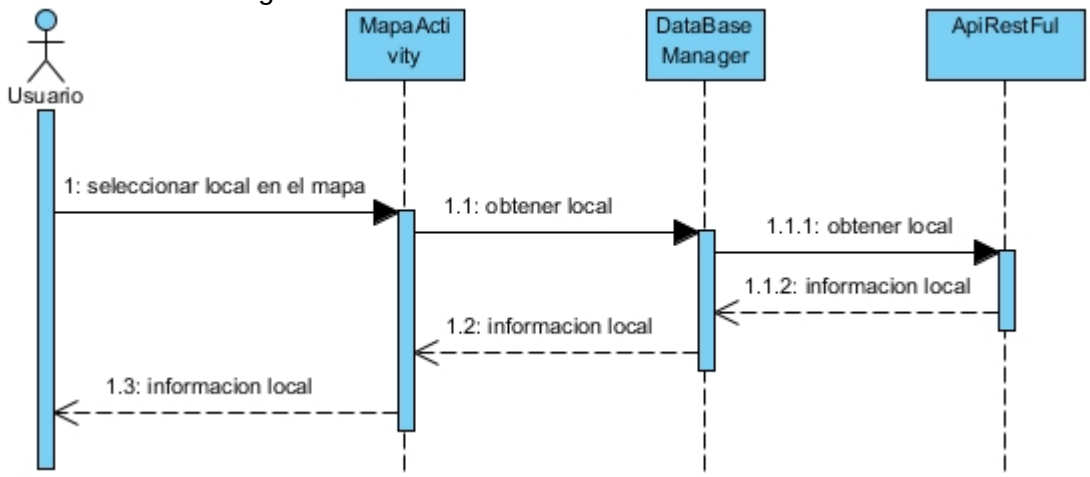


Figura 60: Diagrama de secuencias de buscar locales.

DI26. Como llegar. (Figura 61)

Flujo del diagrama.

1. Una vez elegido un local, el cliente pulsa sobre “cómo llegar” y el sistema llama a la api de GoogleMaps para iniciar el cómo llegar.

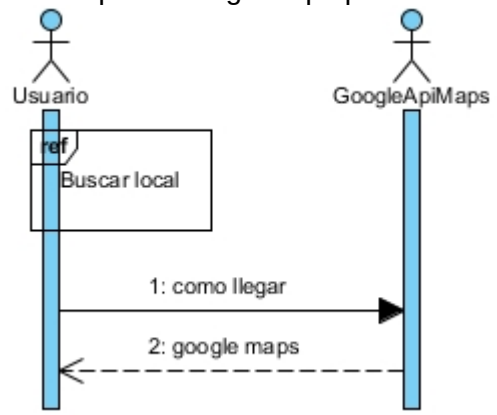


Figura 61: Diagrama de secuencias de como llegar.

DI27. Consultar el estado de las mesas de un local. (Figura 62)

Flujo del diagrama.

1. Una vez elegido un local, el usuario selecciona el local para acceder a sus datos y MapaActicity lanza un mensaje a EspaciosActivity para que se muestre la lista de mesas del local.
2. EspaciosActivity llama a ApiRestFul para que le proporcione los datos y se los muestra al usuario.

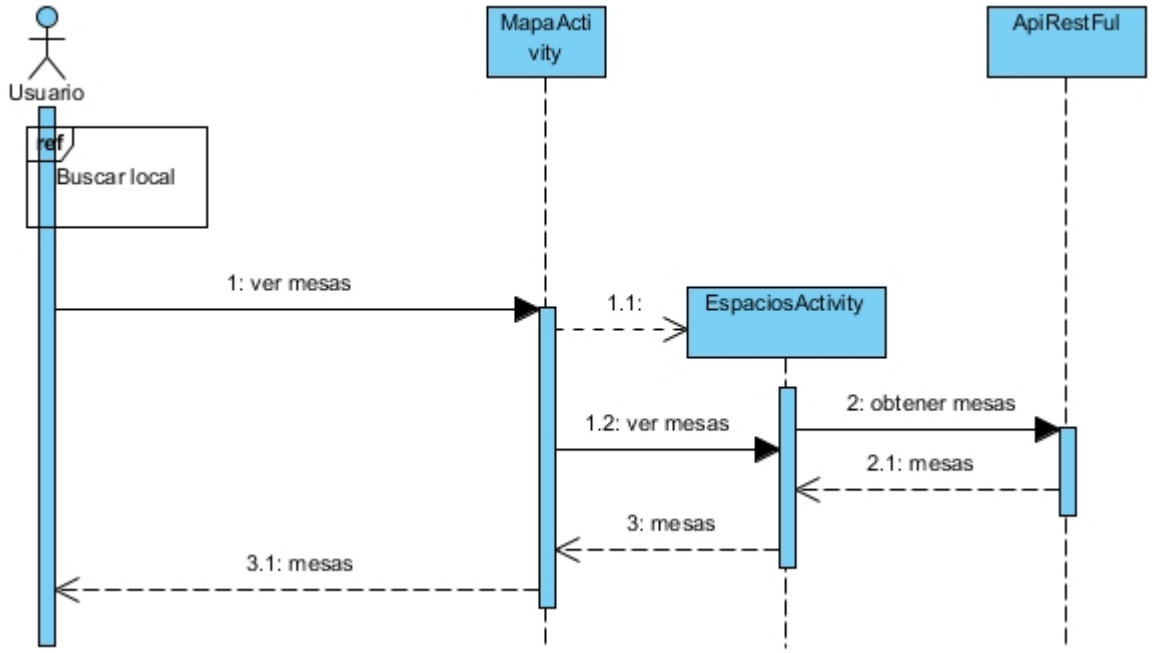


Figura 62: Diagrama de secuencias de consultar el estado de las mesas.

DI28.Consultar la carta de un local. (Figura 63)

Flujo del diagrama.

1. Una vez elegido un local, el usuario selecciona el local para acceder a sus datos y MapaActicity lanza un mensaje a StockActivity para que se actualice la lista de productos e ingredientes del local.
2. StockActivity llama a ApiRestFul para que le proporcione los datos y se los muestra al usuario.

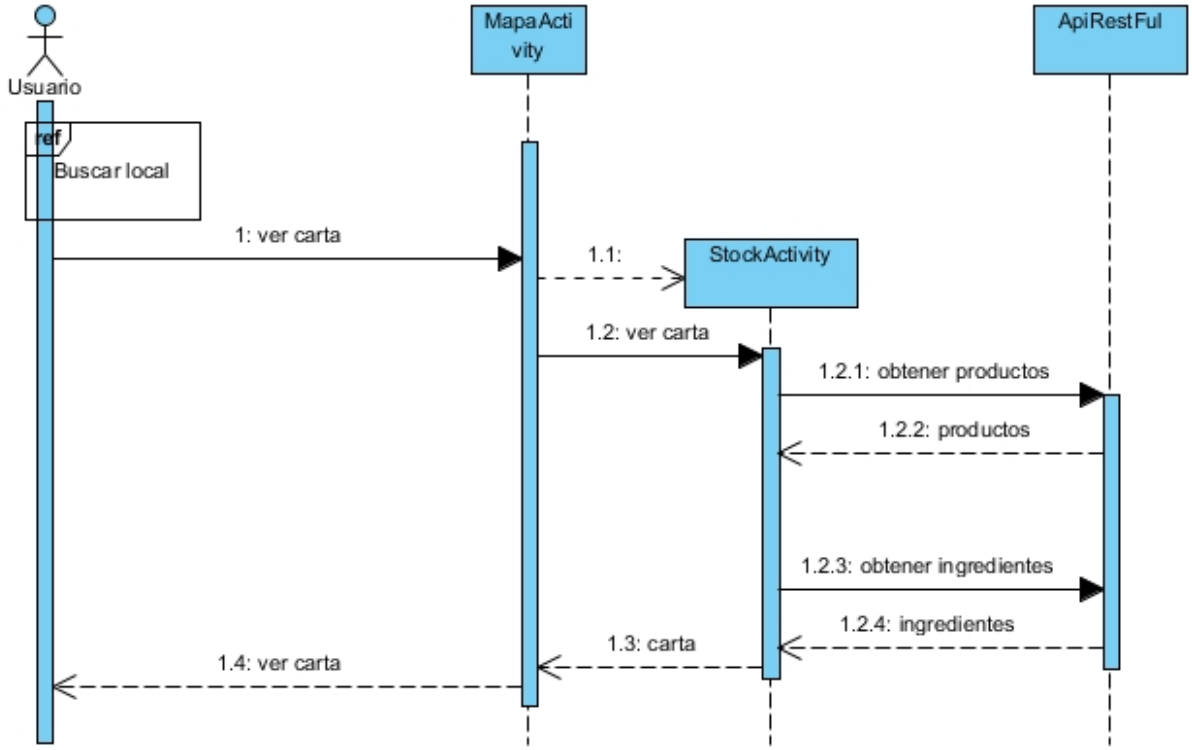


Figura 63: Diagrama de secuencias de consultar la carta de un local.

DI29. Contactar con el local. (Figura 64)

Flujo del diagrama.

1. Una vez elegido un local, el usuario selecciona el local y pulsa “obtener información” para acceder a sus datos y MapaActivity lanza un mensaje a RestApiRestful para que muestre los datos de contacto del local.

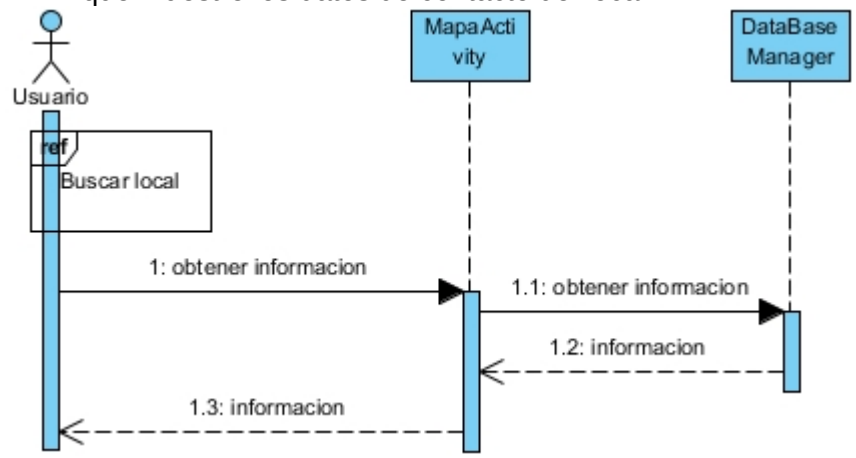


Figura 64: Diagrama de secuencias de contactar con el local.

DI30. Escanear un código QR. (Figura 65)

Flujo del diagrama.

1. Un usuario logueado y sin ninguna mesa asociada desde la pantalla InicioCliente selecciona “Escanear codigo”.
2. La clase InicioCliente llama a la clase SimpleEscanerActivity y esta clase inicia el lector QR.
3. Una vez que el lector detecte un código QR, llama a la clase DataBaseManager para obtener la información del código.
4. DataBaseManager llama a la clase ApiRestFul para generar los datos en DataBaseManager y devuelve la mesa escaneada al usuario.

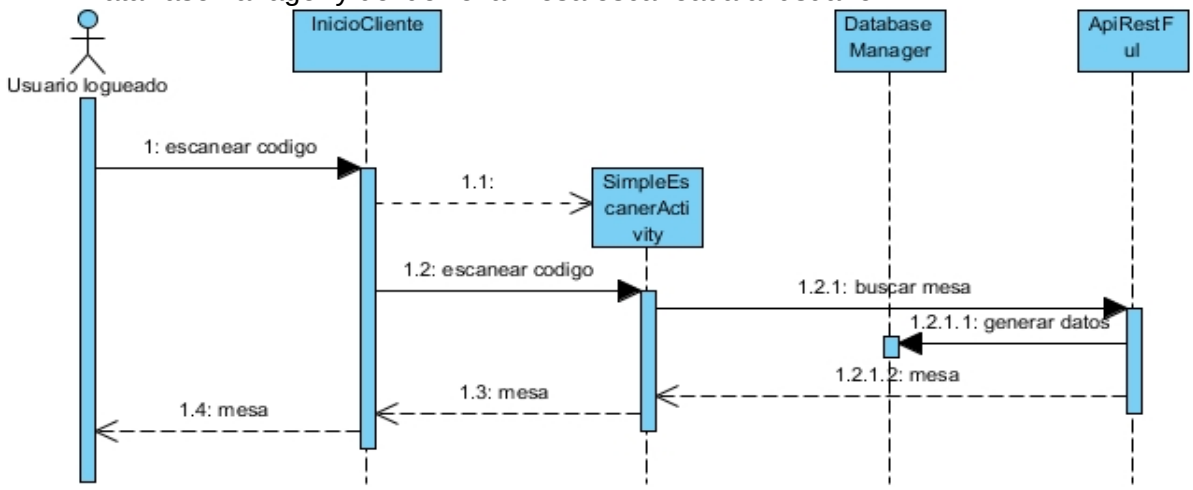


Figura 65: Diagrama de secuencias de escanear un código QR.

DI31. Generar cuenta como empleado. (Figura 66)

Flujo del diagrama.

1. Un empleado del local desde EstadoComandaActivity pulsa sobre “Generar cuenta”.
2. EstadoComandaActivity lanza un mensaje al empleado para establecer el modo de pago.
3. El empleado selecciona el modo de pago y EstadoComandaActivity llama a DataBaseManager para generar la cuenta.
4. DataBaseManager mediante métodos internos genera la cuenta y devuelve el importe a pagar a EstadoComandaActicity, que lo devuelve al empleado.
5. Una vez que se haya confirmado el pago, DataBaseManager lanza un mensaje a ApiRestFul para limpiar la mesa de comandas.

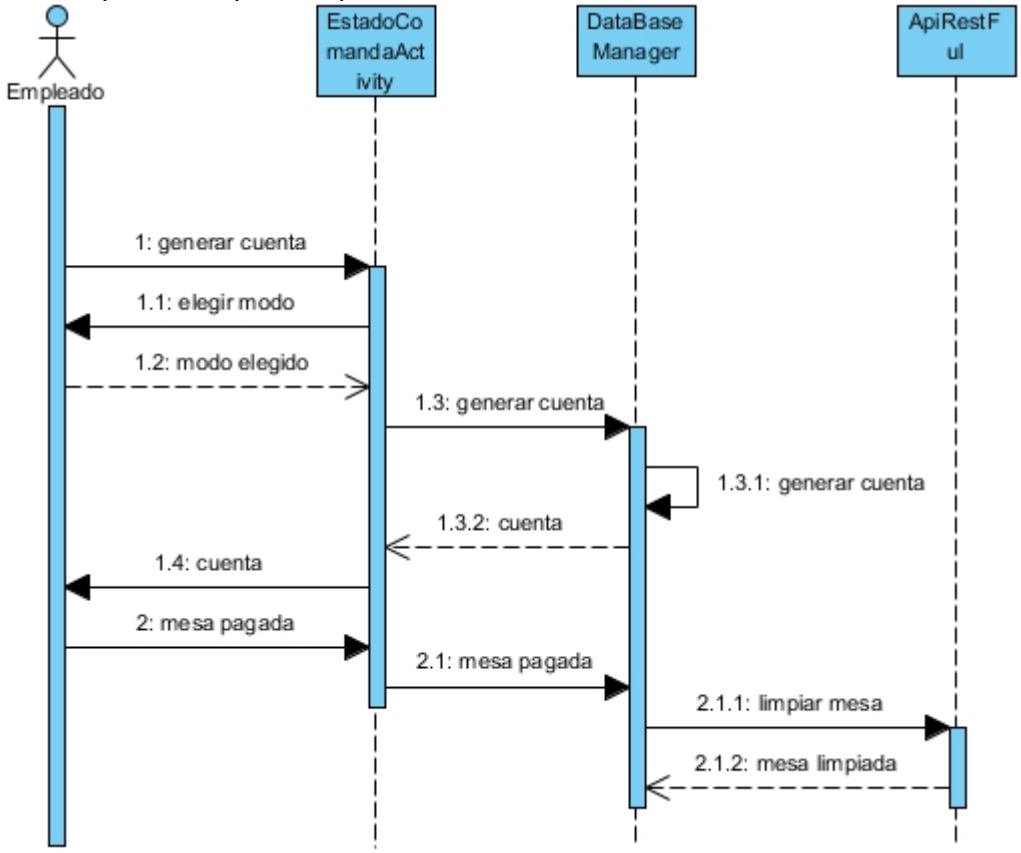


Figura 66: Diagrama de secuencias de generar cuenta como empleado.

DI32. Generar cuenta como cliente. (Figura 67)

Flujo del diagrama.

1. El cliente desde EstadoComandaActicity, pulsa sobre GenerarCuenta.
2. EstadoComandaActivity lanza un mensaje al cliente para establecer el modo de pago.
3. El cliente selecciona el modo de pago y EstadoComandaActivity llama a DataBaseManager para que gener la cuenta.
4. A su vez, EstadoComandaActivity lanza un mensaje al Empleado asignado a la mesa para que atienda la petición.

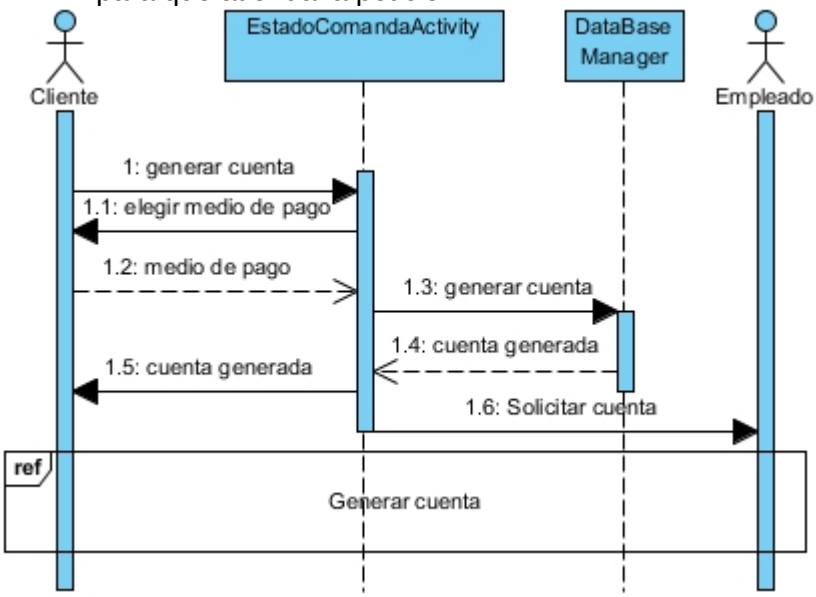


Figura 67: Diagrama de secuencias de generar cuenta como cliente.

DI33. Limpiar mesa. (Figura 68)

Flujo del diagrama.

1. Un empleado desde MainActivity, mantiene pulsado sobre una de las mesas que no está en estado “Libre”.
2. MainActivity lanza un mensaje para confirmar que se desea limpiar la mesa y el empleado confirma la acción.
3. Una vez confirmada la acción, MainActivity lanza un mensaje a DataBaseManager para limpiar la mesa y esta lanza un mensaje a ApiRestFul para que limpie la mesa en la api.
4. El sistema informa al empleado de que la mesa ha sido limpiada.

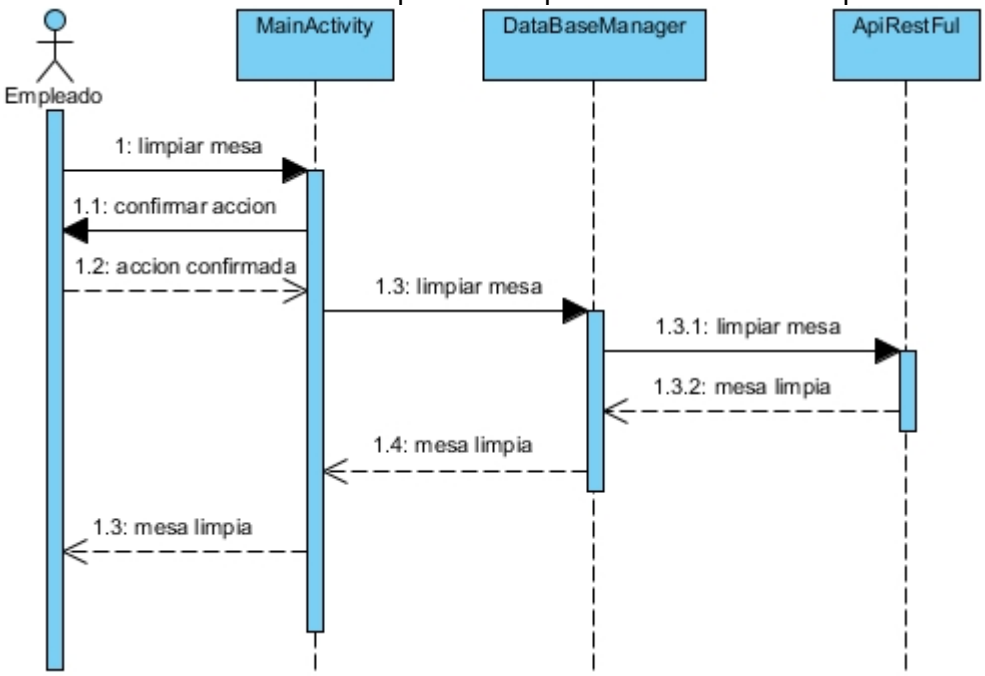


Figura 68: Diagrama de secuencias de limpiar mesa.

DI34. Valorar servicio. (Figura 69)

Flujo del diagrama.

1. Una vez acabado el proceso de generar cuenta, la clase EstadoComandaActivity lanza al cliente una petición para valorar el servicio ofrecido por el local.
2. El cliente realiza la valoración y EstadoComandaActivity envía esta valoración a DataBaseManager.
3. DataBaseManager envía la información a ApiRestFul para que guarde la valoración en la api y devuelve un mensaje informando que la valoración se ha realizado con éxito.

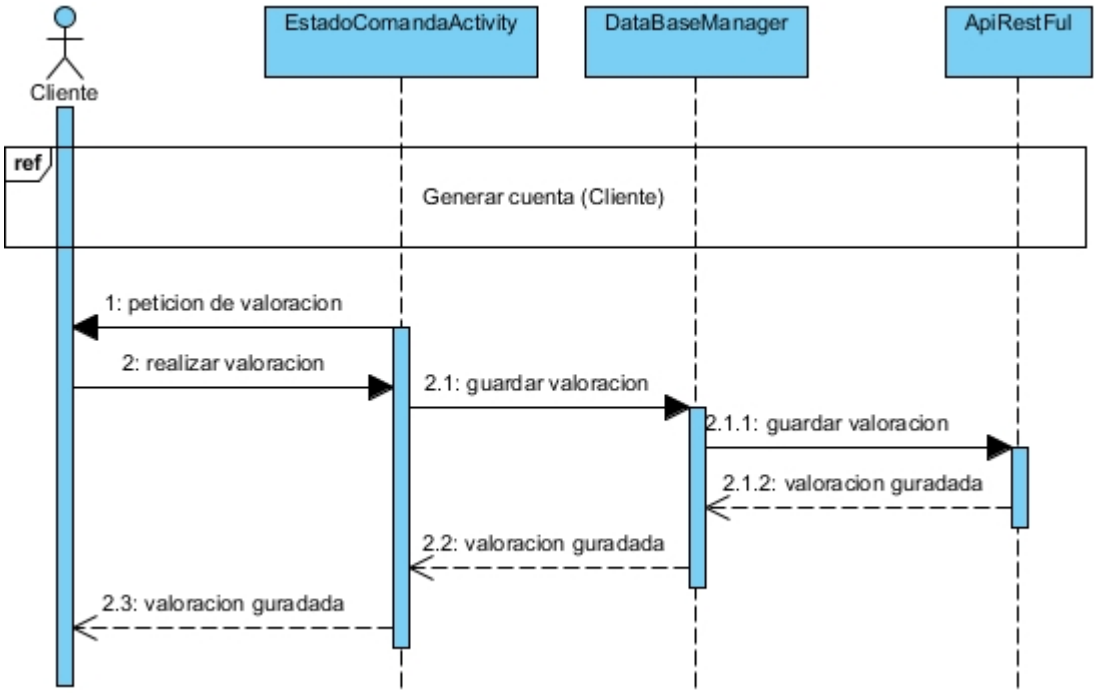


Figura 69: Diagrama de secuencias de generar valorar servicio.

5.5. Diagramas de secuencias de la aplicación web.

DI01. Registrarse. (Figura 70)

Flujo del diagrama.

1. Un usuario desde la página de inicio, pulsa sobre “Crear una cuenta gratis”.
2. La página web le lleva a la página de registro.
3. El usuario rellena los datos de registro.
4. La página valida los datos y muestra al usuario la página de Login.

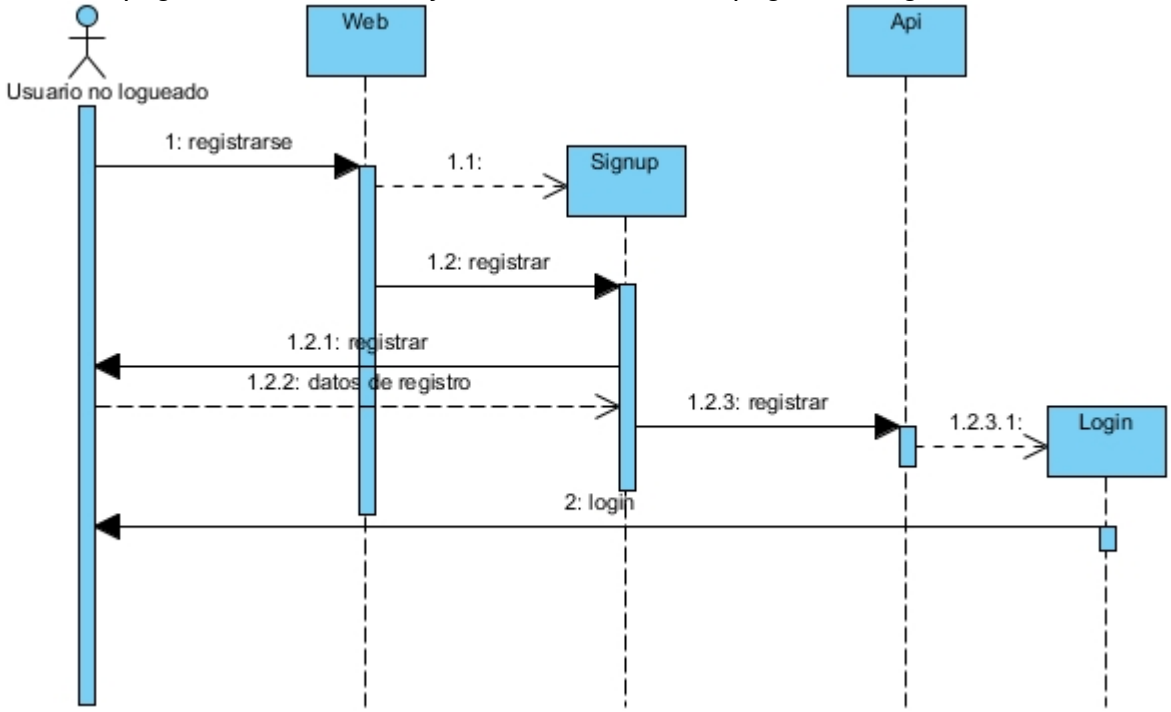


Figura 70: Diagrama de secuencias de registrarse.

DI02 Login y Logout. (Figura 71)

Flujo del diagrama.

1. El usuario desde la página principal, pulsa sobre “Iniciar sesión o crear cuenta”
   1.1. La página devuelve al usuario el formulario para iniciar sesión.
   1.2. El usuario rellena los datos.
   1.3. La página web valida los datos y permite al usuario acceder al sitio.
2. El usuario desde el menú lateral pulsa sobre “Cerrar sesión”
   2.1. La página web cierra la sesión del usuario y muestra la pantalla de inicio.

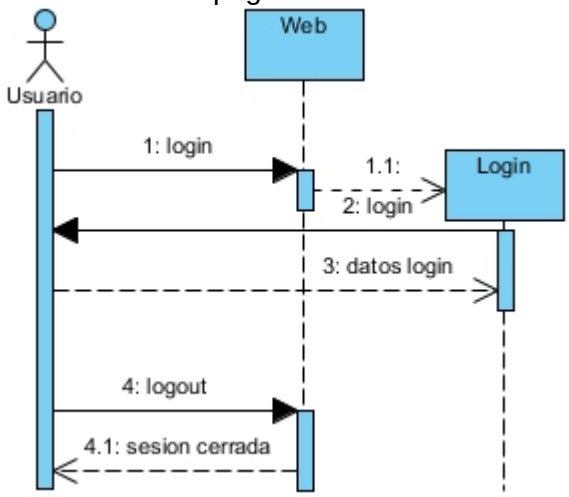


Figura 71: Diagrama de secuencias de login y logout.

DI03 Crear empleado. (Figura 72)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar empleados.
2. La página web muestra el panel de control de empleados.
3. El usuario pulsa sobre “+” y rellena los datos mostrados.
4. La página web valida los datos y crea el empleado en la api.

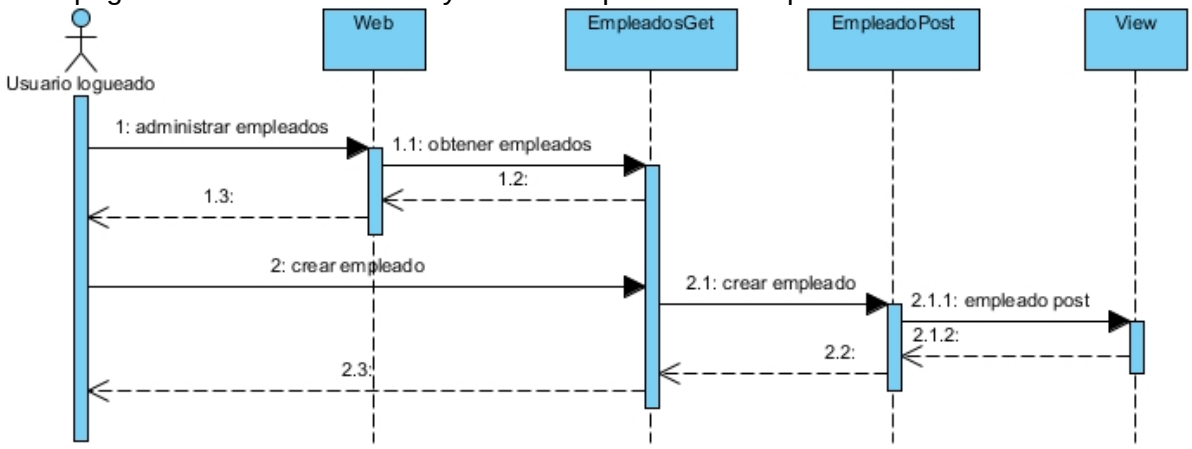


Figura 72: Diagrama de secuencias de crear empleado.

DI04 Eliminar empleado. (Figura 73)

Flujo del diagrama.

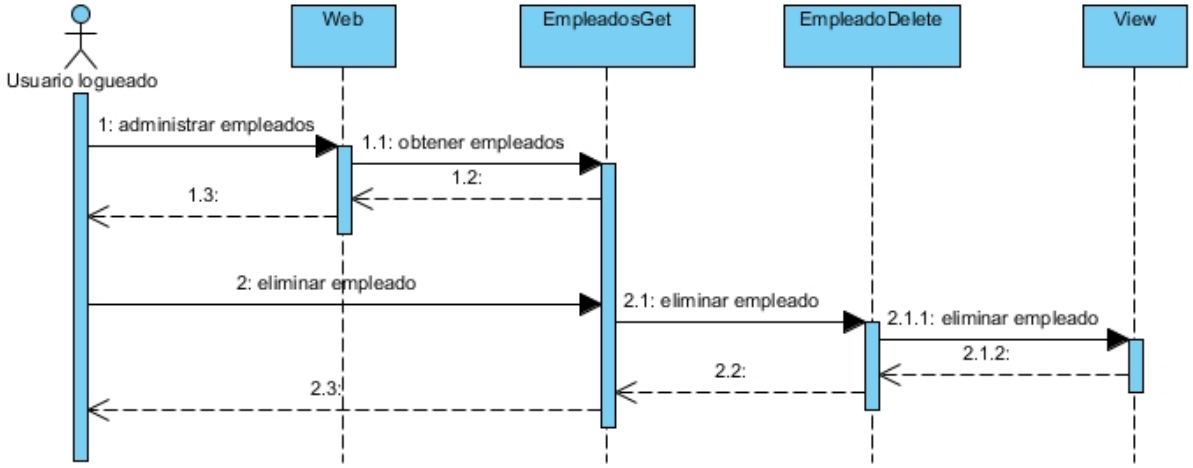


Figura 73: Diagrama de secuencias de eliminar empleado.

DI05 Crear mesa. (Figura 74)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar mesas”.
2. La página web muestra el panel de control de mesas.
3. El usuario pulsa sobre “+” y rellena los datos mostrados.
4. La página web valida los datos y crea la mesa en la api.

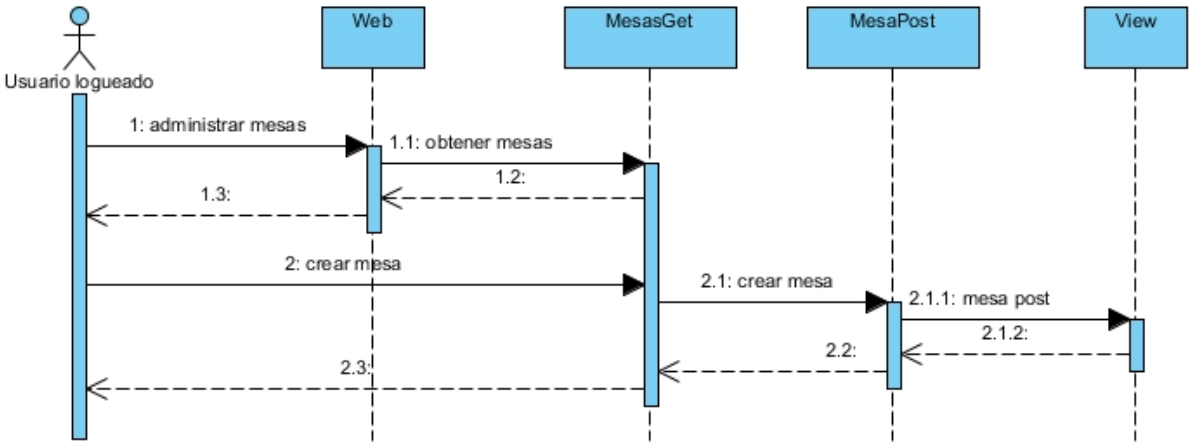


Figura 74: Diagrama de secuencias de crear mesa.

DI06 Modificar mesa. (Figura 75)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar mesas”.
2. La página web muestra el panel de control de mesas.
3. El usuario pulsa sobre “Editar” en una de las mesas y rellena los datos.
4. La página web valida los datos y modifica la mesa en la api.

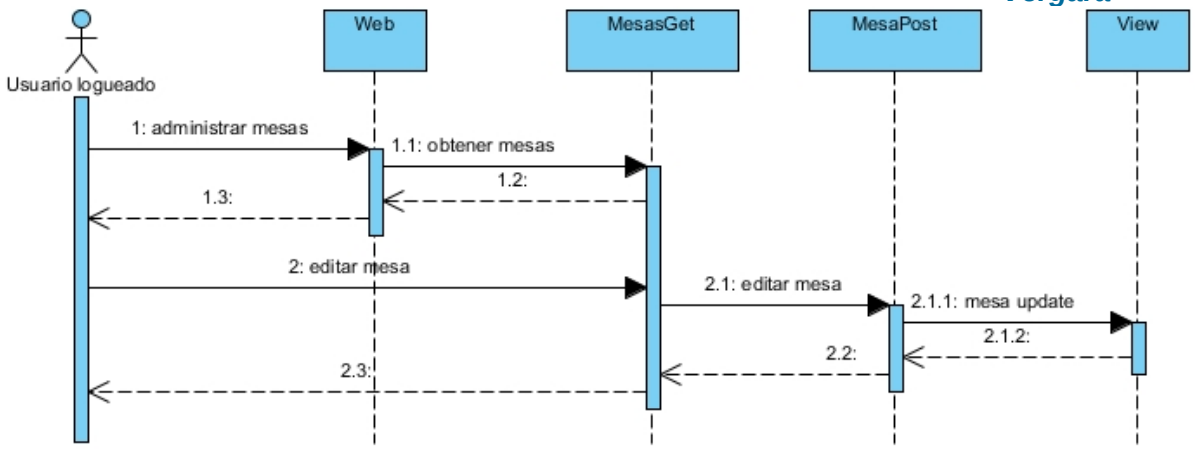


Figura 75: Diagrama de secuencias de modificar mesa.

DI07 Eliminar mesa. (Figura 76)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar mesas”.
2. La página web muestra el panel de control de mesas.
3. El usuario pulsa sobre “Eliminar” en una de las mesas y confirma la acción.
4. La página web elimina la mesa en la api.

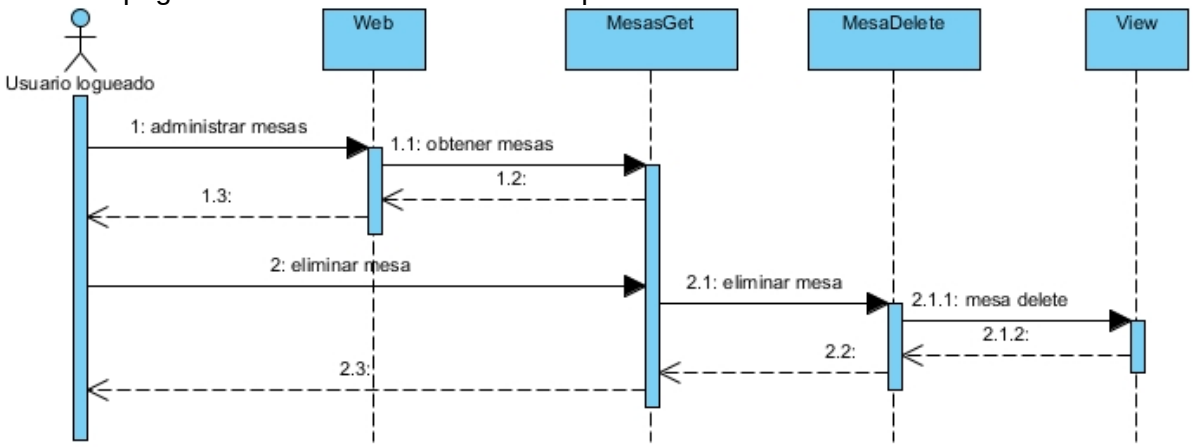


Figura 76: Diagrama de secuencias de eliminar mesa.

DI08 Crear espacio. (Figura 77)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar mesas”.
2. La página web muestra el panel de control de mesas.
3. El usuario pulsa sobre “+” en el apartado de “Espacio”.
4. La página web muestra el panel de control de espacios.
5. El usuario pulsa sobre “+” y rellena los datos mostrados.
6. La página web valida los datos y crea el espacio en la api.

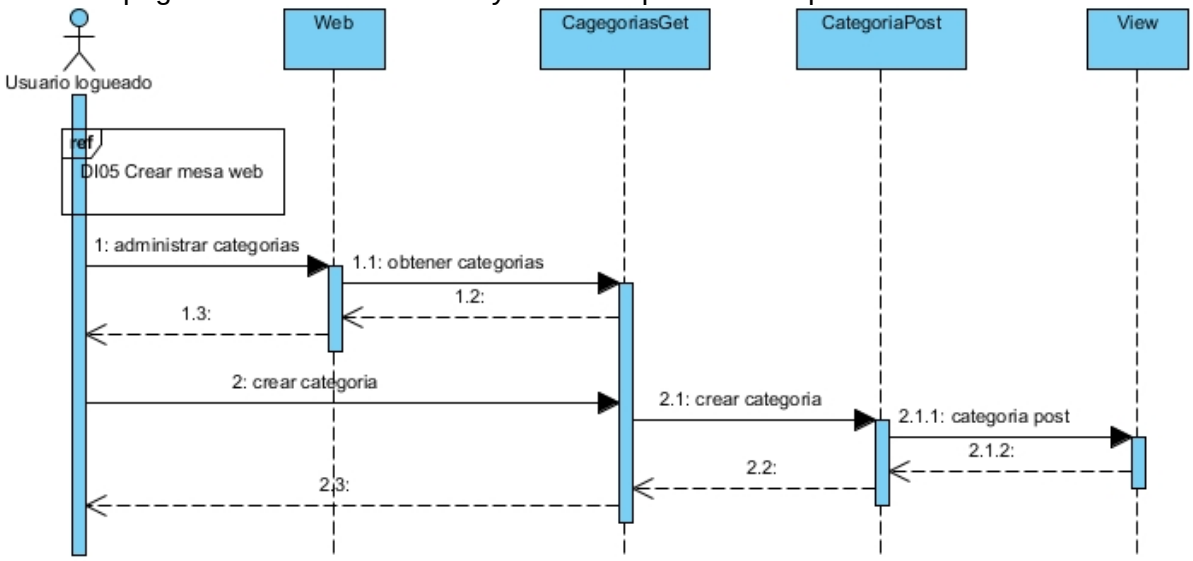


Figura 77: Diagrama de secuencias de crear espacio.

DI09 Modificar espacio. (Figura 78)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar mesas”.
2. La página web muestra el panel de control de mesas.
3. El usuario pulsa sobre “+” en el apartado de “Espacio”.
4. La página web muestra el panel de control de espacios.
5. El usuario pulsa sobre “Editar” en uno de los espacios y rellena los datos.
6. La página web valida los datos y modifica el espacio en la api.

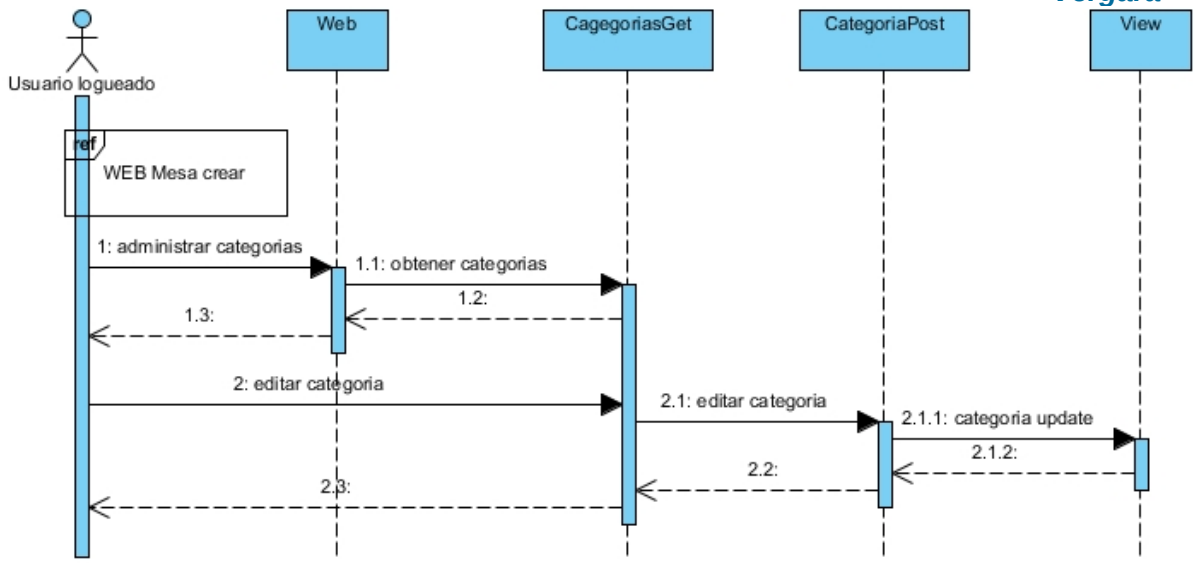


Figura 78: Diagrama de secuencias de modificar espacio.

DI10 Eliminar espacio. (Figura 79)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar mesas”.
2. La página web muestra el panel de control de mesas.
3. El usuario pulsa sobre “+” en el apartado de “Espacio”.
4. La página web muestra el panel de control de espacios.
5. El usuario pulsa sobre “Eliminar” en uno de los espacios y confirma la acción.
6. La página web elimina el espacio en la api.

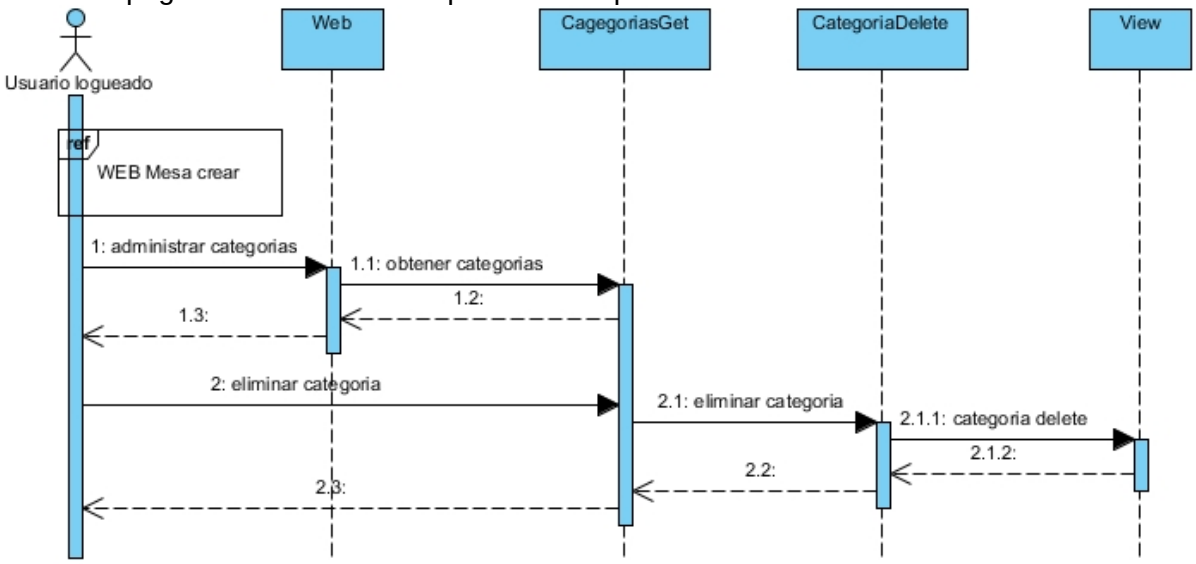


Figura 79: Diagrama de secuencias de eliminar espacio.

DI11 Crear producto. (Figura 8)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar productos”.
2. La página web muestra el panel de control de productos.
3. El usuario pulsa sobre “+” y rellena los datos mostrados.
4. La página web valida los datos y crea el producto en la api.

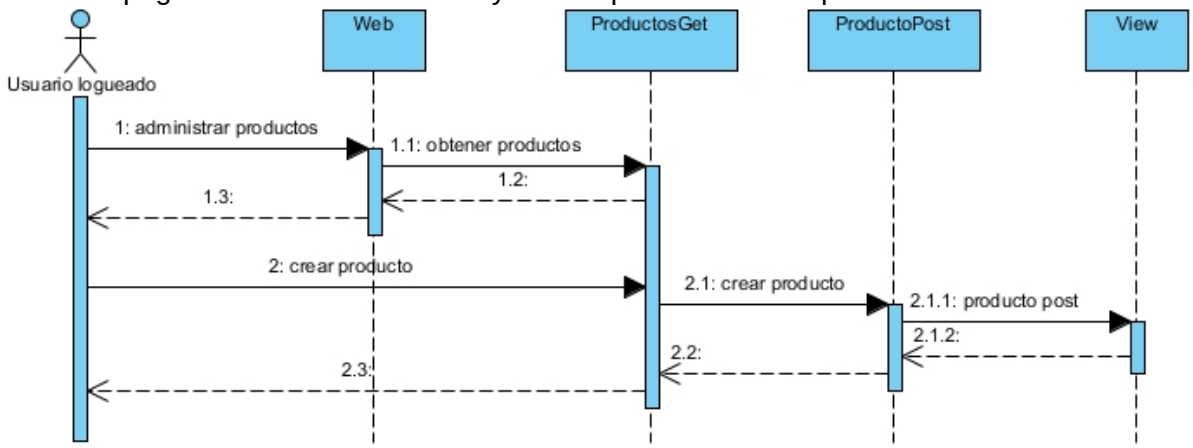


Figura 80: Diagrama de secuencias de crear producto.

DI12 Modificar producto. (Figura 81)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar productos”.
2. La página web muestra el panel de control de productos.
3. El usuario pulsa sobre “Editar” en uno de los productos y rellena los datos.
4. La página web valida los datos y modifica el producto en la api.

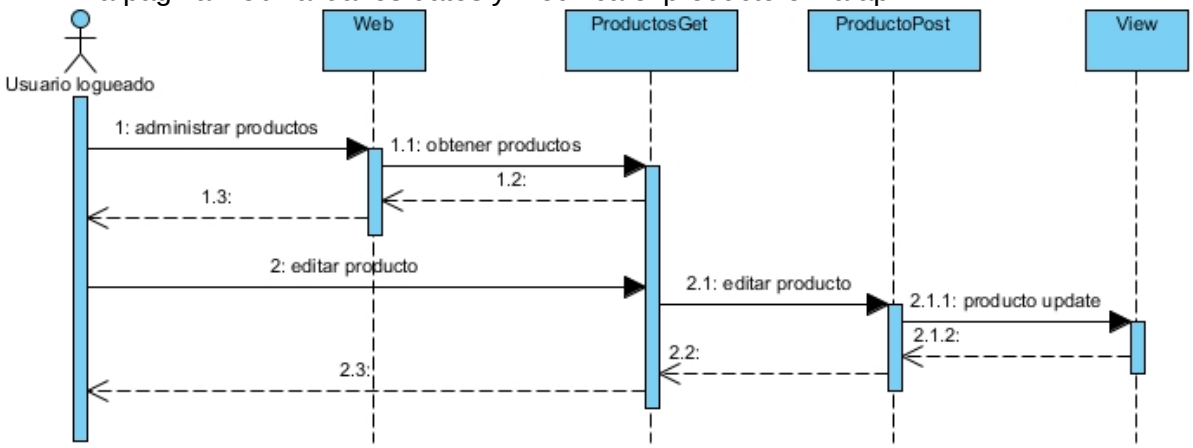


Figura 81: Diagrama de secuencias de modificar producto.

DI13 Eliminar producto. (Figura 82)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar productos”.
2. La página web muestra el panel de control de productos.
3. El usuario pulsa sobre “Eliminar” en uno de los productos y confirma la acción.
4. La página web elimina el producto en la api.

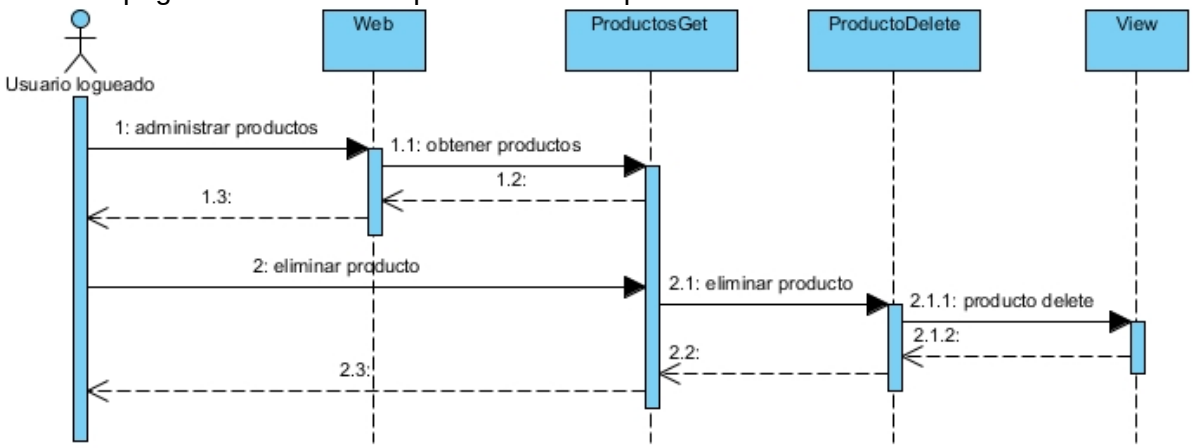


Figura 82: Diagrama de secuencias de eliminar producto.

DI14 Crear ingrediente. (Figura 83)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar ingredientes”.
2. La página web muestra el panel de control de ingredientes.
3. El usuario pulsa sobre “+” y rellena los datos mostrados.
4. La página web valida los datos y crea el ingrediente en la api.

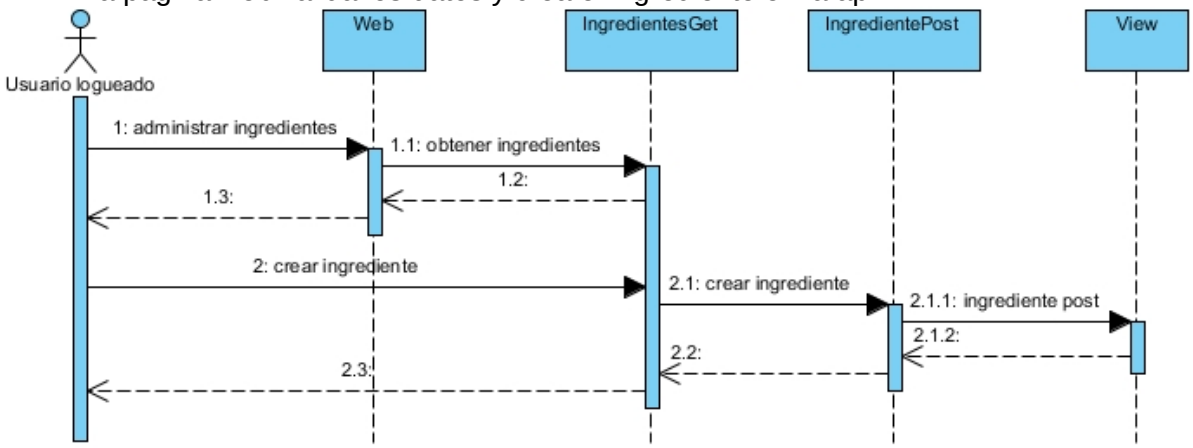


Figura 83: Diagrama de secuencias de crear ingrediente.

DI15 Modificar ingrediente. (Figura 84)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar ingredientes”.
2. La página web muestra el panel de control de ingredientes.
3. El usuario pulsa sobre “Editar” en uno de los ingredientes y rellena los datos.
4. La página web valida los datos y modifica el ingrediente en la api.

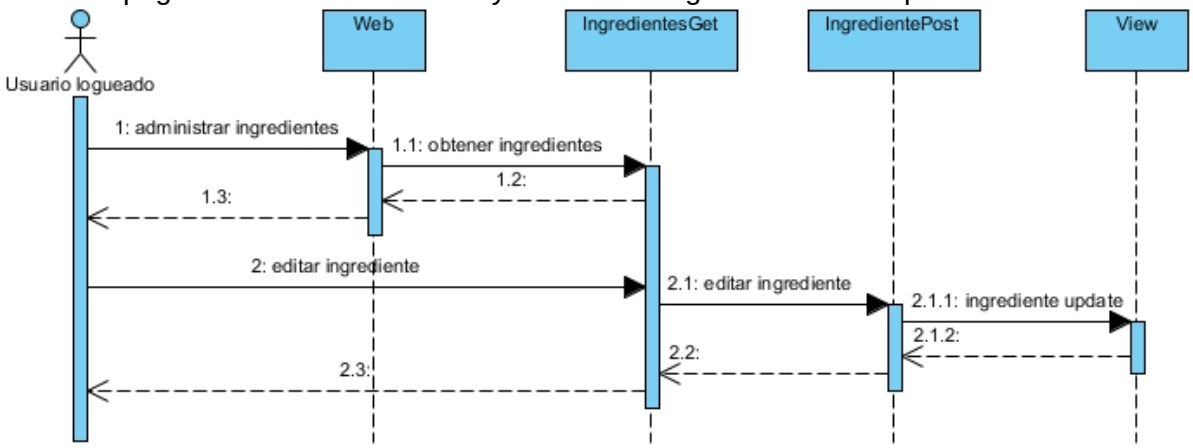


Figura 84: Diagrama de secuencias de modificar ingrediente.

DI16 Eliminar ingrediente. (Figura 85)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar ingredientes”.
2. La página web muestra el panel de control de ingredientes.
3. El usuario pulsa sobre “Eliminar” en uno de los ingredientes y confirma la acción.
4. La página web elimina el ingrediente en la api.

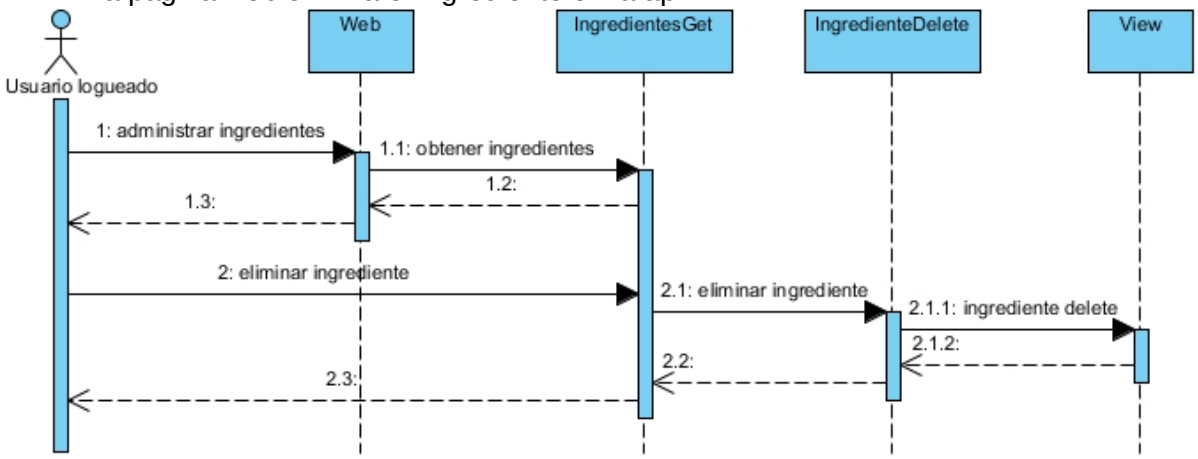


Figura 85: Diagrama de secuencias de eliminar ingrediente.

DI17 Crear categoría. (Figura 86)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar productos”.
2. La página web muestra el panel de control de productos.
3. El usuario pulsa sobre “+” en el apartado de “Categoría”.
4. La página web muestra el panel de control de categorías.
5. El usuario pulsa sobre “+” y rellena los datos mostrados.
6. La página web valida los datos y crea la categoría en la api.

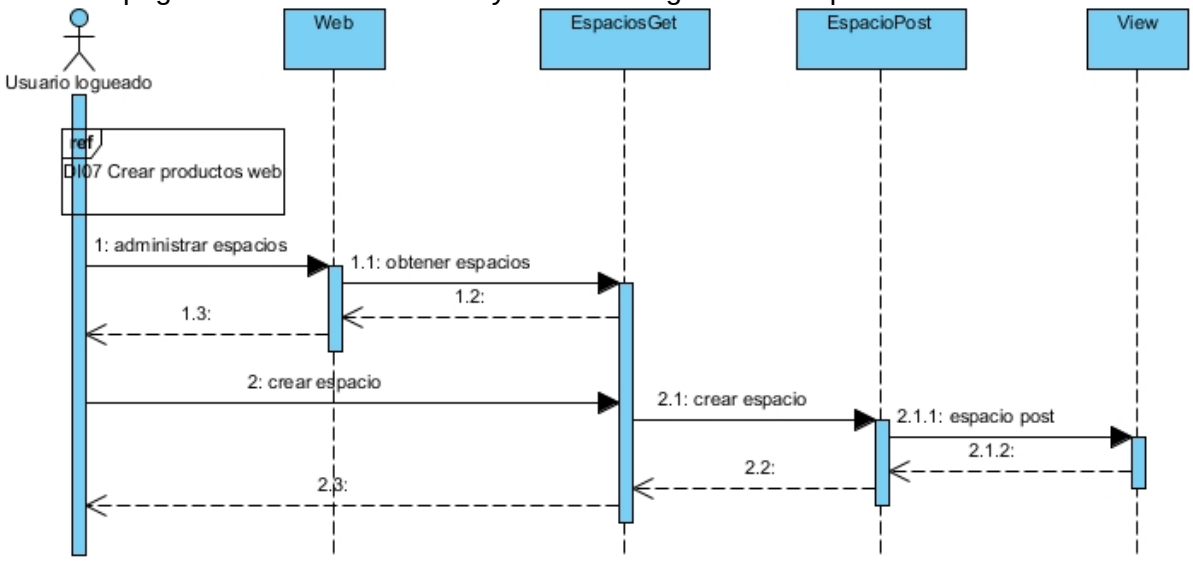


Figura 86: Diagrama de secuencias de crear categoría.

DI18 Modificar categoría. (Figura 87)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar productos”.
2. La página web muestra el panel de control de productos.
3. El usuario pulsa sobre “+” en el apartado de “Categoría”.
4. La página web muestra el panel de control de categorías.
5. El usuario pulsa sobre “Editar” en una de las categorías y rellena los datos.
6. La página web valida los datos y modifica la categoría en la api.

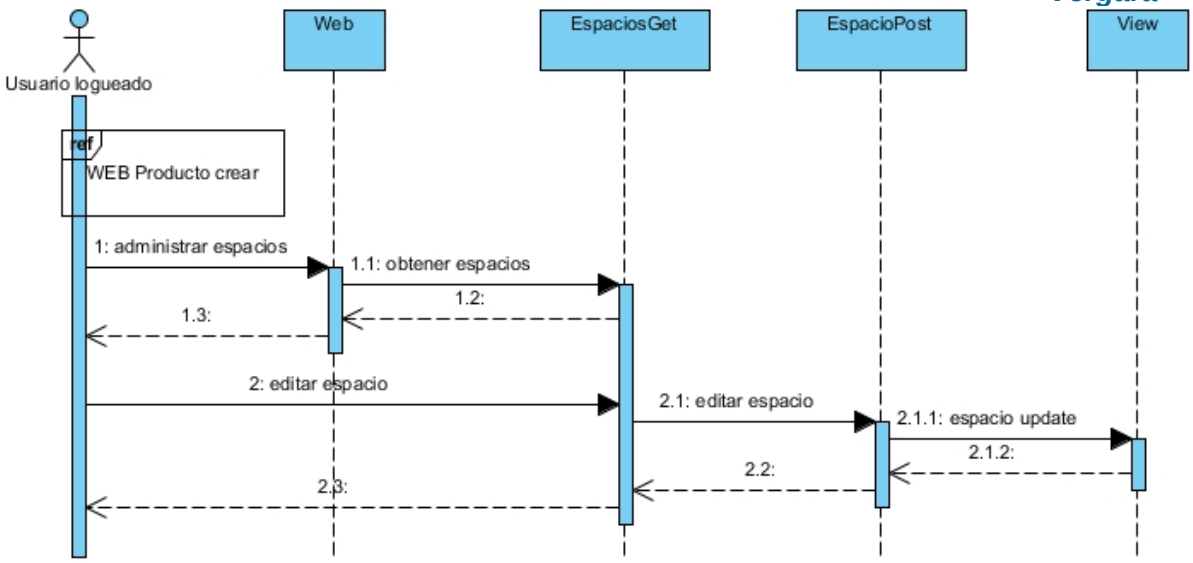


Figura 87: Diagrama de secuencias de modificar categoría.

DI19 Eliminar categoría. (Figura 88)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar productos”.
2. La página web muestra el panel de control de productos.
3. El usuario pulsa sobre “+” en el apartado de “Categoría”.
4. La página web muestra el panel de control de categorías.
5. El usuario pulsa sobre “Eliminar” en una de las categorías y confirma la acción.
6. La página web elimina la categoría en la api.

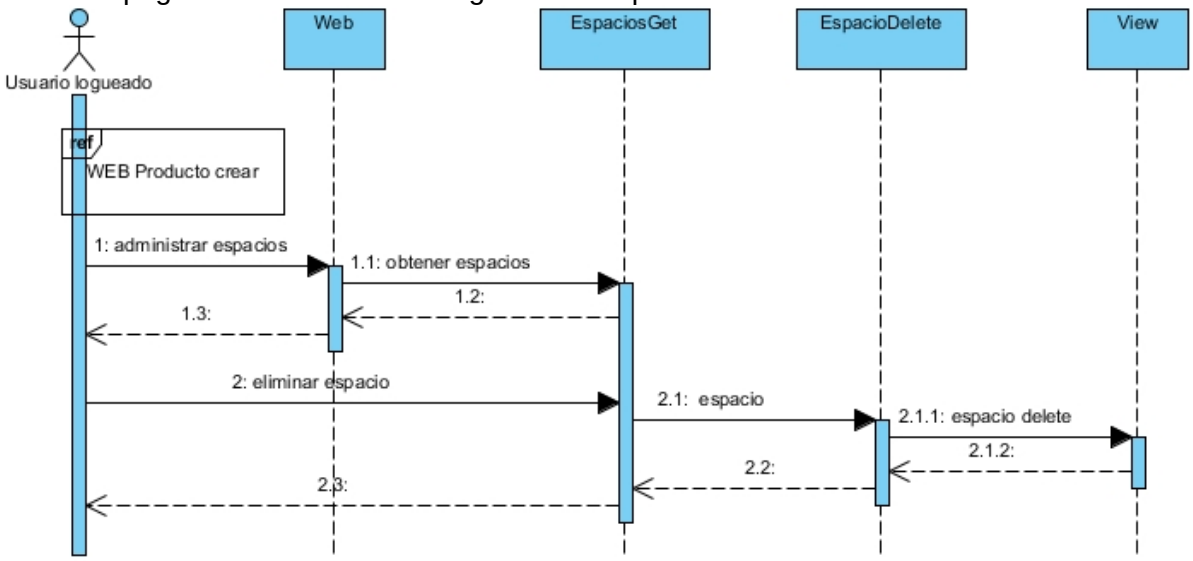


Figura 88: Diagrama de secuencias de eliminar categoría.

DI20 Crear especial. (Figura 89)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar especiales”.
2. La página web muestra el panel de control de especiales.
3. El usuario pulsa sobre “+” y rellena los datos mostrados.
4. La página web valida los datos y crea el especial en la api.

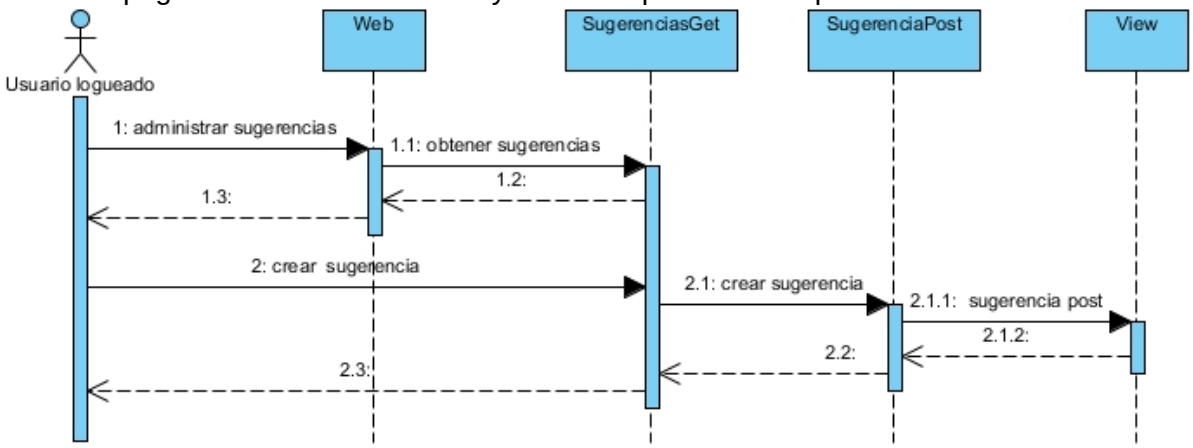


Figura 89: Diagrama de secuencias de crear especial.

DI21 Modificar especial. (Figura 90)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar especiales”.
2. La página web muestra el panel de control de especiales.
3. El usuario pulsa sobre “Editar” en uno de los especiales y rellena los datos.
4. La página web valida los datos y modifica el especial en la api.

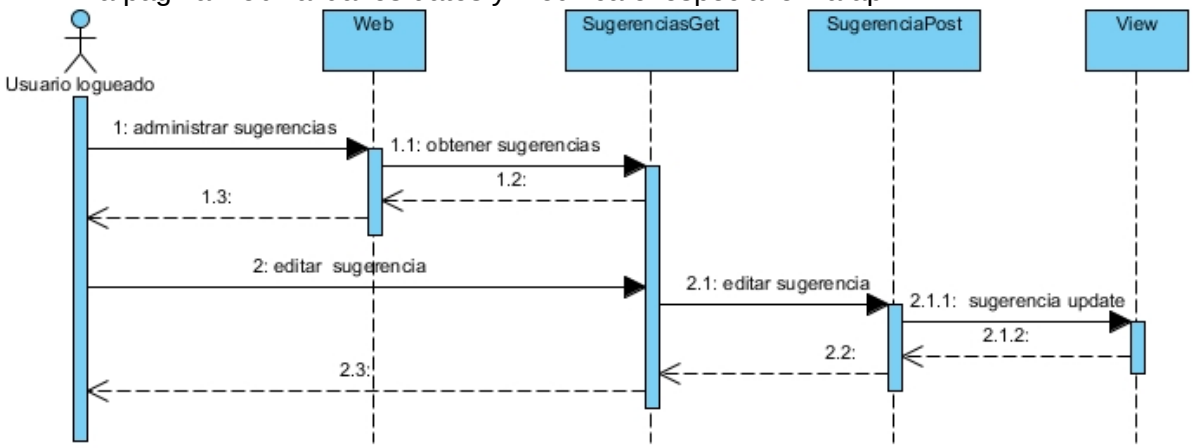


Figura 90: Diagrama de secuencias de modificar especial.

DI22 Eliminar especial. (Figura 91)

Flujo del diagrama.

1. Un usuario accede al menú lateral y pulsa sobre “Administrar especiales”.

2. La página web muestra el panel de control de especiales.
3. El usuario pulsa sobre “Eliminar” en uno de los especiales y confirma la acción.
4. La página web elimina el especial en la api.

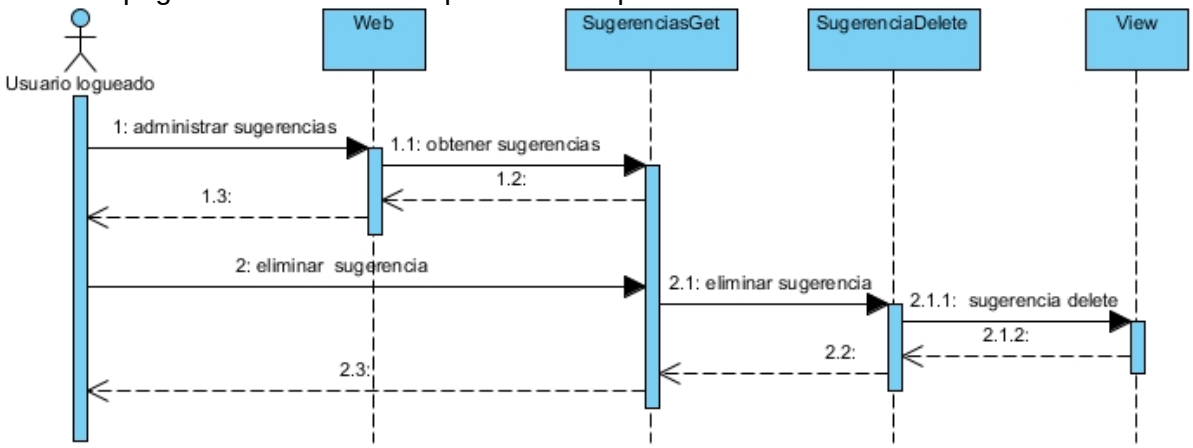


Figura 91: Diagrama de secuencias de crear especial.

5.6. Diagrama de base de datos.

Un diagrama de base de datos muestra las tablas pertenecientes a una base de datos y cómo se relacionan entre ellas.(Figura 92)

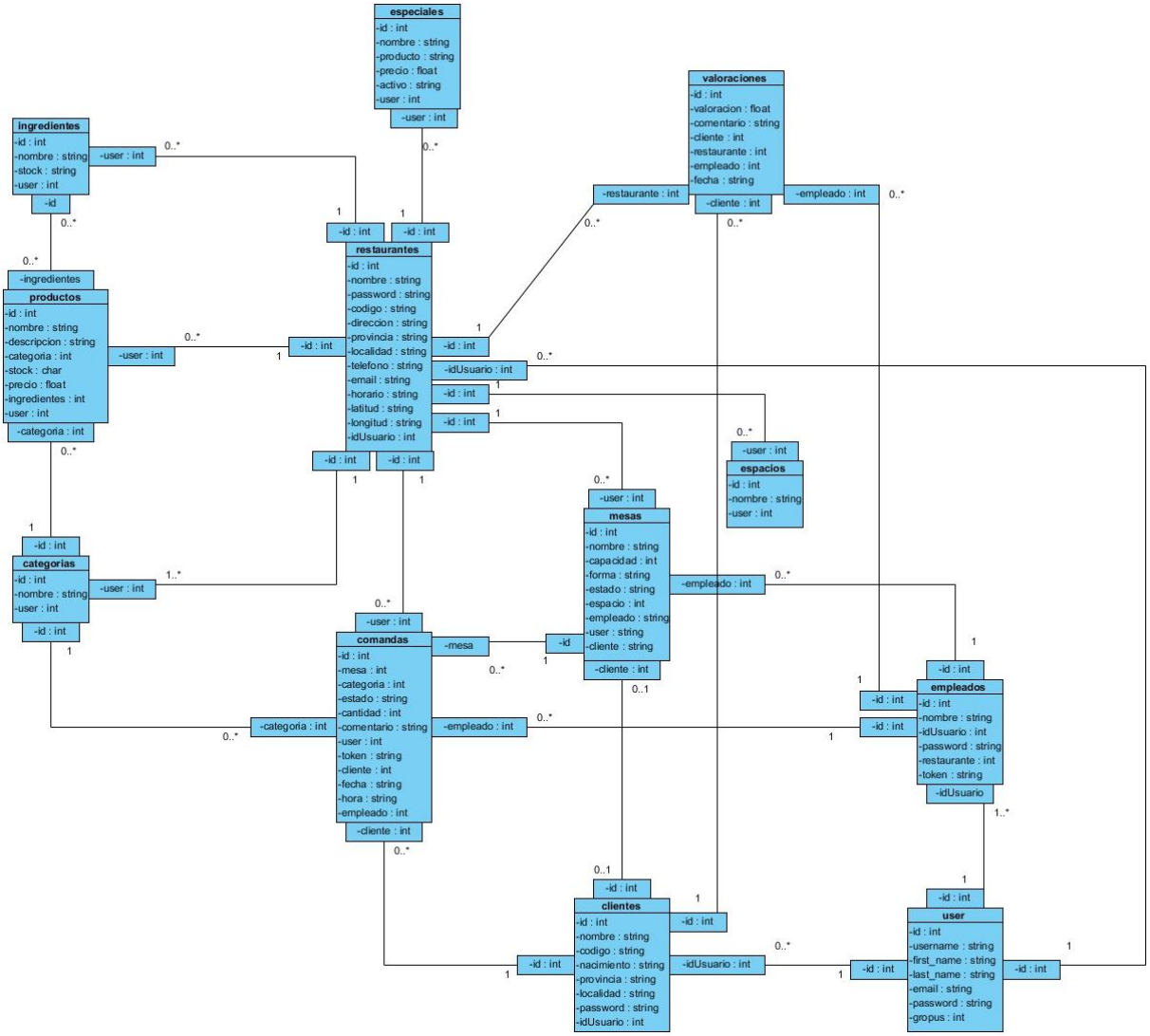


Figura 92: Diagrama de base de datos.

6. Implementación.

6.1. Entorno de desarrollo.

En el siguiente apartado se van a especificar las condiciones bajo las que se ha realizado el proyecto, tanto a nivel de hardware como a nivel software.

6.1.1. Entorno hardware.

Para la realización del proyecto se ha usado un equipo MSI modelo MS-7817 con Windows 10 de 64 bits con un procesador Intel Core i5 de 3.2HGz, una memoria RAM de 16.0 GB con tarjeta gráfica Intel HD Graphics 4600 y versión de DirectX 12.

6.1.2. Entorno software.

A nivel de software se ha utilizado Android Studio 2.2.2

El proyecto cuenta con la versión de sdk es la versión 25, siendo la mínima la versión 21 y la versión de google services es 3.0.0.

Se han utilizado, además, dos emuladores distintos para simular la interacción con el usuario, siendo ambos un terminal Nexus 5 de 4,95" tal como se muestra en la Figura 93, con 1GB de memoria RAM y 4GB de memoria ROM, una resolución de 1080x1920 píxeles, versión Marshmallow, con api 23 y Android 6.0 y 32 bits como se muestra en la figura 94.

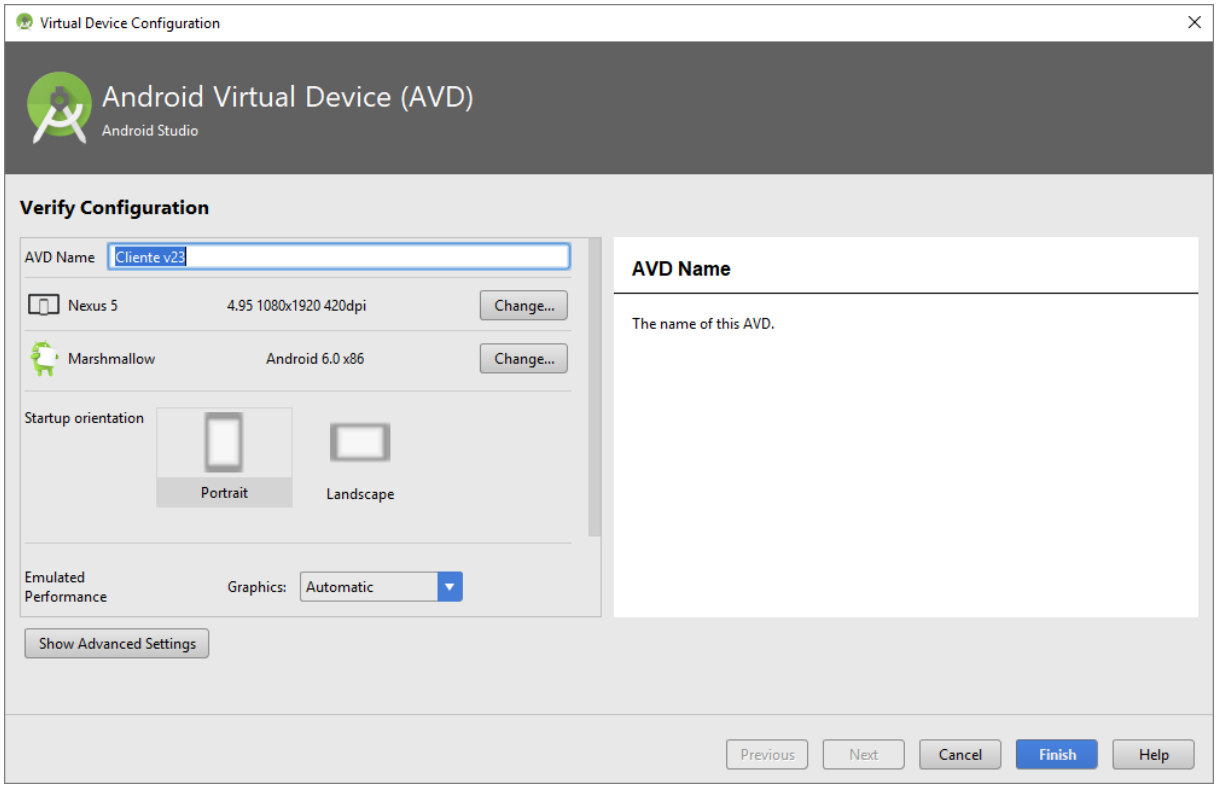


Figura 93: Servicio Virtual Android.

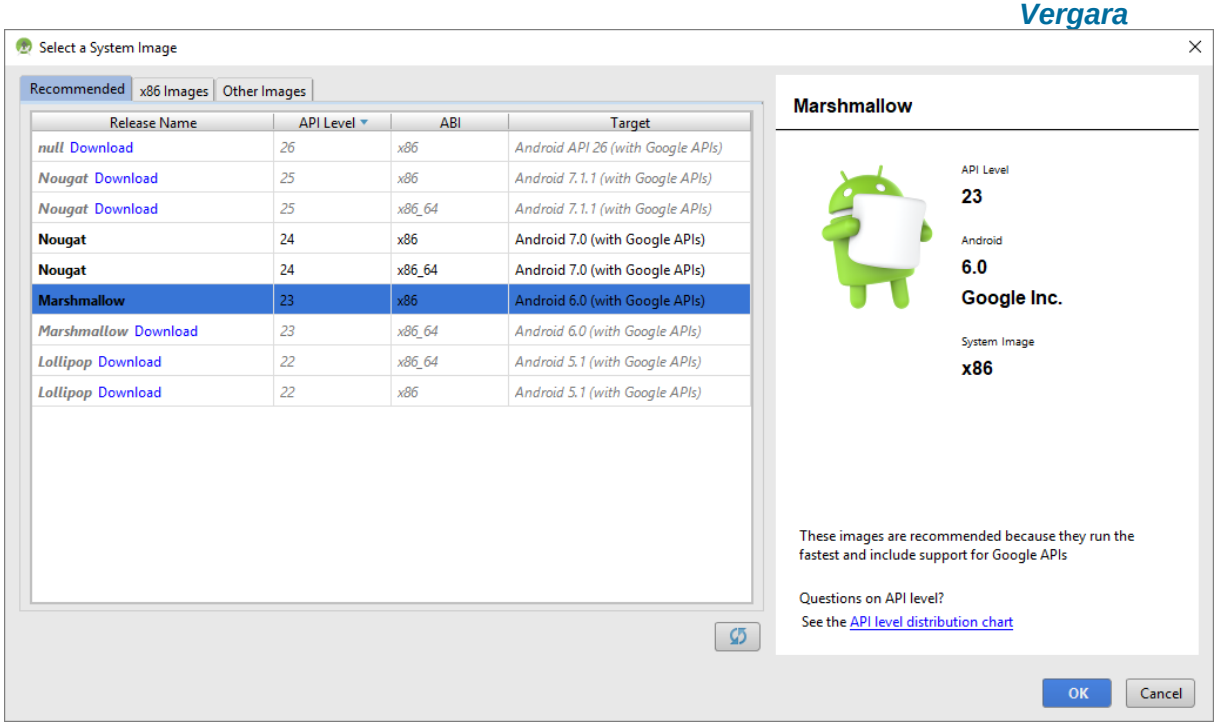


Figura 94: Especificaciones hardware.

6.2. Estructura de un proyecto Android.

Por defecto, un proyecto Android consta de.

- La carpeta manifests.
- La carpeta java donde se ubica el código fuente.
- La Carpeta layout donde se encuentran los layouts o vistas de cada pantalla de la aplicación.
- La carpeta menu, donde se encuentran los archivos que definen las plantillas de los menús de las activities.
- El resto de carpetas pertenecientes a res, que incluyen diversos recursos cómo imágenes, iconos, valores constantes, etc.
- Archivos grandle.

En la figura 95 se muestra una imagen de la estructura de un proyecto vacío.

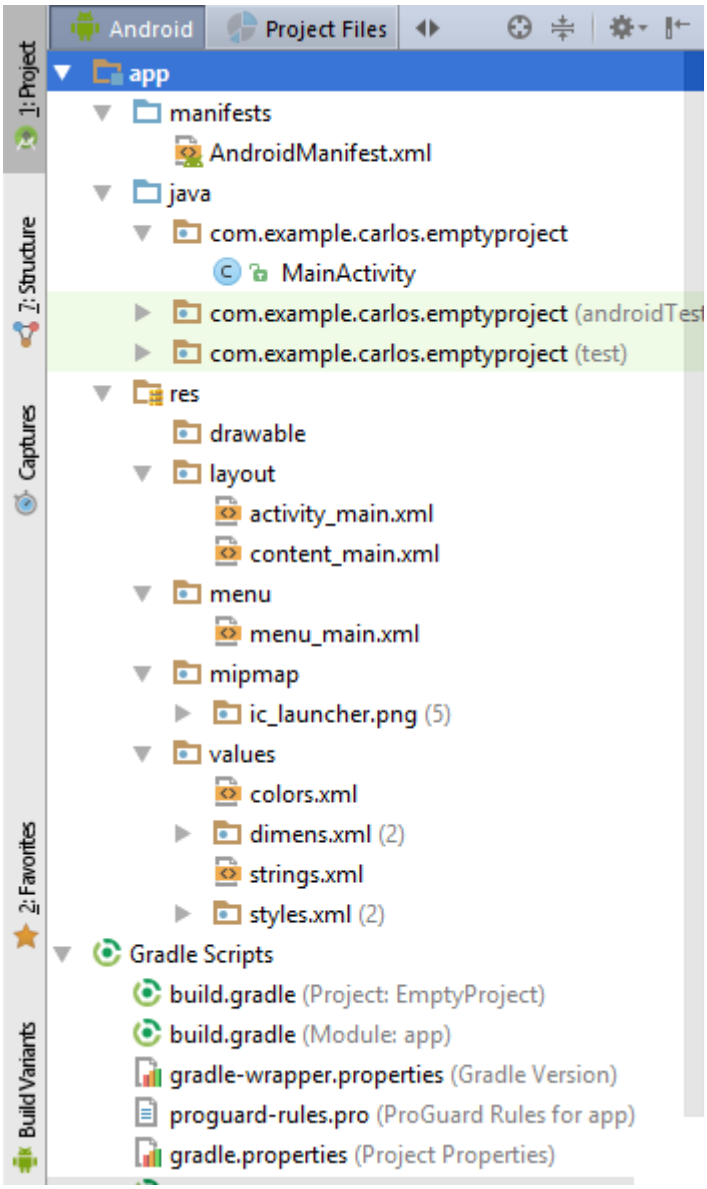


Figura 95: Estructura de un proyecto vacío.

6.2.1. manifests.

Dentro de la carpeta manifest podemos encontrar el archivo AndroidManifest.xml, en el cual se reflejan todas las configuraciones básicas del proyecto, permisos, y la configuración de todas las activities que contiene el proyecto.[49]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.carlos.comandual">
        <!-- Permiso para escribir en el dispositivo (para generar códigos QR)
-->
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <!-- Permisos para GPS y acceso a internet -->
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
   <uses-permission android:name="android.permission.INTERNET" />
        <!-- Permisos a la vibración del dispositivo y la cámara. -->
   <uses-permission android:name="android.permission.VIBRATE" />
   <uses-permission android:name="android.permission.WAKE_LOCK" />
   <uses-permission android:name="android.permission.CAMERA" />
   <uses-feature android:name="android.hardware.camera" />
   <uses-feature android:name="android.hardware.camera.autofocus" />
        <!-- Definición de las activities -->
   <application
       android:name=".MyApplication"
       android:allowBackup="true"
       android:icon="@drawable/ic_main"
       android:label="@string/app_name"
       android:supportsRtl="true"
       android:theme="@style/AppTheme">
       <activity
           android:name=".LoginActivity"
           android:label="@string/app_name"
           android:theme="@style/AppTheme.NoActionBar">
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
           <!-- Definición de los servicios de Firebase -->
       <service android:name=".MiFirebaseMessagingService">
           <intent-filter>
               <action android:name="com.google.firebase.MESSAGING_EVENT" />
           </intent-filter>
```

```
        </service>
        <service android:name=".MiFirebaseInstanceIdService">
            <intent-filter>
                <action android:name="com.google.firebase.INSTANCE_ID_EVENT" />
            </intent-filter>
        </service>
        <service
            android:name=".RegistrationService"
            android:exported="false" />
        <!-- Definicion de la API key del mapa de Google basado en APIs -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
        <activity
            android:name=".MapActivity"
            android:label="@string/title_activity_map"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity android:name=".SimpleScannerActivity" />
        [...]
</manifest>
```

6.2.2. Grandle.

Grandle es un conjunto de archivos para la compilación del proyecto de forma rápida y fácil.

En él se encuentra información como la versión de SDK y la versión mínima, la versión del proyecto y las distintas librerías y dependencias asociadas a dicho proyecto.[49]

```
apply plugin: 'com.android.application'
android {
   compileSdkVersion 25
   buildToolsVersion '25.0.0'
   defaultConfig {
       applicationId "com.example.carlos.comandual"
       minSdkVersion 21
       targetSdkVersion 25
       versionCode 1
       versionName "1.0"
   }
   buildTypes {
       release {
           minifyEnabled false
           proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
       }
   }
}
```

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile files('libs/droidText.0.4.jar')
    compile 'me.dm7.barcodescanner:zbar:1.9.1'
    compile 'me.dm7.barcodescanner:zbar:1.9.1'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'
    compile 'com.android.support:support-v4:25.3.1'
    compile 'com.android.support:recyclerview-v7:25.3.1'
    compile 'com.google.firebase:firebase-messaging:10.2.6'
    compile 'com.google.android.gms:play-services-gcm:10.2.6'
    compile 'com.mcxiaoke.volley:library:1.0.+'
    compile 'com.google.android.gms:play-services:10.2.6'
    compile 'com.google.zxing:core:3.2.1'
    compile 'com.journeyapps:zxing-android-embedded:3.4.0'
    compile 'com.google.zxing:android-integration:3.2.1'
    compile 'com.squareup.okhttp:okhttp:2.4.0'
    compile 'com.jjoe64:graphview:4.2.1'
    compile 'junit:junit:4.12'
    testCompile 'junit:junit:4.12'
}
apply plugin: 'com.google.gms.google-services'
```

6.2.3. Estructura de una Activity.

Las activities son cada una de las pantallas con las que el usuario puede interactuar en la aplicación.[50]

En nuestra aplicación, la activity más usada es la “Navigation Drawer Activity” por lo que se va a explicar la estructura de dicha plantilla.

La plantilla consta de los métodos:

- `protected void onCreate(Bundle savedInstanceState)`

Este método es el que inicializa la activity y muestra el layout.

- `public void onBackPressed()`

Con este método se controla la acción a realizar tras pulsar el botón de regreso.

- `public boolean onCreateOptionsMenu(Menu menu)`

En este método se especifica la plantilla de menú.

- `public boolean onOptionsItemSelected(MenuItem item)`

Este método se maneja el controlador de los componentes del menú.

- `public boolean onNavigationItemSelected(MenuItem item)`

Por último, este método genera la vista del menú desplegable controla los componentes del menú desplegable, aunque debido a que en la aplicación es el mismo menú para todas las activities, se maneja desde la clase DataBaseManager.

6.3. Código relevante.

6.3.1. DataBaseManager.

La clase DataBaseManager es la que se encarga de interactuar con la API, la base de datos local y la aplicación, y conectarlas entre sí.

DataBaseManager implementa la interfaz DataBase, que contiene todos los métodos necesarios para conectar la aplicación con la base de datos y otra interfaz, DataBaseApi, que contiene los métodos para conectar la aplicación con la API.

6.3.2. Volley Singleton

Volley es una librería desarrollada por Google para optimizar el envío de peticiones Http (Post,Get,Update,Delete) desde la aplicación Android hacia servidores externos, devolviendo un archivo JSON. [51]

A continuación se muestra un ejemplo del código necesario para realizar una petición.

```
//Creamos un objeto RequestQueue, lo inicializamos.
RequestQueue mRequestQueue;
mRequestQueue = VolleySingleton.getInstance().getmRequestQueue();
//Creamos un objeto StringRequest y lo inicializamos con los parámetros(tipo de consulta,
    url, listener)
StringRequest request;
request = new StringRequest(Request.Method.GET, url, new Response.Listener<String>() {
   @Override
   public void onResponse(String response) {

       try {
           //Inicializamos el JSON donde se van a guardar todos los datos.
           JSONArray array = new JSONArray(response);
           JSONObject object;
           //Recorremos el array
           for (int i = 0; i < array.length(); i++) {
               object = array.getJSONObject(i);
               //Obtenemos los elementos del objeto JSON
               String id = object.getString("id");
               String nombre = object.getString("nombre");
               String direccion = object.getString("direccion");
               String localidad = object.getString("etLocalidad");
               String provincia = object.getString("etProvincia");
               String latitud = object.getString("latitud");
               String longitud = object.getString("longitud");
           }
       } catch (Exception e) {
           String err = e.toString();
           Log.e("Err", err);
       }
   }
}, new Response.ErrorListener() {
```

```
    @Override
    public void onErrorResponse(VolleyError error) {
        String err = error.toString();
        Log.e("Err", err);

    }
});
//Añadimos el StringRequest a la cola de Request
mRequestQueue.add(request);
```

6.3.3. Encrypt

Esta clase se encarga de encriptar y desencriptar las contraseñas y datos sensibles de la base de datos.

Para ello, utiliza un sistema de encriptación con la función de derivación de claves PBKDF2 [52] con un número de iteraciones de 1000, que recibe como parámetros la sal [53] y el valor que se desea cifrar o descifrar.[54]

A continuación se muestra un ejemplo de uso tanto de encriptación como de desencriptación.

Para encriptación.

```
String encMsg = Encrypt.encrypt(SEED, msgToEncode);
```

Para desencriptación.

```
String decMsg = Encrypt.decrypt(SEED, msgToDecode);
```

6.3.4. Firebase

Firebase es una plataforma desarrollada por Google cuyo objetivo es facilitar la creación de apps, disponiendo de características como análisis de uso de la app, permite la autentificación mediante google o facebook, dispone de una base de datos en línea y un sistema de notificaciones, entre otras muchas funcionalidades.

En nuestro caso, vamos a usar Firebase para la gestión de notificaciones entre dispositivos.

En el siguiente código se registra se registra el dispositivo en Firebase y este dispositivo obtiene un token que será el mismo hasta que se desinstale la aplicación.

```
@Override
protected void onHandleIntent(Intent intent) {
    InstanceID miId = InstanceID.getInstance(this);

    //FirebaseMessaging.getInstance().subscribeToTopic("/topics/my_little_topic");

    String string1 = getString(R.string.gcm_defaultSenderId);
    String string2 = GoogleCloudMessaging.INSTANCE_ID_SCOPE;
    try{
        String registrationToken = miId.getToken(string1, string2);
        Log.d("Registration Token", registrationToken);

        GcmPubSub subscription = GcmPubSub.getInstance(this);
        String topic = "/topics/"+registrationToken;
        //subscription.subscribe(registrationToken, "/topics/empleados", null);
        subscription.subscribe(registrationToken, topic, null);
        //subscription.unsubscribe(registrationToken,"/topics/my_little_topic");

    }catch (Exception e){
        e.printStackTrace();
    }
}
```

Mediante el siguiente código, se recibe la notificación y se muestra en el dispositivo como una alerta.

```
public static final String TAG = "NOTICIAS";
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    super.onMessageReceived(remoteMessage);
    String from = remoteMessage.getFrom();
    Log.d(TAG,"Mensaje recibido de: " + from);
    if(remoteMessage.getNotification() != null){
        Log.d(TAG,"Notificacion: " + remoteMessage.getNotification().getBody());
        String title = remoteMessage.getNotification().getTitle();
        if(!title.equals("Nueva comanda")){
            mostrarNotificacion(remoteMessage);
```

```
        }
    }
    if(remoteMessage.getData().size()>0){
        //Log.d(TAG,"Data: " + remoteMessage.getData());
        String data;
        Map<String,String > map = remoteMessage.getData();
        Iterator<String> it = map.keySet().iterator();
        ArrayList<String> arrayList = new ArrayList<>();
        while (it.hasNext()){
            data = it.next();
            data += " = " + map.get(data);
            arrayList.add(data);
        }
        Collections.sort(arrayList,String.CASE_INSENSITIVE_ORDER);
        Log.d(TAG, " Data: " + arrayList.toString());
    }
}
private void mostrarNotificacion(RemoteMessage remoteMessage) {
    String title = remoteMessage.getNotification().getTitle();
    String body = remoteMessage.getNotification().getBody();
    Uri soundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    Intent intent = new Intent(this, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    PendingIntent pendingIntent =
     PendingIntent.getActivity(this,0,intent,PendingIntent.FLAG_ONE_SHOT);
    NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_add_alert_black_24dp)
            .setContentTitle(title)
            .setContentText(body)
            .setAutoCancel(true)
            .setSound(soundUri)
            .setContentIntent(pendingIntent);
    NotificationManager notificationManager = (NotificationManager)
     getSystemService(Context.NOTIFICATION_SERVICE);
    notificationManager.notify(0,notificationBuilder.build());
}
```

Una notificación nueva se realiza mediante el siguiente método.

```
static boolean enviarNotificacion(String title, String body, String to) {
    RequestQueue mRequestQueue;
    mRequestQueue = VolleySingleton.getInstance().getmRequestQueue();
    String urlCMS = "https://fcm.googleapis.com/fcm/send";
    final String jsonString =
            "{" +
                    "'to':'/topics/" + to + "'," +
                    "'notification':{" +
```

```
                "'title':'" + title + "'," +
                "'body':'" + body + "'" + "}" + "}";
        try {
            JSONObject jsonObject = new JSONObject(jsonString);
            JsonObjectRequest jsonObjReq = new JsonObjectRequest(Request.Method.POST,
                    urlCMS, jsonObject,
                    new Response.Listener<JSONObject>() {
                        @Override
                        public void onResponse(JSONObject response) {}
                    }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    VolleyLog.d("Response", "Error: " + error.getMessage());
                }
            }) {
                @Override
                public Map<String, String> getHeaders() throws AuthFailureError {
                    Map<String, String> params = new HashMap<>();
                    params.put("Content-Type", "application/json");
                    params.put("Authorization", "key=AIzaSyDAE9MwmYI9VjGjTqd6qLzOadga0tqvurs");
                    return params;
                }
            };
            mRequestQueue.add(jsonObjReq);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }
```

7. Planificación.

7.1. Introducción.

En el siguiente apartado se va llevar a cabo una comparación de la planificación diseñada a priori y del tiempo empleado realmente para observar cuánto se ha desviado de las previsiones iniciales.

Como se ha mencionado anteriormente, la metodología empleada para este proyecto ha sido Kanban, por lo que la planificación se ajusta en la medida de lo posible a la filosofía de dicha metodología.

7.2. Software utilizado.

El programa empleado para la planificación ha sido ProjectLibre, debido a que es un software de libre uso con la posibilidad de crear informes, diagramas de gantt y todo lo necesario para llevar a cabo la planificación, por lo que no se ha estudiado la posibilidad de usar otros softwares ya sean gratuitos o con licencia.[51]

7.3. Planificación.

Actividad	Trabajo estimado (H)	Trabajo real (H)	Duración estimada (D)	Duración real (D)	Inicio previsto	Inicio real	Fin previsto	Fin real
Proyecto	**300**	**350**	**60**	**70**	19/06/2017	6/02/2017	**3/05/2017**	19/06/2017
Documentación	**50**	**70**	**10**	**14**	1/03/2017	6/02/2017	**17/02/2017**	1/03/2017
Manuales de usuario	10	15	2	3	9/02/2017	6/02/2017	8/02/2017	9/02/2017
Diagramas	20	25	4	5	17/02/2017	9/02/2017	13/02/2017	17/02/2017
Investigación	15	25	3	5	28/02/2017	20/02/2017	16/02/2017	28/02/2017
Maquetación	5	5	1	1	1/03/2017	28/02/2017	17/02/2017	1/03/2017
Api DJango	**15**	**30**	**3**	**6**	7/03/2017	24/02/2017	**21/02/2017**	7/03/2017
Definición de la base de datos	5	5	1	1	27/02/2017	24/02/2017	17/02/2017	27/02/2017
Implementación de la base de datos	7	20	1,4	4	6/03/2017	27/02/2017	20/02/2017	6/03/2017
Test de la base de datos	3	5	0,6	1	7/03/2017	6/03/2017	21/02/2017	7/03/2017
Aplicación móvil	**185**	**160**	**37**	**32**	8/05/2017	6/03/2017	**19/04/2017**	8/05/2017
Base de datos	10	10	2	2	8/03/2017	6/03/2017	22/02/2017	8/03/2017
Registro, login y logout	10	15	2	3	14/03/2017	8/03/2017	24/02/2017	14/03/2017
Gestionar mesas	**42**	**25**	**8,4**	**5**	22/03/2017	14/03/2017	**9/03/2017**	22/03/2017
CRUD mesas	10	10	2	2	16/03/2017	14/03/2017	28/02/2017	16/03/2017
Limpiar mesas	5	2	1	0,4	17/03/2017	16/03/2017	1/03/2017	17/03/2017

Asignar mesas	7	3	1,4	0,6	17/03/2017	17/03/2017	3/03/2017	17/03/2017
Generar QR	15	7	3	1,4	21/03/2017	20/03/2017	8/03/2017	21/03/2017
Leer QR	5	3	1	0,6	22/03/2017	21/03/2017	9/03/2017	22/03/2017
CRUD espacios	15	7	3	1,4	24/03/2017	22/03/2017	14/03/2017	24/03/2017
CRUD productos	10	7	2	1,4	27/03/2017	24/03/2017	16/03/2017	27/03/2017
CRUD ingredientes	10	7	2	1,4	29/03/2017	28/03/2017	20/03/2017	29/03/2017
CRUD categorías	15	7	3	1,4	31/03/2017	29/03/2017	23/03/2017	31/03/2017
CRUD especiales	10	10	2	2	4/04/2017	31/03/2017	27/03/2017	4/04/2017
Gestionar comandas	**20**	**33**	**4**	**6,6**	24/04/2017	5/04/2017	**31/03/2017**	24/04/2017
CRUD comandas	10	18	2	3,6	18/04/2017	5/04/2017	29/03/2017	18/04/2017
Generar cuenta	10	15	2	3	24/04/2017	18/04/2017	31/03/2017	24/04/2017
CRUD empleados	10	7	2	1,4	25/04/2017	24/04/2017	4/04/2017	25/04/2017
Gráfica	5	5	1	1	27/04/2017	26/04/2017	5/04/2017	27/04/2017
Demo	3	2	0,6	0,4	27/04/2017	27/04/2017	5/04/2017	27/04/2017
Notificaciones	10	20	2	4	5/05/2017	27/04/2017	7/04/2017	5/05/2017
Test	15	5	3	1	8/05/2017	5/05/2017	19/04/2017	8/05/2017
Aplicación web	**50**	**90**	**10**	**18**	19/06/2017	9/05/2017	**3/05/2017**	19/06/2017
Subida a la red	7	15	1,4	3	12/05/2017	9/05/2017	21/04/2017	12/05/2017
Registro, login y logout	5	15	1	3	18/05/2017	12/05/2017	24/04/2017	18/05/2017
CRUD mesas	5	13	1	2,6	23/05/2017	18/05/2017	25/04/2017	23/05/2017
CRUD espacios	5	10	1	2	26/05/2017	23/05/2017	26/04/2017	26/05/2017
CRUD productos	5	7	1	1,4	29/05/2017	26/05/2017	27/04/2017	29/05/2017
CRUD ingredientes	5	7	1	1,4	31/05/2017	30/05/2017	28/04/2017	31/05/2017
CRUD categorías	5	5	1	1	12/06/2017	31/05/2017	1/05/2017	12/06/2017
CRUD empleados	5	5	1	1	14/06/2017	13/06/2017	2/05/2017	14/06/2017
CRUD especiales	5	3	1	0,6	14/06/2017	14/06/2017	3/05/2017	14/06/2017
Test	3	10	0,6	2	19/06/2017	15/06/2017	3/05/2017	19/06/2017

7.4. Conclusiones.

Se puede observar en la tabla comparativa que en general, se han empleado 50 horas más de las previstas.
Esto es debido a que a que:

- Se han empleado 20 horas más a la creación de la documentación que en un principio se estimaba en 50.
- Se ha empleado el doble de tiempo (15 horas) en definir, implementar y probar la api que da soporte a todo el sistema (móvil y web)
- Se han empleado 40 horas más en la creación de la página web, que se estimaba en un principio en 50 horas.
- A pesar de las 85 horas anteriormente mencionadas, la aplicación móvil se ha completado 25 horas antes de lo previsto.

Estas 50 horas añadidas al tiempo estimado han repercutido en 10 días en la duración real, haciendo los 60 días estimados en 70 días reales empleados (se presupone que trabajados y de provecho), siendo un día laboral de 5 horas.

Respecto a las fechas, se puede observar un retraso considerable en la fecha estimada de fin del proyecto y la fecha real, siendo la desviación de un mes medio, debido a festivos no tenidos en cuenta, asuntos propios y la eficiencia real de los días trabajados, debido a que en una estimación optimista se habían definido 5 horas de trabajo diarias cuando realmente de esas 5 horas empleadas, sólo 4 han sido efectivas.

8. Resultados.

Una vez acabado el proyecto, se ha sometido la aplicación a diversos test de pruebas y de rendimiento gracias a la herramienta Test Lab de Firebase.

8.1. Pruebas.

En la siguiente tabla se especifican los terminales que se han han utilizado y los resultados de los test.

Nombre del terminal	API	Tipo de terminal	Duración	Incidencias
Nexus 5	23	Físico	45 s	
HTC One (M8)	19	Físico		La aplicación no admite la versión de SO especificada
Xperia Z3	21	Físico	39 s	
Galaxy S7	23	Físico	3m 38s	
LG G3	19	Físico		La aplicación no admite la versión de SO especificada
Nexus 6P	26	Virtual	5m 16s	
Nexus 5	23	Virtual	47s	
Nexus 10	22	Virtual	5m 14s	
Nexus 6	21	Virtual	5m 11s	
Nexus 9	23	Virtual	1m 6s	

8.2. Rendimiento.

A continuación, se muestra el rendimiento de tres terminales físicos en relación al consumo de CPU, memoria y acceso a internet.

8.2.1. Nexus 5, nivel de API 23.

Se ha realizado un test de rendimiento con un terminal físico Nexus 5 con nivel de API 23, cuyos resultados pueden observarse en la figura 96.

Figura 96: Rendimiento de un nexus 5, con nivel de API 23.

En la figura se observa que en el momento de iniciar la aplicación, llega a consumir un 30% del total de la CPU, valor que baja considerablemente durante el resto de la ejecución.

Por otro lado, se observa un consumo de memoria de 120 KB y se diferencian tres peticiones de acceso a internet, la más alta con 50 KB y 30 y menos de 15 KB las otras dos.

8.2.2. Z3, nivel de API 21.

Se ha realizado un test de rendimiento con un terminal físico Z3 con nivel de API 21, cuyos resultados pueden observarse en la figura 97.

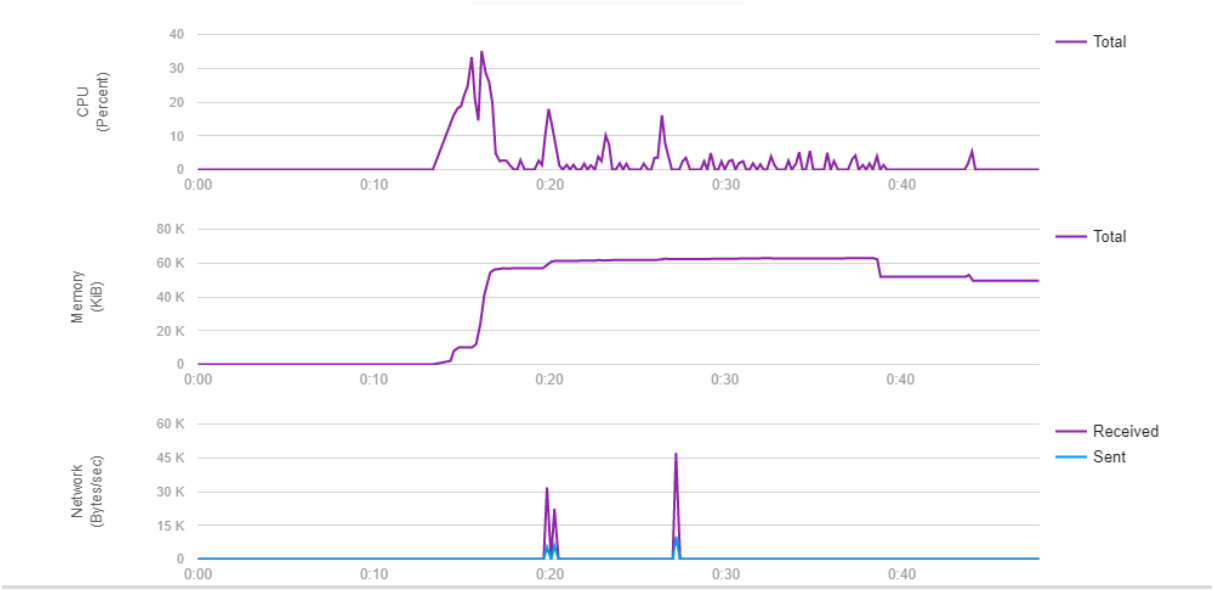


Figura 97. Rendimiento de un Z3, con nivel de API 21.

Los valores obtenidos son similares a los descritos anteriormente con el Nexus 5.

En la figura se observa que en el momento de iniciar la aplicación, llega a consumir más de un 30% del total de la CPU, valor que baja considerablemente durante el resto de la ejecución, teniendo algunos picos del 20% de consumo.

Respecto a la memoria, se registra un consumo de 50 KB, menos de la mitad del resultado obtenido con el Nexus 5.

Se han realizado tress peticiones de acceso a internet, la más alta con 45 KB y 30 y menos de 15 KB las otras dos, valores similares a los datos obtenidos con el Nexus 5.

8.2.3. Galaxy S7, nivel de API 23

En último lugar, se ha realizado un test de rendimiento con terminal físico Galaxy S7 con nivel de API 23, cuyos resultados pueden observarse en la figura 98.

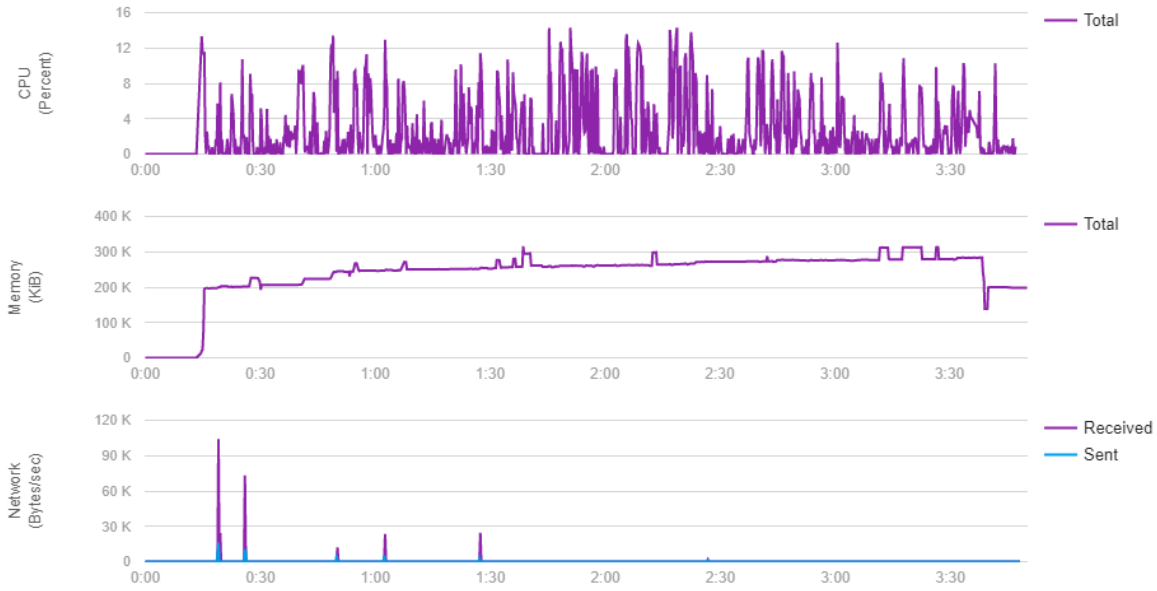


Figura 98. Rendimiento de un Z3, con nivel de API 21.

En este caso, se ha hecho un examen exhaustivo de la aplicación, en el que se ha obtenido que el pico máximo de consumo de CPU es del 14% en momentos en los que la aplicación hace la carga de datos.

Por otro lado, se observa que el consumo de memoria máximo es de 312KB, coincidiendo los picos de consumo de memoria con las pantallas emergentes.

En último lugar, se observa en la gráfica que los picos máximos de consumo de datos se deben a la carga inicial de datos, ya que se guardan en local para evitar el consumo constante de datos.

8.3. Conclusiones.

Si bien los resultados dependen de las condiciones del dispositivo en el que se ejecutan y de su nivel de API, se puede decir que las pruebas han resultado ser un buen instrumento para contemplar cómo se comporta la aplicación bajo condiciones reales de uso, aunque aún se puede trabajar más en disminuir, por ejemplo, el consumo de memoria, el cual es elevado en todos los test realizados.

9. Conclusiones y trabajos futuros.

En conclusión, se puede decir que se ha construido una aplicación más funcional que la mayoría de apps analizadas, más robusta y resistente a fallos, que a pesar de tener una aplicación web que la complementa, funciona independientemente y no necesita módulos ni aplicaciones para su correcto funcionamiento, intuitiva además de poseer un manual de usuario y atractiva a la vista.

Por otra parte, si bien se ha intentado hacer una aplicación completa que cubra las necesidades de restaurantes, empleados y clientes, debido a la limitación de tiempo y recursos se han dejado una serie de funcionalidades sin desarrollar, algunas de las cuales son las siguientes.

- Crear un chat interno por el cual los empleados de un restaurante puedan comunicarse mediante mensajería instantánea.
- Integrar en la aplicación redes sociales para que los usuarios puedan actualizar sus estados de forma cómoda.
- Permitir a los usuarios de la aplicación registrarse y loguearse mediante redes sociales como Facebook y Google+ para agilizar y facilitar el proceso.
- Habilitar la aplicación para que pueda ser visualizada en distintos idiomas.
- Permitir al cliente pagar mediante Paypal sin necesidad de solicitar la atención de un camarero, ahorrando tiempo y recursos.
- Crear un sistema de gestión de reservas para que clientes puedan reservar mesas en los locales.
- Realizar un estudio profundizado de las ventas y valoraciones del local para mejorar la toma de decisiones.
- Monetizar la aplicación, capando las funciones del restaurante con objetivo de que sólo se pueda tener acceso si el dispositivo cuenta con una clave software que de acceso a toda la funcionalidad, y mediante la inserción de anuncios en la parte de los clientes.Otra forma de monetizar la aplicación, sería mediante el alquiler de la aplicación a los locales por un módico precio.
- Si bien la aplicación es segura, siempre se puede mejorar la seguridad contra ataques de terceros.

10. Referencias.

Metodologías.

[1] Modelos y metodologías para el desarrollo de software (4 de sept de 2017)
http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm

Modelo en cascada.

[2] Modelo en cascada (4 de sept de 2017)
https://www.freelancer.es/community/articles/proceso-del-desarrollo-software
http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm
http://fasesmodelocascada.blogspot.com.es/

[3] Ventajas e inconvenientes del modelo en cascada (4 de sept de 2017)
http://metodologiaencascada.blogspot.com.es/

Modelo incremental.

[4]. Modelo incremental (4 de sept de 2017)
https://www.freelancer.es/community/articles/proceso-del-desarrollo-software

[5] Ventajas e inconvenientes del modelo incremental (4 de sept de 2017) http://isw-udistrital.blogspot.com.es/2012/09/ingenieria-de-software-i.html

Modelo en espiral.

[6] Modelo en espiral (4 de sept de 2017) http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm

[7] Ventajas del modelo en espiral (4 de sept de 2017)
https://sites.google.com/site/proyectoadpmodelosdedesarrollo/home/modelos-de-desarrollo/modelo-espiral-ventajas-y-desventajas

[8] Desventajas del modelo en espiral (4 de sept de 2017)
http://modeloespiral.blogspot.com.es/2009/08/desventajas.html

Metodologías ágiles.

[9] Metodologías ágiles (4 de sept de 2017) http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm

[10] Manifiesto ágil (4 de sept de 2017) http://blog.conectart.com/metodologias-agiles/#MANIFIESTO_AGIL

Scrum

[11] Scrum (4 de sept de 2017) https://proyectosagiles.org/que-es-scrum/

[12] Ventajas y desventajas de scrum (4 de sept de 2017)
https://es.quora.com/Cu%C3%A1les-son-las-ventajas-y-desventajas-de-agile-scrum

Kanban

[13] Kanban (4 de sept de 2017) http://blog.conectart.com/metodologias-agiles/#kanban

[14] Principios del método Kanban (4 de sept de 2017) https://www.coriaweb.hosting/metodo-kanban-cuales-principios-basicos/

[15] Ventajas y desventajas de Kanban (4 de sept de 2017) http://kanbanuji.blogspot.com.es/2013/04/ventajas-y-desventajas-del-uso-de-kanban.html

Xtreme Programming

[16] Extreme programming (4 de sept de 2017) http://blog.conectart.com/metodologias-agiles/#1_Scrum

[17] Ventajas y desventajas del XP (4 de sept de 2017) https://iswugaps2extremeprogramming.wordpress.com/2015/09/14/ventajas-y-desventajas/

Nativas VS Híbridas VS Web

[18] Desarrollo de apps hibridas vs apps nativas (4 de sept de 2017 enlace caído) http://www.bitbotic.com/desarrollo-de-apps-hibridas-vs-apps-nativas-ventajas-e-inconvenientes-de-cada-tecnica/

Android VS IOS

Accesibilidad

[19] Accesibilidad (4 de sept de 2017) https://www.sozpic.com/desarrollo-ios-vs-android-por-donde-empiezo/

Curva de aprendizaje.

[20] Curva de aprendizaje (4 de sept de 2017) https://www.sozpic.com/desarrollo-ios-vs-android-por-donde-empiezo/

Gestión de la memoria.

[21] ARC (Automatic Reference Counting) (4 de sept de 2017) http://www.migueldiazrubio.com/introduccion-a-arc/

[22] Garbage Collector (4 de sept de 2017) https://ayddup.wordpress.com/2011/08/08/la-memoria-en-java-garbage-collector-y-el-metodo-finalize/

[23] Gestión de la memoria. (4 de sept de 2017) https://www.sozpic.com/desarrollo-ios-vs-android-por-donde-empiezo/

http://www.elandroidelibre.com/2014/10/diferencias-entre-desarrollar-aplicaciones-para-ios-o-android.html

Documentación.

[24] Documentación (4 de sept de 2017) https://www.sozpic.com/desarrollo-ios-vs-android-por-donde-empiezo/

UIViewController vs Activity.

[25] UIViewController vs Activity. http://www.elandroidelibre.com/2014/10/diferencias-entre-desarrollar-aplicaciones-para-ios-o-android.html

Emulador/Simulador.

[26] Emulador vs Simulador (4 de sept de 2017) https://www.sozpic.com/desarrollo-ios-vs-android-por-donde-empiezo/ http://www.elandroidelibre.com/2014/10/diferencias-entre-desarrollar-aplicaciones-para-ios-o-android.html

Licencias.

[27] Licencias (4 de sept de 2017) http://blog.ticsandroll.es/android-vs-ios-para-que-sistema-operativo-desarrollo/

IDE Android.

Basic4Android.

[28] Basic 4 Android (4 de sept de 2017) https://www.unocero.com/2014/06/04/basic-4-android-para-programar-apps-facilmente/
https://prezi.com/wigriupwjajd/basic-4-android/
http://www.forosdeelectronica.com/f26/hablando-basic4android-109033/

App inventor.

[29] App inventor (4 de sept de 2017) https://betabeers.com/forum/desarrollar-apps-android-eclipse-o-android-studio-1284/

Eclipse.

[30] Eclipse (4 de sept de 2017)
https://es.wikipedia.org/wiki/Eclipse_(software)

Android Studio.

[31] Android Studio (4 de sept de 2017) https://betabeers.com/forum/desarrollar-apps-android-eclipse-o-android-studio-1284/

Desktop App VS Web App.

[32] Desktop App VS Web App (4 de sept de 2017) http://www.buyto.es/general-diseno-web/diferencias-entre-aplicaciones-web-y-aplicaciones-desktop
http://byspel.com/aplicaciones-de-escritorio-vs-aplicaciones-web/

Lenguaje de la WebApp.

PHP.

[33] Ventajas y desventajas de PHP (4 de sept de 2017)
http://www.registrodominiosinternet.es/2013/08/lenguajes-programacion-web-ventajas.html
https://www.sigmareef.com/desarrollo-web/programacion-php-vs-nodejs/

NodeJS.

[34] Ventajas y desventajas de NodeJS (4 de sept de 2017)
https://www.sigmareef.com/desarrollo-web/programacion-php-vs-nodejs/

Ruby.

[35] Ventajas y desventajas de Ruby (4 de sept de 2017)
http://www.registrodominiosinternet.es/2013/08/lenguajes-programacion-web-ventajas.html
https://es.wikipedia.org/wiki/Ruby
http://www.lawebdelprogramador.com/foros/Ruby/909799-Ventajas-de-Ruby.html

Python.

[36] Ventajas y desventajas de Python (4 de sept de 2017)
https://es.quora.com/Qu%C3%A9-ventajas-nos-ofrece-Python-respecto-a-otros-lenguajes-de-programaci%C3%B3n

Arquitectura Rest VS Soap

Rest vs Soap

[37] Rest (4 de sept de 2017)
http://qode.pro/blog/web-services-rest-vs-soap/

[38] Ventajas y desventajas de Rest (4 de sept de 2017)
http://qbit.com.mx/blog/2012/02/14/rest-vs-soap/

[39] Soap (4 de sept de 2017) (enlace caído)
http://movilforum.com/apis-seguimos-con-el-rest-vs-soap/ http://qode.pro/blog/web-services-rest-vs-soap/

[40] Ventajas y desventajas de Soap (4 de sept de 2017)
http://qbit.com.mx/blog/2012/02/14/rest-vs-soap/

Api y WebApp

Django Rest Framework

[41] Django Rest Framework (4 de sept de 2017)
http://www.django-rest-framework.org/

Django

[42] Django (4 de sept de 2017)
https://www.djangoproject.com/

Diagramas de casos de uso.

[47] Diagramas de casos de uso. (4 de sept de 2017)
https://www.infor.uva.es/~chernan/Ingenieria/Teoria/Tema3D.pdf

Diagrama de secuencias.

[48] Definición diagrma de secuencias (4 de sept de 2017)
https://msdn.microsoft.com/es-es/library/dd409377.aspx

Implementación.

Manifiests y Grandle

[49] Manifiests (4 de sept de 2017) https://androidstudiofaqs.com/conceptos/cual-es-la-estructura-de-un-proyecto-en-android-studio

Estructura de una activity

[50] Definición de activity (4 de sept de 2017)
https://androidstudiofaqs.com/conceptos/que-es-un-activity-en-android

Código relevante.

[51] Volley Singleton (4 de sept de 2017)
http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/

[52] PBKDF2 (4 de sept de 2017)
https://en.wikipedia.org/wiki/PBKDF2

[53] Sal (4 de sept de 2017)
https://es.wikipedia.org/wiki/Sal_(criptograf%C3%ADa)

[54] Encrypt (4 de sept de 2017)
http://www.ssaurel.com/blog/how-to-use-cryptography-in-android-applications-2/

Planificación.

[52] ProjectLibre (4 de sept de 2017)
https://es.wikipedia.org/wiki/ProjectLibre
Tutorial sobre ProjectLibre
https://www.youtube.com/watch?v=UOTwDczsSAw&list=PLtXusbeR8BLSLSSMmF6aXXusRhVbOJaPT

Bibliografía.

Análisis de requisitos.

[43] Definición de requisito.
Ingeniería del Software. Un enfoque desde la guía SWEBOK. ->Pág. 116

[44] Requisitos funcionales.
Ingeniería del Software. Un enfoque desde la guía SWEBOK. ->Pág. 123

[45] Requisitos no funcionales.
Ingeniería del Software. Un enfoque desde la guía SWEBOK. ->Pág. 124

[46] Actores.
Ingeniería del Software. Un enfoque desde la guía SWEBOK. ->Pág. 118

Figuras.

11. Anexos

Figuras.

Aplicación móvil para locales.

1. Registrar restaurante.

Para registrarnos como local, pinchamos en el botón “Registrate” (Figura 1.a) y a continuación seleccionaremos “Registrarse como Empresa” (Figura 1.b).

Figura 1.a: Pantalla de login

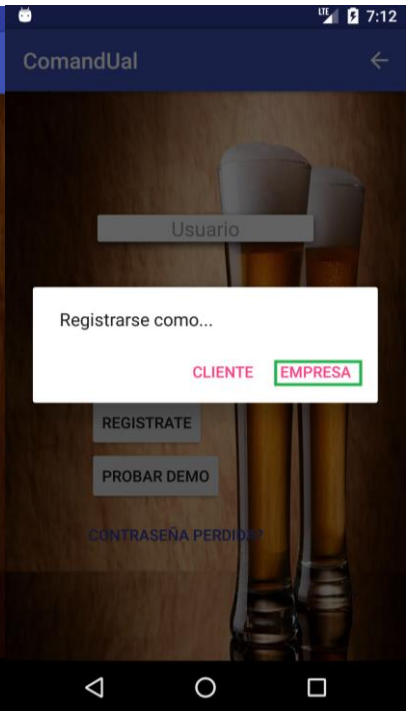


Figura 1.b: Registrarse como empresa

Figura 1.c: Pantalla registro.

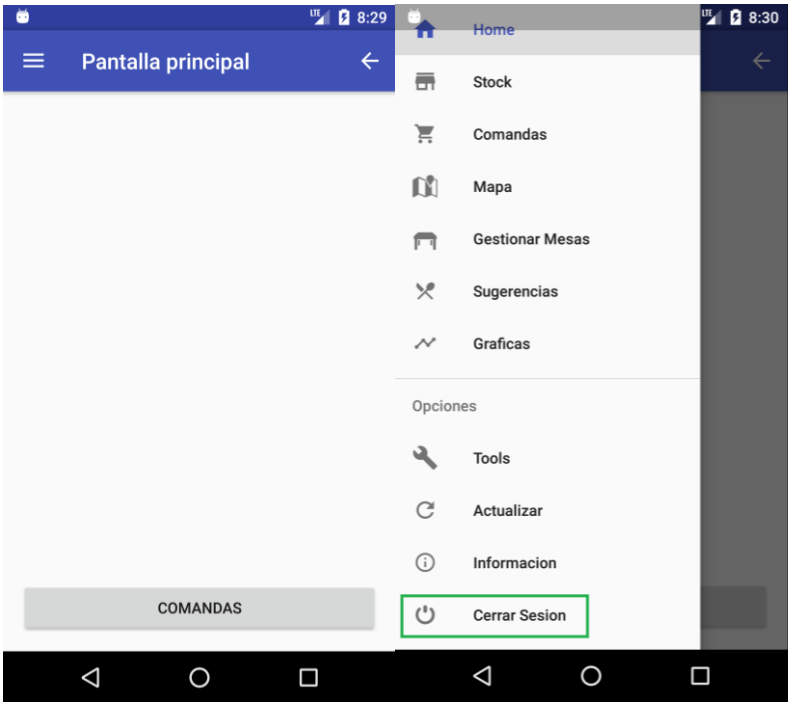
Figura 1.d: Crear restaurante.

2. Login y Logout.

Para loguearse en la aplicación basta con poner el nombre de usuario y contraseña y pinchar sobre “Login” (Figura 2).

Figura 2: Pantalla login

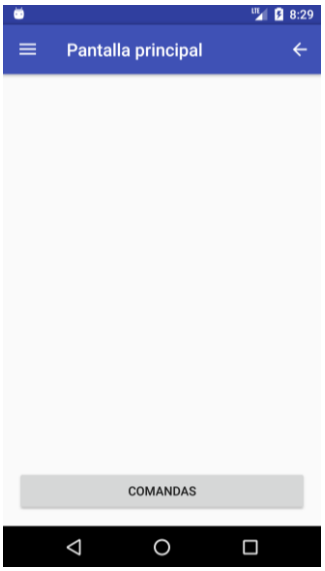
Para cerrar sesión, desde cualquier pantalla de la aplicación abrimos el menú desplegable pulsando el botón de la parte superior izquierda y pulsamos sobre “Cerrar sesión”.



3. Pantalla principal.

Al iniciar sesión se muestra el estado de las mesas en cada uno de los espacios.

Puesto que no hay mesas y espacios creados, se nos mostrará en blanco.(Figura 3)



Para poder explicar mejor las funciones que tiene la aplicación, seguiremos el ejemplo con un restaurante ya creado y actualizado.

Por defecto, la primera pantalla que se nos mostrará será un listado de las mesas del local con un código de colores que depende del estado de la mesa (Figura 3.c):

- Rojo: Pendiente de atención.
- Negro: Ocupada.
- Verde: Libre.

Figura 3.c: Pantalla principal

Además de visualizar mesas y espacios se pueden observar varias acciones, como cambiar de espacio, crear una nueva mesa o ir a detalles.

3.1. Creación de espacios y mesas.

Para crear una nueva mesa o un nuevo espacio, pulsaremos sobre el botón “+” (Figura 3.a) situado detrás de la última mesa, A continuación, para crear un nuevo espacio pulsaremos sobre la lista desplegable de espacios (Figura 3.b) y pulsaremos “Nuevo espacio” (Figura 3.c). A continuación, escribiremos el nombre y seleccionaremos “Crear espacio”. (Figura 3.d)

Figura 3.a: Añadir mesa

Figura 3.b: Pantalla crear mesa

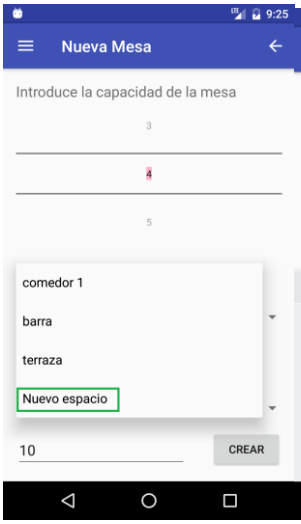


Figura 3.c: Nuevo espacio

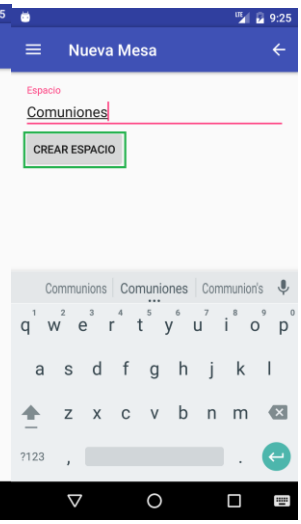


Figura 3.d: Crear espacio

Para crear una mesa nueva se rellenan los datos de la mesa y pulsamos sobre crear (Figura 3.d).

A continuación, ya podemos ver el nuevo espacio y la nueva mesa (FIgura 3.e).

Figura 3.d: Crear mesa.

Figura 3.e: Ver espacio.

3.2. Detalles de mesa.

Si mantenemos pulsada una mesa durante unos segundos, tendremos varias opciones disponibles dependiendo del estado de la mesa.

Si la mesa está ocupada, tendremos la opción de limpiarla para dejarla en estado de libre (Figura 3.f).

Figura 3.f: Limpiar mesa

Si mantenemos pulsado sobre una mesa limpia o libre, tendremos la opción de generar un código qr de esa mesa para que los clientes la puedan escanear como se puede observar en las imágenes Figura 3.g y Figura 3.h.

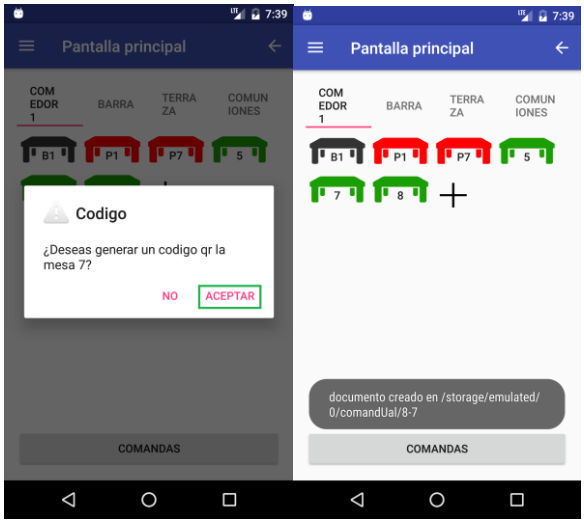


Figura 3.g Generar código Figura 3.h: Código generado

Si pulsamos sobre una mesa, accederemos a los detalles de esta (Figura 3.i) . Desde aquí, podremos realizar nuevas comandas, ver el estado de las que ya existen o generar la cuenta.

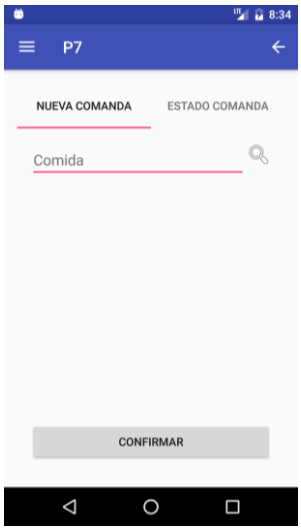


Figura 3.i detalles de mesa

Una vez aquí, podemos generar una nueva comanda de dos formas distintas.

La primera, mediante la carta de productos del local y la segunda, escribiendo el nombre del producto.

3.3. Comanda mediante buscador.

Para realizar el pedido mediante el buscador, sólo deberemos introducir el nombre del producto en él, y seleccionar el producto deseado (Figura 3.j).

Como ejemplo, vamos a seleccionar 2 platos de arroz a la cubana y 2 platos conbinados, con 3 fantas de naranja y una fanta de Limón.

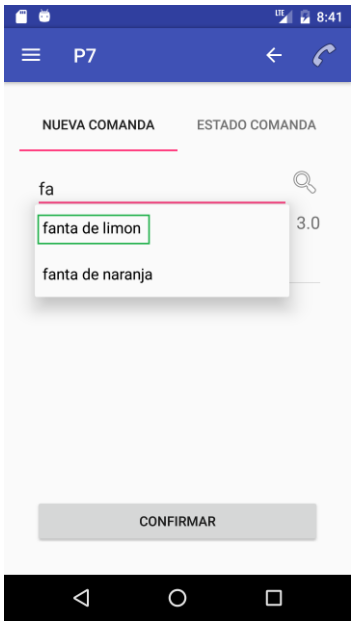


Figura 3.j: Buscador

Como podemos comprobar, el buscador muestra todos los productos que contengan las palabras introducidas.

3.4. Comanda mediante carta.

Para realizar un pedido mediante la carta, pulsaremos sobre la lupa y a continuación se nos mostrará un menú desplegable con todas las categorías y si pinchamos sobre una, podemos comprobar que se muestran todos los productos pertenecientes a dicha categoría como se puede observar en la Figura 3.k.

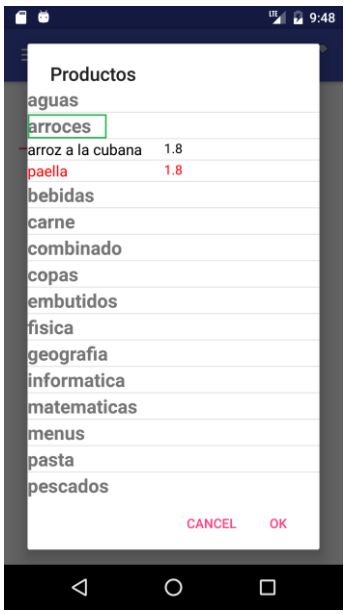


Figura 3.k: Carta de productos.

3.5. Productos fuera de stock.

Aquellos productos que no se encuentren disponibles estarán señalados en rojo en la carta y además, se podrá acceder a ellos pero no se permitirá elegirlos, como se observa en la figura 3.l.

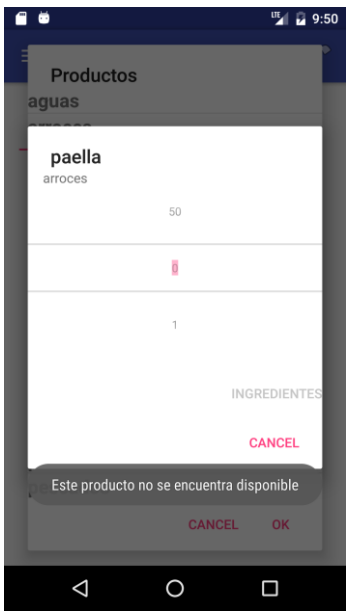


Figura 3.l: Producto fuera de stock

3.6. Elegir ingredientes.

Para añadir o quitar ingredientes de un producto deberemos pinchar sobre el botón “Ingredientes” a la hora de elegir el producto.

Después, aparecerá una lista en la que se podrá marcar o desmarcar los ingredientes. Además, también se podrá buscar un ingrediente concreto mediante el buscador, como se muestra en la figura 3.m.

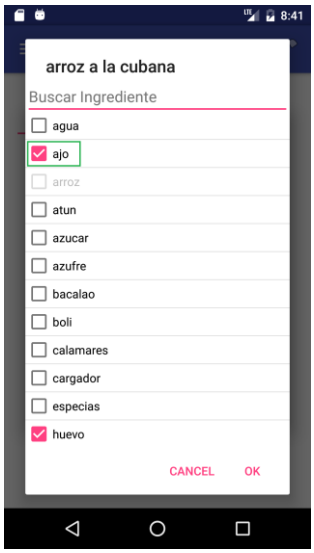


Figura 3.m: Elegir ingredientes

3.7. Cancelar pedido.

Podemos cancelar un pedido antes de enviarlo a cocina pulsando sobre el producto y seleccionando “Cancel” (Figura 3.n) Una vez que hayamos confirmado (Figura 3.o), se habrá eliminado el producto de la lista de pedidos.

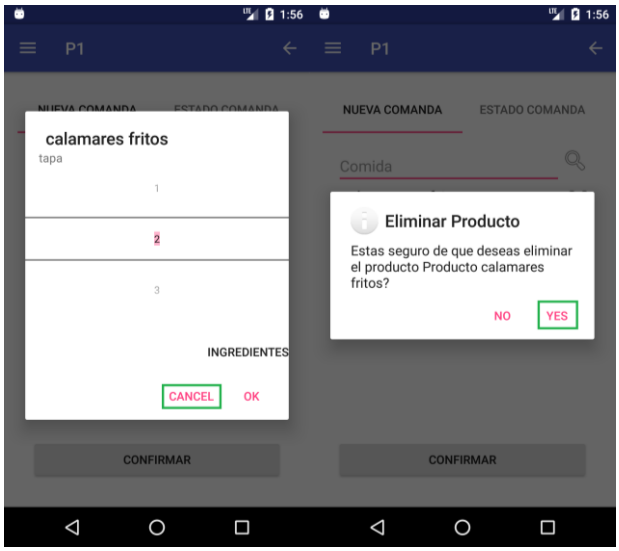


Figura 3.n: Cancelar comanda Figura 3.o: Confirmar

3.8. Confirmar pedido.

Una vez que ya hayamos seleccionado los productos aparecerá un resumen del pedido para comprobar que esté todo correcto, y a continuación pulsamos sobre “Confirmar” (Figura 3p)

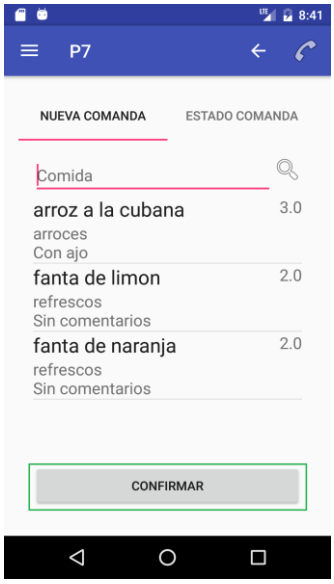


Figura 3.p: Confirmar

3.9. Estado del pedido.

Podemos saber en todo momento en que estado se encuentra nuestro pedido si pulsamos la pestaña al lado de “Nueva comanda” llamada “Estado comanda”. En ella se pueden ver distintos estados dependiendo de la fase en la que se encuentre:

1. En Cola.
2. Sirviendo.
3. Servido.
4. Pagado.
5. Cancelado

Si pulsamos uno de ellos podemos actualizar el estado de la comanda a su siguiente estado como podemos observar en las figuras 3.q y 3.r.

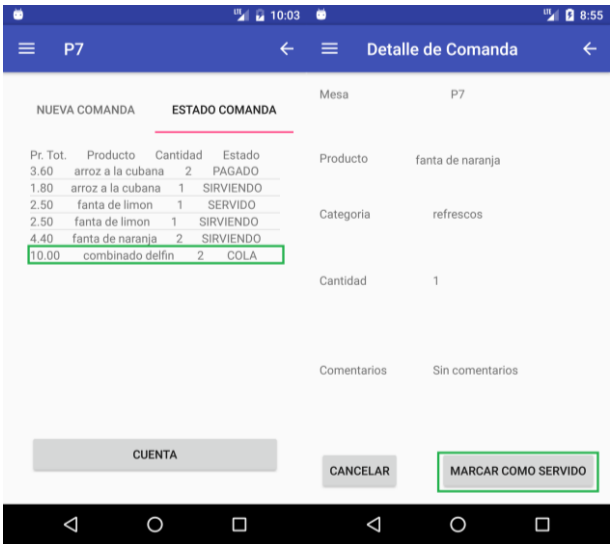


Figura 3.q: Estado inicial. Figura 3.r: actualizar.

Si en vez de actualizar el estado de la comanda a su siguiente estado, pulsamos sobre cancelar y confirmamos, habremos cancelado la comanda, como se puede observar en las figuras 3.s.

Figura 3.s: Cancelar comanda

3.10. Comanda.

Dentro de cada espacio se puede observar un botón en la parte inferior llamado “Detalles” (Figura 3.t). Si pulsamos sobre él, accederemos a la lista de comandas en curso del local.También podemos acceder a las comandas mediante el menú situado en la esquina superior izquierda y pulsando sobre “Comandas” (Figura 3.u).

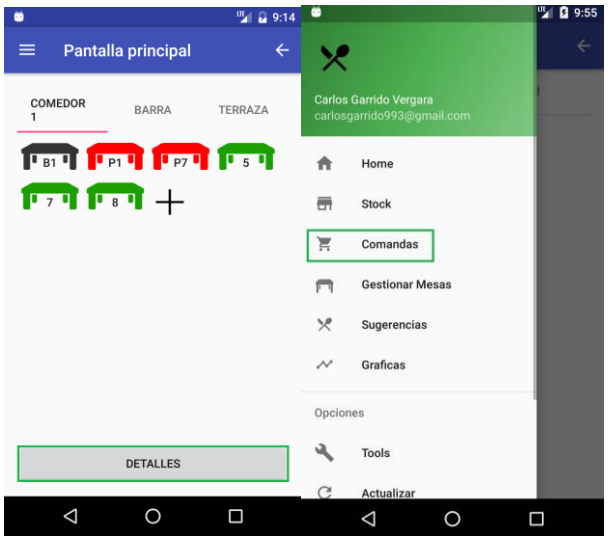


Figura 3.t: Detalles

Figura 3.u:Comandas.

Una vez ahí, podremos pulsar sobre cada una de la comanda e indicar que ya está lista para ser servida como se puede observar en la figura 3.v.

Figura 3.v: Comandas.

3.11. Cancelar comanda.

Si en vez de marcar como sirviendo, pulsamos sobre “Cancelar” se nos mostrará un mensaje de confirmación para cancelar la comanda como se observa en la figura 3.w.

Figura 3.w: Cancelar comanda.

3.12. Cuenta.

Para generar la cuenta de una mesa, desde el estado de comanda de una mesa, pulsamos sobre “Cuenta” (Figura 3.x).

Una vez ahí, se nos mostrará el importe total y tendremos la opción de elegir si se va a pagar mediante efectivo o mediante tarjeta bancaria (Figura 3.y).

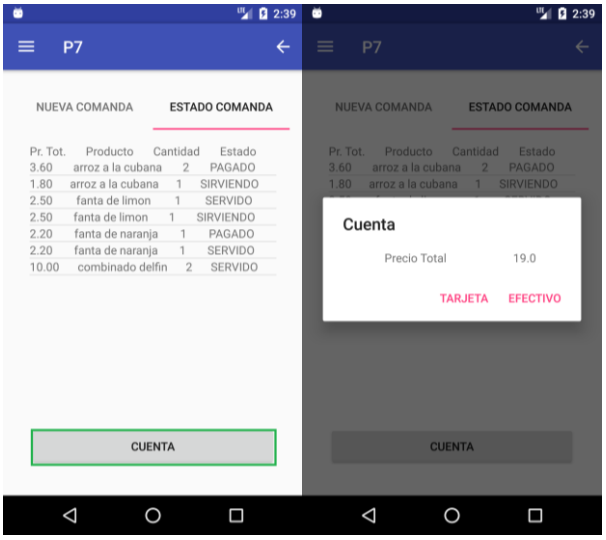


Figura 3.x: Cuenta. Figura 3.y: Modo de pago.

3.13. Limpiar mesa y generar código QR.

Si en la pantalla principal mantenemos pulsado sobre una mesa, tendremos varias acciones dependiendo del estado de la mesa.

- Si la mesa está ocupada, se limpiará de comandas y pasará al estado de libre (Figura 3.z).

Figura 3.z: Limpiar mesa.

- Si la mesa está libre, se generará un código QR de la mesa.

4. Stock.

Si queremos ver el stock del local y administrarlo, podremos hacerlo accediendo al menú de stock. Para ello abrimos el menú desplegable de la esquina superior izquierda y pulsamos sobre stock. Una vez ahí, podemos ver los diferentes productos e ingredientes de los que dispone el local, como podemos observar en la figura 4.a.

Figura 4.a: Stock.

4.1. Productos.

La pantalla por defecto que se visualizará es la de los productos, y dentro de ella encontraremos información de ellos al pulsar sobre cada uno como la opción de insertar un nuevo producto, categoría o ingrediente, o un buscador para buscar un producto en concreto.

4.1.1. Cambiar stock de un producto.

Si queremos cambiar el estado de un producto de disponible a agotado, buscamos el producto mediante el buscador o buscando directamente en la lista de productos

Una vez localizado el producto, lo seleccionamos y pulsamos en la pestaña de “En stock” dependiendo del estado que tenga el producto.(Figura 4.c)

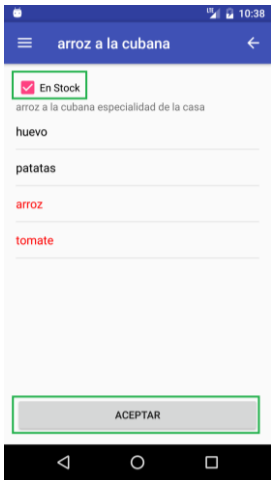


Figura 4.c: Cambiar estado Stock

4.1.2. Cambiar stock de un ingrediente en un producto.

Puede suceder que se haya acabado un ingrediente que compone el producto, y en ese caso podremos modificarlo pulsando sobre el ingrediente y sobre “Aceptar”, como se muestra en la figura 4.d.

Figura 4.d Cambio de stock de un ingrediente

4.1.3. Insertar producto.

Si queremos insertar un producto nuevo al local, deberemos pulsar sobre el botón “+”.

Aquí, podremos añadir una categoría o un ingrediente en caso de que hiciera falta, pulsando sobre el botón “+” en el desplegable de categorías.

A continuación, se rellena el nombre de la categoría y pulsamos sobre “Ok”.

Para insertar un nuevo ingrediente, seguiremos el mismo proceso.

Finalmente, rellenamos los datos del producto y pulsamos sobre “OK” como se muestra en la figura 4.e.

Figura 4.e: Creación de un producto.

4.1.4. Modificar producto

Para modificar un producto pulsamos sobre el producto deseado y modificamos los datos que deseemos cambiar. Finalmente, pulsamos sobre “Aceptar”.

4.1.5. Eliminar producto.

Si queremos eliminar un producto, pulsamos sobre el botón “Eliminar” ubicado en la la parte inferior de la pantalla y pulsamos sobre “Aceptar” como se muestra en la figura 4.f.

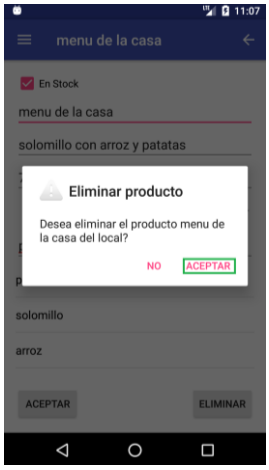


Figura 4.f: Eliminar producto.

4.2. Ingredientes.

Si seleccionamos la pestaña “Ingredientes” comprobaremos que tenemos algo muy parecido a productos

4.2.1. Insertar ingrediente.

Para insertar un ingrediente pulsamos sobre el botón “+” y a continuación introducimos el nombre del ingrediente. Por último, pulsamos sobre “OK”.

4.2.2. Modificar ingrediente.

Para modificar un ingrediente, lo elegimos en la lista y lo seleccionamos.

A continuación, modificamos el nombre y pulsamos sobre “ACEPTAR”.

4.2.3. Modificar stock ingrediente.

Para modificar el stock de un producto seleccionamos el producto de la lista y marcamos o desmarcamos el botón “En stock” y a continuación pulsamos sobre “ACEPTAR”.

4.2.4. Eliminar ingrediente.

Para eliminar un ingrediente, al igual que con el resto de acciones, seleccionamos el ingrediente deseado, pulsamos sobre “ELIMINAR”.

5. Mapa.

Para acceder al mapa y buscar restaurantes, pulsaremos sobre el menú lateral de la esquina superior izquierda y pulsaremos sobre “Mapa”. Una vez dentro, se nos mostrarán todos los locales que han configurado su ubicación en la app.

Además, mediante el buscador, podremos buscar un restaurante mediante su nombre como se muestra en la figura 5.a.

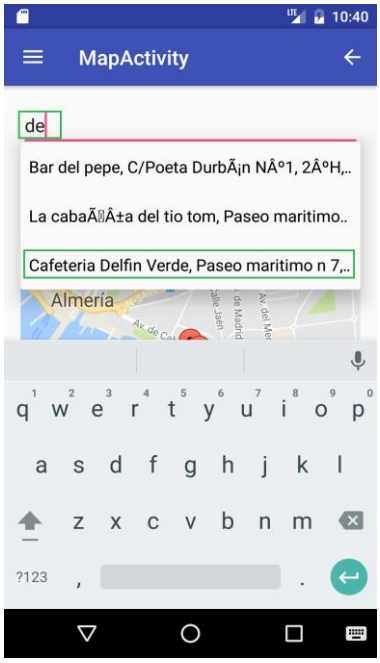


Figura 5.a: Buscador de local

Si pinchamos sobre un local y pulsamos sobre “Aceptar”, se nos mostrará las mesas en cada uno de los espacios del local dibujadas de un color dependiendo de si están libres (verde) o están ocupadas (rojo). Si es el local en el que nos encontramos, tendremos la opción de pulsar sobre la mesa para ver nuestras comandas, como podemos observar en la figura 5.b.

Figura 5.b: Estado de mesas.

Si por el contrario al pulsar sobre un local pinchamos en “Cancel” se nos abrirán en la esquina inferior derecha dos iconos que nos llevarán a google maps para mostrar el “Cómo llegar” (Figura 5.c).

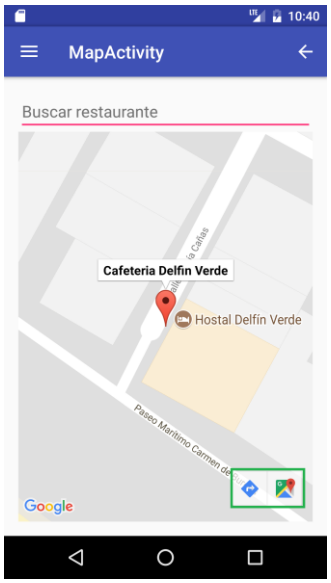


Figura 5.c: Cómo llegar.

6. Gestionar mesas.

Si desde cualquier lugar de la aplicación pulsamos sobre el botón situado en la esquina superior derecha y seguidamente pulsamos “Gestionar mesas”, accederemos al panel mostrado en la figura 6.a.

Figura 6.a: Gestionar mesas.

6.1. Insertar mesa.

Para insertar una mesa, pulsamos sobre el botón “+” y a continuación introducimos los datos de la mesa. Por último, pulsamos sobre “OK”, como podemos observar en la figura 6.c.

Figura 6.c: Crear mesa.

6.2. Modificar mesa.

Para modificar una categoría, la elegimos en la lista y lo seleccionamos.

A continuación, modificamos el nombre y pulsamos sobre “ACEPTAR”.

6.3. Eliminar mesa.

Para eliminar una mesa, al igual que con el resto de acciones, seleccionamos la mesa deseada, pulsamos sobre “ELIMINAR” y luego confirmamos.

6.4. Asignar mesas.

Para asignar mesas a un empleado, para que le lleguen las notificaciones de la mesa, pulsamos sobre el botón “ASIGNAR”.

Para asignar las mesas, se seleccionan las mesas y pulsamos “ASIGNAR”(Figura 6.d). A continuación confirmamos la selección.

Figura 6.d: Asignar mesas.

6.5. Generar códigos qr.

Para generar los códigos qr desde la ventana de asignar mesas, pulsamos sobre el botón de “Generar código”.

A continuación. seleccionamos las mesas cuyos códigos queremos generar y por último pulsamos sobre “OK” (Figura 6.e).

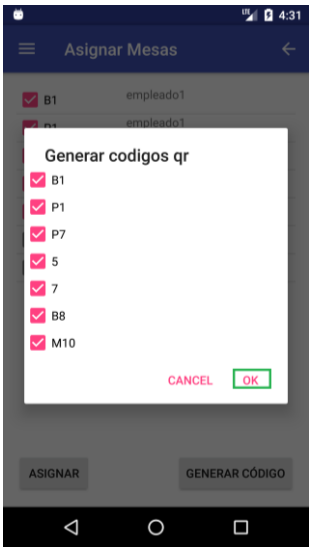


Figura 6.e: Generar código QR

7. Sugerencias

Cada vez que abrimos de nuevo la aplicación, se nos mostrará un mensaje con los platos especiales que se ofertan en el local, pero también podemos ver las sugerencias siempre que queramos de la siguiente forma.

En el menú desplegable, seleccionamos “Sugerencias” y automáticamente se nos mostrará una lista de todas las sugerencias que el local pone a disposición del cliente.

7.1. Insertar sugerencia.

Para insertar una sugerencia pulsaremos sobre el botón “+” en la esquina inferior derecha, rellenamos los datos y pulsamos sobre “OK”(Figura 7.a).

Figura 7.a: Crear sugerencia.

7.2. Modificar stock.

Para modificar si una sugerencia es visible al público o no, basta con seleccionar la pestaña en la parte izquierda de la sugerencia.

7.3. Modificar sugerencia.

Para modificar una sugerencia, pulsamos sobre el botón situado a la derecha de la sugerencia. Una vez pulsado, se modifican los datos y pulsamos sobre “OK”.

7.4. Eliminar sugerencia.

Para eliminar una sugerencia, pulsamos sobre el botón de la derecha de la sugerencia y pulsamos sobre “Eliminar”. A continuación, pulsamos sobre “OK”.

8. Gráfica.

Si abrimos el menú situado en la esquina superior izquierda y pulsamos sobre “Gráficas”, accederemos al apartado de estadísticas.(Figura 8).

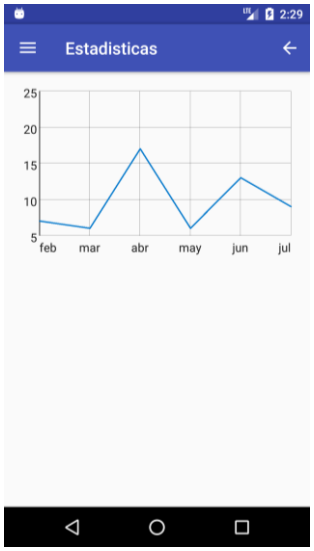


Figura 8: Gráfica.

En dicha gráfica se pueden observar el número de comandas generadas en los últimos 6 meses.

9. Tools.

En cualquier momento podemos modificar algunos datos de nuestra cuenta como la dirección o el nombre de pila y apellidos. Para ello, abrimos el menú lateral y pulsamos sobre “Tools”.

El resultado se observa en la figura 9.a.

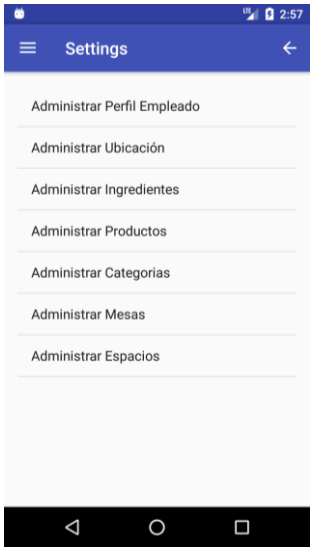


Figura 9.a

9.1. Administrar perfil empleado.

Para modificar el perfil del empleado, se pulsa sobre “Administrar perfil empleado”.

A continuación se rellenan los datos deseados y pulsamos “Actualizar”, como vemos en la figura 9.b.

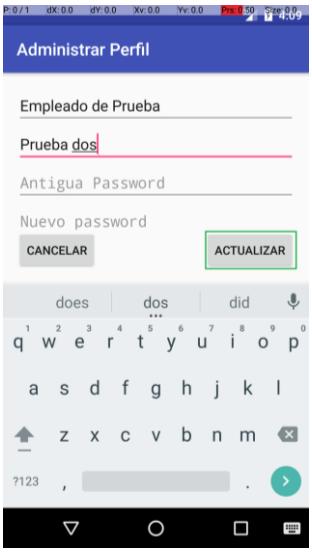


Figura 9.b: Administrar perfil

9.2. Administrar ubicación.

Véase sección 5.

9.3. Administrar ingredientes.

Véase sección 4.7.

9.4. Administrar productos.

Véase sección 4.1.

9.5. Administrar categorías.

9.5.1. Insertar categorías.

Para insertar una categoría pulsamos sobre el botón “+” y a continuación introducimos el nombre de la categoría. Por último, pulsamos sobre “OK”.

9.5.2. Modificar categoría.

Para modificar una categoría, la elegimos en la lista y lo seleccionamos.

A continuación, modificamos el nombre y pulsamos sobre “ACEPTAR”.

9.5.3. Eliminar categorías.

Para eliminar una categoría, al igual que con el resto de acciones, seleccionamos la categoría deseada, pulsamos sobre “ELIMINAR” y luego confirmamos.

9.6. Administrar mesas.

Véase sección 6.

9.7. Administrar espacios.

9.7.1. Insertar espacio.

Para insertar un espacio pulsamos sobre el botón “+” y a continuación introducimos el nombre del espacio. Por último, pulsamos sobre “OK”.

9.7.2. Modificar espacio.

Para modificar un espacio, lo elegimos en la lista y lo seleccionamos.

A continuación, modificamos el nombre y pulsamos sobre “ACEPTAR”.

9.7.3. Eliminar espacio.

Para eliminar un espacio, al igual que con el resto de acciones, seleccionamos el espacio deseado, pulsamos sobre “ELIMINAR” y luego confirmamos.

10. Actualizar.

Para actualizar la base de datos local de la aplicación, desde cualquier pantalla de la aplicación, pulsamos sobre la esquina superior derecha y pulsamos sobre “Actualizar”.

11. Información.

Desde la sección de información podemos ver la versión de la aplicación, los datos de contacto y la última actualización del software.(Figura 11)

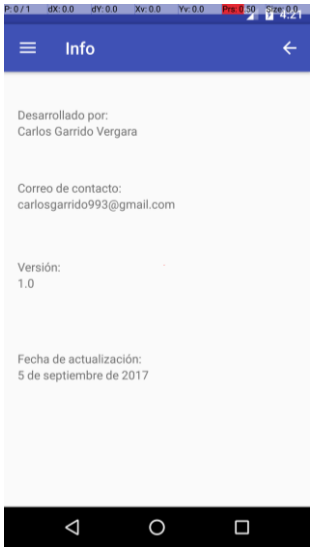


Figura 10. Información.

Aplicación WEB para locales.

12. Registrar restaurante.

Para registrarnos en la página web primero deberemos habernos registrado en la aplicación móvil.

Una vez hecho esto, desde la página principal (https://comandual.herokuapp.com/), pulsamos sobre “Iniciar sesión o crear una cuenta nueva”.

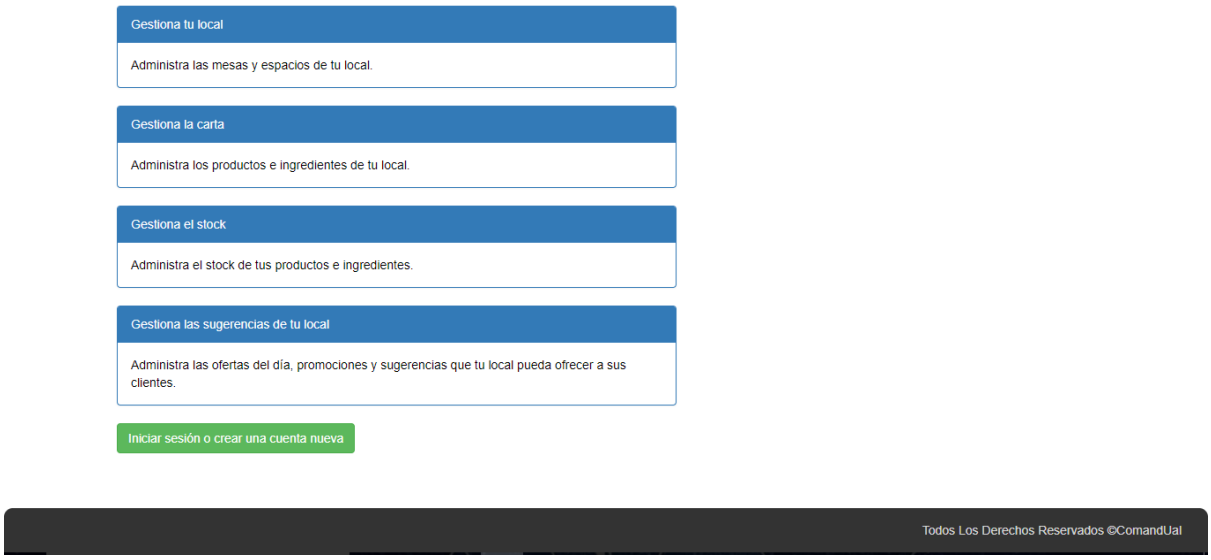


Figura 11: Página inicial de la aplicación web.

A continuación pulsamos sobre “Crear una cuenta gratuita”.

En el siguiente paso rellenamos los datos y pulsamos sobre “Crear” y nos devolverá a la página de login. Deberemos tener en cuenta las siguientes restricciones.

- El nombre de usuario debe de coincidir con el nombre de usuario de la aplicación web.
- Ni el usuario ni el email deben estar ya registrados en la aplicación web.
- La contraseña no puede ser menor de 8 caracteres, contener el nombre de usuario ni contener sólo números o sólo letras.

13. Login y Logout.

13.1. Login.

Para loguearse en la página web, deberemos acceder a la página principal (https://comandual.herokuapp.com/) y pulsar sobre “Iniciar sesión o crear una cuenta nueva”.

A continuación introducimos el nombre de usuario y contraseña y pulsamos sobre “Iniciar sesión”.

13.2. Logout.

Podemos cerrar sesión de dos maneras distintas.

Desde cualquier pantalla de la aplicación abrimos el menú desplegable pulsando el botón de la parte superior izquierda y pulsamos sobre “Cerrar sesión”.

Desde la pantalla principal pulsando sobre “Cerrar sesión”.

14. Pantalla principal.

Una vez iniciada sesión la página web se dirigirá a la pantalla principal.

Además, se puede acceder a ella desde el menú lateral, pulsando sobre el nombre de usuario o escribiendo en el navegador la dirección de la pantalla principal de la página web.(Figura 12)

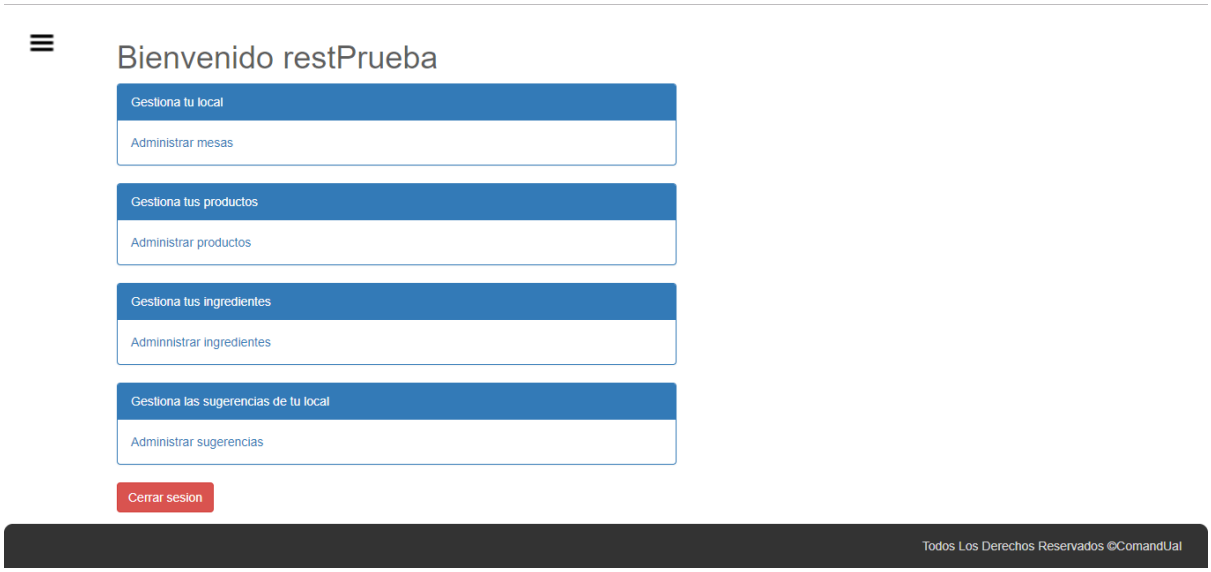


Figura 12: Pantalla principal de usuario.

15. Administrar mesas y espacios.

Podemos administrar mesas desde la pantalla principal o desde el menú lateral pulsando sobre “Administrar mesas”

15.1. Crear mesa.

Para crear una nueva mesa, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”(Figura 13.a).

Rellenamos los datos y pulsamos sobre “Guardar”.(Figura 13.b)

Figura 13.a Panel de control de mesas.

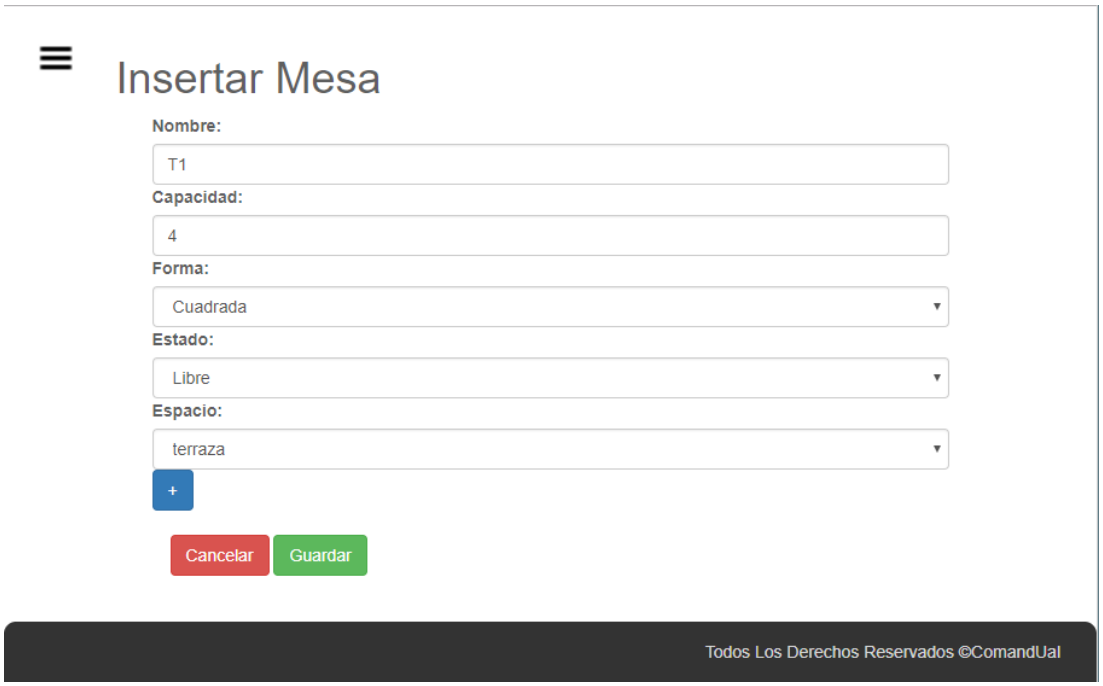


Figura 13.b: Crear mesa.

15.2. Editar mesa.

Para editar una mesa, desde la pantalla de administrar mesas, pulsamos sobre la acción “Editar” en la mesa deseada.

Seguidamente, se modifican los datos deseados y se pulsa “Guardar”.

15.3. Eliminar mesa.

Para eliminar una mesa, desde la pantalla de administrar mesas, pulsamos sobre la acción “Eliminar” en la mesa deseada.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar” (Figura 13.c).

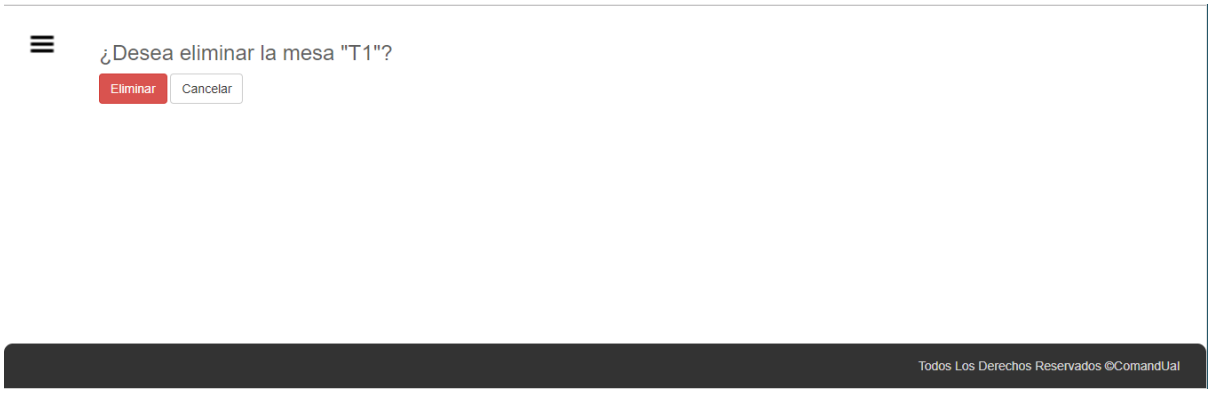


Figura 13.c: Eliminar mesa.

15.4. Crear espacio.

Para crear un nuevo espacio pulsaremos sobre el botón “+” situado al lado de la opción para elegir espacio en el modo crear o editar mesa y se nos mostrará el panel de administración de espacios.

En el siguiente paso, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”, rellenamos los datos y pulsamos sobre “Guardar”.(Figura 13.d)

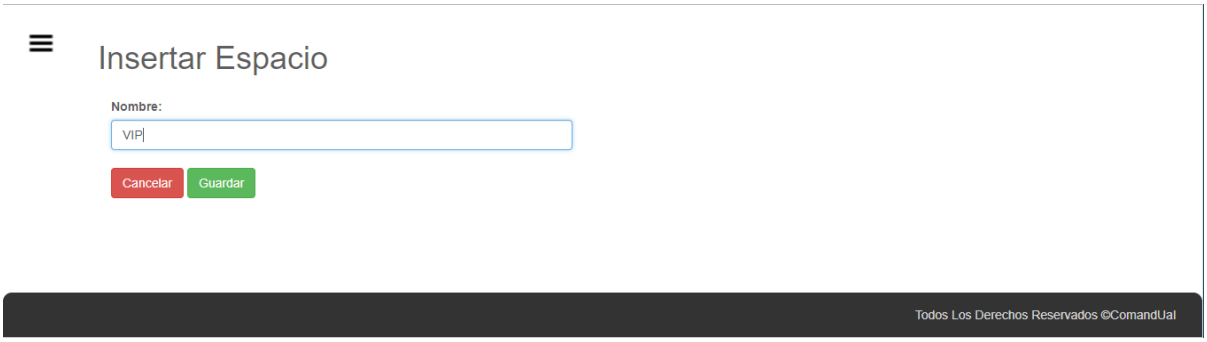


Figura 13.d: Insertar espacio

15.5. Editar espacio.

Para editar un espacio, desde la pantalla de administrar espacios, pulsamos sobre la acción “Editar” en el espacio deseado.

Seguidamente, se modifican los datos deseados y se pulsa “Guardar”.

15.6. Eliminar espacio.

Para eliminar un espacio, desde la pantalla de administrar espacios, pulsamos sobre la acción “Eliminar” en el espacio deseado.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar”.(Figura 13.e)

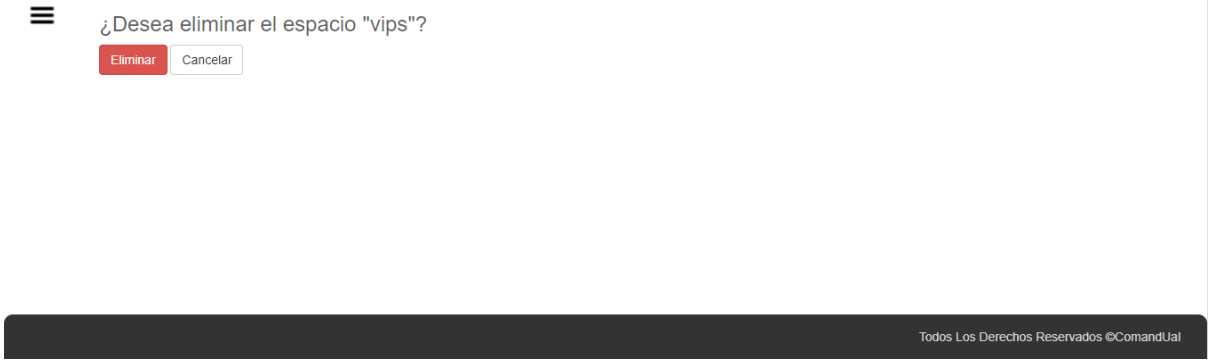


Figura 13.e: Eliminar espacio.

16. Administrar productos y categorías.

Podemos administrar productos desde la pantalla principal o desde el menú lateral pulsando sobre “Administrar productos”.

16.1. Crear producto.

Para crear un nuevo producto, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”.

Rellenamos los datos y pulsamos sobre “Guardar”.

16.2. Editar producto.

Para editar un producto, desde la pantalla de administrar productos, pulsamos sobre la acción “Editar” en el producto deseado.

Seguidamente, se modifican los datos deseados y se pulsa “Guardar”.

16.3. Eliminar producto.

Para eliminar un producto, desde la pantalla de administrar producto, pulsamos sobre la acción “Eliminar” en el producto deseado.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar”.

16.4. Crear categoría.

Para crear una nueva categoría, pulsaremos sobre el botón “+” situado al lado de la opción para elegir categoría en el modo crear o editar producto y se nos mostrará el panel de administración de categorías.

En el siguiente paso, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”, rellenamos los datos y pulsamos sobre “Guardar”.

16.5. Editar categoría.

Para editar una categoría, desde la pantalla de administrar categorías, pulsamos sobre la acción “Editar” en la categoría deseada.

Seguidamente, se modifican los datos deseados y se pulsa “Guardar”.

16.6. Eliminar categoría.

Para eliminar una categoría, desde la pantalla de administrar categorías, pulsamos sobre la acción “Eliminar” en la categoría deseada.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar”.

17. Administrar ingredientes.

Podemos administrar ingredientes desde la pantalla principal o desde el menú lateral pulsando sobre “Administrar ingredientes”.

17.1. Crear ingrediente.

Para crear un nuevo ingrediente, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”.

Rellenamos los datos y pulsamos sobre “Guardar”.

17.2. Editar ingrediente.

Para editar un ingrediente, desde la pantalla de administrar ingredientes, pulsamos sobre la acción “Editar” en el ingrediente deseado.

Seguidamente, se modifican los datos deseados y se pulsa “Guardar”.

17.3. Eliminar ingrediente.

Para eliminar un ingrediente, desde la pantalla de administrar ingredientes, pulsamos sobre la acción “Eliminar” en el ingrediente deseado.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar”.

18. Administrar sugerencias.

Podemos administrar sugerencias desde la pantalla principal o desde el menú lateral pulsando sobre “Administrar sugerencias”.

18.1. Crear sugerencia.

Para crear una nueva sugerencia, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”.

Rellenamos los datos y pulsamos sobre “Guardar”.

18.2. Editar sugerencia.

Para editar una sugerencia, desde la pantalla de administrar sugerencias, pulsamos sobre la acción “Editar” en la sugerencia deseada.

Seguidamente, se modifican los datos deseados y se pulsa “Guardar”.

18.3. Eliminar sugerencia.

Para eliminar una sugerencia, desde la pantalla de administrar sugerencias, pulsamos sobre la acción “Eliminar” en la sugerencia deseada.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar”.

19. Administrar empleados.

A diferencia del resto de opciones, para dar seguridad, sólo podremos administrar empleados desde el menú lateral pulsando sobre “Administrar empleados”.

19.1. Crear empleado.

Para crear un nuevo empleado, pulsaremos sobre el botón “+” a la derecha de la columna “Acciones”.

Rellenamos los datos y pulsamos sobre “Guardar”.

19.2. Eliminar empleado.

Para eliminar un empleado, desde la pantalla de administrar empleados, pulsamos sobre la acción “Eliminar” en el empleado deseado.

En el siguiente paso, confirmamos la acción pulsando sobre “Eliminar”.

Aplicación móvil para clientes.

20. Registrar usuario.

Para registrarnos como usuario pinchamos en el botón “Registrate” y a continuación seleccionaremos “Registrarse como Cliente”.

21. Login y Logout.

Para loguearse en la aplicación basta con poner el nombre de usuario y contraseña y pinchar sobre “Login”.

Para cerrar sesión, desde cualquier pantalla de la aplicación abrimos el menú desplegable pulsando el botón de la parte superior izquierda y pulsamos sobre “Cerrar sesión”.

22. Escanear mesa.

Por defecto, la primera pantalla que se nos mostrará será la de escanear la mesa de un restaurante. Para ello, pulsaremos sobre el botón “Escanear mesa” y a continuación aparecerá un lector de código qr en el que deberemos enfocar el código insertado en la mesa y ya podremos acceder a la mesa.

23. Realizar pedido.

Tenemos dos formas de realizar un pedido a cocina, bien eligiendo el producto directamente desde un buscador de productos o bien mediante una carta de productos.

23.1. Mediante buscador.

Para realizar el pedido mediante el buscador, sólo deberemos introducir el nombre del producto en él, y seleccionar el producto deseado.

Como ejemplo, vamos a seleccionar 2 platos de arroz a la cubana y 2 platos conbinados, con 3 fantas de naranja y una fanta de Limón.

Como podemos comprobar, el buscador muestra todos los productos que contengan las palabras introducidas.

23.2. Mediante carta.

Para realizar un pedido mediante la carta, pulsaremos sobre la lupa y a continuación se nos mostrará un menú desplegable con todas las categorías y si pinchamos sobre una, podemos comprobar que se muestran todos los productos pertenecientes a dicha categoría.

23.3. Productos fuera de stock.

Aquellos productos que no se encuentren disponibles estarán señalados en rojo en la carta y además, se podrá acceder a ellos pero no se permitirá elegirlos.

23.4. Elegir ingredientes.

Para añadir o quitar ingredientes de un producto deberemos pinchar sobre el botón “Ingredientes” a la hora de elegir el producto.

Después, aparecerá una lista en la que se podrá marcar o desmarcar los ingredientes. Además, también se podrá buscar un ingrediente concreto mediante el buscador.

23.5. Confirmar pedido.

Una vez que ya hayamos seleccionado los productos aparecerá un resumen del pedido para comprobar que esté todo correcto, y a continuación pulsamos sobre “Confirmar pedido”.

23.6. Estado del pedido.

Podemos saber en todo momento en que estado se encuentra nuestro pedido si pulsamos la pestaña al lado de “Nueva comanda” llamada “Estado comanda”. En ella se pueden ver distintos estados dependiendo de la fase en la que se encuentre:

1. En Cola.
2. Sirviendo.
3. Servido.
4. Pagado.

24. Llamar al camarero.

Si en algún momento es necesario llamar al camarero, se podrá hacer sin necesidad de que el camarero nos vea llamarlo.

Para ello, pulsaremos sobre el botón con un teléfono que se encuentra en la parte superior derecha de la pantalla.

25. Stock.

Si queremos ver qué productos ofrece el local, podremos verlos accediendo al menú de stock. Para ello abrimos el menú desplegable de la esquina superior izquierda y pulsamos sobre stock. Una vez ahí, podemos ver los diferentes productos e ingredientes de los que dispone el local.

25.1. Productos.

La pantalla por defecto que se visualizará es la de los productos, y dentro de ella encontraremos información sobre ellos al pulsar sobre cada uno, así como un buscador para buscar un producto en concreto.

25.2. Ingredientes.

Si seleccionamos la pestaña “Ingredientes” comprobaremos que tenemos algo muy parecido a productos.

26. Mapa.

Para acceder al mapa y buscar restaurantes, pulsaremos sobre el menú lateral de la esquina superior izquierda y pulsaremos sobre “Mapa”. Una vez dentro, se nos mostrarán todos los locales que han configurado su ubicación en la app.

Además, mediante el buscador, podremos buscar un restaurante mediante su nombre.

Si pinchamos sobre un local y pulsamos sobre “Aceptar”, se nos mostrará las mesas en cada uno de los espacios del local dibujadas de un color dependiendo de si están libres (verde) o están ocupadas (rojo). Si es el local en el que nos encontramos, tendremos la opción de pulsar sobre la mesa para ver nuestras comandas.

Si por el contrario al pulsar sobre un local pinchamos en “Cancel” se nos abrirán en la esquina inferior derecha dos iconos que nos llevarán a google maps para mostrar el “Cómo llegar”.

27. Sugerencias.

Cada vez que abrimos de nuevo la aplicación, se nos mostrará un mensaje con los platos especiales que se ofertan en el local, pero también podemos ver las sugerencias siempre que queramos de la siguiente forma.

En el menú desplegable, seleccionamos “Sugerencias” y automáticamente se nos mostrará una lista de todas las sugerencias que el local pone a disposición del cliente.(Figura 14).

Figura 14: Sugerencias.

28. Tools.

En cualquier momento podemos modificar algunos datos de nuestra cuenta como la dirección o el nombre de pila y apellidos. Para ello, abrimos el menú lateral y pulsamos sobre “Tools”. A continuación pulsamos sobre Administrar perfil.

Una vez dentro, se cargará la información de nuestra cuenta y se podrá modificar.

29. Pedir cuenta.

Si queremos pedir la cuenta, debajo de la lista de estados de las comandas, veremos la opción “Cuenta”, sobre la que deberemos pinchar. una vez pulsado se nos mostrará el total a pagar y si deseamos pagar con tarjeta o en efectivo. Una vez elegido, se mandará una notificación al empleado para que atienda la petición.

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

Aplicación para la gestión de comandas de locales que permite a los empleados del local administrar espacios con mesas, su carta con productos, categorías de productos, ingredientes y platos, generar comandas y cuentas, ubicarse en google Maps así como mostrar información referente al local, y a sus clientes consultar los locales ubicados en google Maps, consultar la carta y el estado de las mesas de un local, escanear una mesa con código QR para generar comandas automáticamente y la cuenta y valorar el servicio ofrecido por el local

/

Application for the management of restaurant commands that allows employees to manage spaces with tables, their menu with products, product categories, ingredients and dishes, generate commands and accounts, locate on Google Maps as well as show information about the local, and their customers, check the locations located on Google Maps, check the menu and the status of a restaurant tables, scan a table with QR code to automatically generate commands and the account and evaluate the service offered by the restaurant.