

19 DE JUNIO DE 2022



DOCUMENTACIÓN SHEISTYLE

SHEILA NIEVA LOPEZ
CIPF VIRGEN DE GRACIA
Desarrollo aplicaciones multiplataforma

Índice

1. Presentación del proyecto	2
1.1. Estudio del mercado	3
1.2. Valor del producto	7
2. Análisis de la solución.....	7
2.1. Requisitos funcionales	7
2.2. Requisitos no funcionales.....	9
2.3. Requisitos de información.....	9
3. Planificación de tareas y estimación de costes.	10
3.1. Planificación y organización de tareas	10
❖ Uso de GitFlow como herramienta de trabajo	15
❖ Wiki del proyecto	17
3.2. Estimación de costes y recursos	18
3.3. Herramientas usadas	19
3.4. Gestión de riesgos.....	20
4. Análisis de escenario	24
4.1. Casos de uso	24
4.2. Diseño de la interfaz y prototipos	26
4.3. Diagrama de clases.....	38
4.4. Diseño de la persistencia de la información	43
❖ Base de datos NoSQL.....	43
❖ Conclusión.....	44
4.5. Diagrama de la persistencia de la información.....	45
4.6. Arquitectura del sistema	46
5.Implementación de la solución	48
5.1. Análisis tecnológico	48
❖ Tecnología del cliente	48
❖ Tecnología del servidor	50
5.2 Elementos a implementar	53
6. Bibliografía.....	95

1. Presentación del proyecto

Este proyecto, va a estar desarrollado en C# y su base de datos estará alojada en Firebase. Se basará en una gestión de nuestra empresa, que será un local de cuidados estéticos. En el desarrollo de esta aplicación buscaremos su sencillez y que sea intuitiva, para poder ser administrada y usada por los propios trabajadores de la empresa.

La aplicación recibe el nombre de SheiStyle.



En la ventana principal de dicha aplicación, aparecerán diferentes opciones.

En la primera opción nos aparecerá un calendario. Al pulsar sobre un día, nos abrirá una ventana emergente en forma de agenda, donde podremos seleccionar la hora deseada, comprobando que esté libre. Además de seleccionar la hora, tendremos que seleccionar el nombre de la cliente (si es nueva, se podrá dar de alta desde aquí ya que se abrirá otra ventana emergente para el registro), el tipo servicio, y dependiendo de esto último, el precio se estimará automáticamente.

Todo esto se registrará en un historial.

Según el número de visitas que tenga la clienta registrada en nuestra peluquería, recibirá un tipo de bono de descuento.

En la siguiente opción, podremos gestionar los clientes. Mostraremos un listado y desde aquí los podremos añadir, modificar y eliminar. Tendremos la opción de visualizar el historial de citas de un cliente seleccionando su nombre.

Otra de las opciones será la sección de precios. En esta nos parecerán todos nuestros servicios con su precio indicado y tendremos opción de modificarlos (su precio) en caso de que la situación lo requiera.

Por último, tendremos un botón de favoritos, donde nos aparecerán los servicios más demandados organizado en sectores. Mostrando para esto un gráfico circular con el porcentaje según demanda de cada sector. Este apartado nos va a ser útil para tener en cuenta nuestros servicios más demandados y poder especializarnos en ellos.

Por otra parte, en la parte superior tendremos un menú donde nos aparecerán dos opciones, como la de salir de la aplicación y otra que será para gestionar la empresa.

En esta última podremos ver toda la información sobre nuestra tienda, como el nombre, la dirección física de donde está situada, formas de contacto, como son el teléfono y la dirección de correo, el número de trabajadores que hay actualmente ofreciendo sus servicios... Todo esto se podrá modificar desde esta pestaña, para así poder tener la información actualizada de nuestra tienda.

Para concluir habrá una opción donde podremos observar los ingresos de cada mes mediante un gráfico, pudiendo concretar así si nuestra empresa va creciendo y tenemos que tomar la decisión de agrandarla, o en caso contrario, debemos analizar qué cosas van mal para poder mejorar.

1.1. Estudio del mercado

Un estudio de mercado consiste en una iniciativa empresarial con el fin de hacerse una idea sobre la viabilidad comercial de una actividad económica

Este estudio tiene un objetivo económico, es decir, su objetivo es generar beneficio económico con la actividad económica cuya validez queremos probar en el mercado. Esta investigación busca anticipar la respuesta de los clientes potenciales y la competencia ante un producto o servicio concreto, por lo que no solo se hacen estudios de mercado cuando queremos emprender una aventura empresarial, si no siempre que queramos probar nuestros productos o servicios, saber cómo podemos mejorarlos, de qué manera posicionarlos en el mercado, etc. De esta manera, con un estudio de mercado bien realizado, conoceremos el perfil y el comportamiento de nuestros clientes, la situación del mercado o industria a la que nos dedicamos, cómo trabaja nuestra competencia, etc. Incluso podemos llegar a descubrir nuevas necesidades que no conocíamos de nuestro público objetivo.

Un estudio de mercado se apoya sobre 4 pilares fundamentales

- La información del sector
- Conocer a nuestro target (público objetivo)
- Conocer a nuestra competencia
- Análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades)

De acuerdo con esto, voy a realizar un estudio de mercado sobre la aplicación, pasando por cada uno de estos puntos:

- **Información del sector:**

El sector de peluquería y estética es el sector económico que engloba locales donde se ofrecen diversos servicios estéticos, principalmente el corte de pelo, pero también otros como afeitado, depilado, manicura o pedicura. Cuando el local ofrece muchos servicios diferentes se denomina salón de belleza, como será nuestro caso.

Esta aplicación de escritorio que crearemos, nos ayudará en la gestión de nuestro local. Tanto para llevar un listado de los clientes que pasan por nuestra tienda, los ingresos que tenemos cada mes como para reservar citas con el tipo de información necesaria.

En conclusión, es un sector que nos permite mantener un tipo de salud corporal. Y algunos de los servicios que se ofrecen en este sector, son muy usuales o necesarios cada cierto tiempo.

- **Target o público objetivo:**

Nos referimos a aquel grupo de personas que debido a sus cualidades y características tiene un alto potencial, o existe una alta probabilidad de que pueda llegar a ser en el futuro un consumidor de nuestro producto o servicio. Es decir, es toda aquella persona que puede llegar a interesarle lo que vendemos /ofrecemos.

Siguiendo estos pasos, podremos llegar a conocer de manera profunda al público al que vamos a destinar todos nuestros esfuerzos para atraerlos a nuestro servicio o bien, fidelizarlos en el caso de que ya sean clientes actuales.

Los pasos a seguir son los siguientes:

- Sexo
- Edad
- Lugar
- Poder adquisitivo
- Clase social

Para nuestro local, todo esto es orientativo en cierta medida, porque este sector puede formar parte de un cambio radical (físicamente) de las personas. Por ejemplo, es habitual que cada persona elija normalmente unos servicios de peluquería similares, pero también puede darse el caso de que deseen un cambio de look totalmente diferente al que suelen llevar. Puede influir también su lugar demográfico, sus costumbres o su ámbito social.

En nuestro caso, podemos definir el público objetivo como personas sin distinción de sexos y sin rango de edad, de cualquier parte del mundo, sin importar la formación educativa, poder adquisitivo o clase social. En realidad,

podría decirse que se trata de cualquier persona con la necesidad o el gusto de cuidar un poco su imagen personal.

- **Análisis de la competencia**

No hay que realizar un análisis muy exhaustivo para saber que el sector al que queremos entrar está plagado de competencia. Y esta competencia son pequeños o grandes centros de belleza, que son fáciles de encontrar en cualquier sitio que nos dirijamos. A pesar de ser un sector básico, los servicios también serán muy similares, aunque todo esto se diferenciará mediante la profesionalidad que nuestro local presentará, el cuidado y trato a nuestros clientes y la actualización continua; siempre estando al día según la moda.

Análisis DAFO:



Para resumir el análisis estratégico, se van a detallar conjuntamente todas las conclusiones extraídas tanto del análisis externo como del interno, formando así una matriz DAFO. En ella, se representan las características internas, es decir, las debilidades y las fortalezas que posee la empresa, y también la situación externa a ella, como son las oportunidades y las amenazas que derivan del entorno, permitiendo así obtener una visión global del contexto en el que se encuentra la organización. Es una herramienta que se utiliza durante la fase de planificación estratégica, la cual sirve para identificar las ventajas competitivas que dispone la empresa en comparación a la demás dentro del sector. Además, es una guía para conocer cuál es el mejor plan de actuación frente a todo lo deducido y tomar las decisiones más adecuadas según las estrategias que se pretenden seguir.

○ **Debilidades**

- No es una empresa reconocida: se trata de una empresa totalmente nueva en el mercado, por lo que los consumidores no pueden tener ninguna opinión y experiencia acerca de ella.
- No lealtad de los consumidores: la empresa se encuentra en un sector en el que los clientes se guían por su confianza y buen trato con los profesionales, a muchos de ellos les parecerá complicado probar nuevos contactos.
- Inversión inicial: aunque no es un gran desembolso inicial, el negocio necesita maquinaria y muchas herramientas para poder llevar a cabo su actividad empresarial.
- Poder negociador de los proveedores alto: la maquinaria y los productos son muy específicos por lo que no existe una gran variedad de empresas que los suministren, así es la empresa la que se tiene que adaptar a las condiciones que imponen los proveedores

○ **Amenazas**

- Elevado número de competidores: hay una gran facilidad para integrarse en el sector y cada vez hay más empresas dedicadas a la imagen y cuidado personal.
- Productos sustitutivos: aunque no se puedan llegar a comparar con los servicios ofrecidos por las empresas del sector belleza, existen productos que prometen tener los mismos resultados que éstas.
- Productos no diferenciados: prácticamente las empresas del sector ofertan más o menos la misma variedad de servicios, por lo que los consumidores tendrán más en cuenta los precios y la calidad.

○ **Fortalezas**

- Precios bajos: se trata de una empresa que presenta precios medios, destinada a todos los consumidores que por su nivel de ingresos no pueden permitirse invertir mucho dinero en el cuidado personal.
- Buena calidad de productos y servicios: la empresa siempre va a intentar cuidar el mínimo detalle para ofrecer sus servicios con la más alta calidad posible.
- Personal formado: los profesionales contratados tendrán conocimientos cualificados para cada puesto de trabajo.
- Buen trato a los clientes: como ya se ha comentado anteriormente, los consumidores en este tipo de empresas buscan la máxima confianza con el trabajador, hecho que la empresa va a tener siempre en cuenta a la hora del contacto con los clientes.

- **Oportunidades**

- Crecimiento del número de los consumidores: cada vez la sociedad tiene una mayor preocupación por la apariencia y cuidado personal, así se considera un sector en crecimiento por su positiva evolución. Cada vez hay más consumidores de este tipo de servicios y en los últimos años, se han ido integrando consumidores de diferentes características, llegando así a un público más amplio y diversificado.
- Aumento del PIB: los niveles de riqueza económica del país han remontado a donde se situaban en los momentos anteriores a la crisis sufrida, lo que significa que la calidad de vida de los habitantes está mejorando. Hecho que conduce a pensar en la mayor disposición al consumo por parte de las familias, y con ello más posibilidades de demanda de los servicios ofrecidos por la empresa.
- Innovaciones tecnológicas: el gran avance sobre la tecnología puede ser aprovechado para invertir en maquinaria más novedosa y sobre todo más sostenible con el medio ambiente. Además, gracias al crecimiento en el uso de la tecnología, ésta puede ser una herramienta útil para mejorar la imagen de la empresa y darse a conocer llegando a más público.
- Aumento de la población: en el estudio sobre la evolución del número de habitantes, se observa un aumento en los últimos años, lo cual puede llegar a suponer una mayor demanda sobre la empresa.

1.2. Valor del producto

Como conclusión a todo esto mencionado anteriormente, diría que todo esto nos ha servido para darnos cuenta que es complicado coger buen standing nada más empezar en este sector, debido a la gran competencia que tenemos. A pesar de ello, vamos a intentar ir creciendo poco a poco, siempre estando al día en todo, en continua formación e innovando. Detalles que nos hagan ser una empresa más cercana a la gente, y puedan confiar en nosotros cada vez más personas.

2. Análisis de la solución

2.1. Requisitos funcionales

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas,

comportamientos y salidas. Tales requisitos pueden ser: detalles técnicos, cálculos, manipulación de datos, ...

Estos serán los requisitos funcionales que nuestra aplicación SheiStyle llevará a cabo. (Cuando nos referimos a 'usuario' en todo momento hacemos referencia a los trabajadores de la empresa que tengan permiso a la gestión de citas).

RF1.- El sistema debe contar con un calendario y una gestión para coger hora.

RF2.- El sistema permitirá al usuario elegir una cita en una fecha concreta.

RF3.- El sistema no dejará coger una hora y fecha que ya tenga una cita reservada.

RF4.- El sistema forzará al usuario a detallar los servicios que se desean consumir para poder reservar la cita.

RF5.- El sistema estimará el coste total de los servicios escogidos en la cita

RF6.- El sistema se encargará de hacer descuentos cada 3 citas cogidas de ese mismo cliente.

RF7.- El sistema permitirá al usuario poder ver y modificar la lista de servicios que llevamos a cabo en nuestro local (con sus precios correspondientes)

RF8.- El sistema contará con un historial de citas de cada cliente.

RF9.- El sistema contará con una búsqueda de clientes que estén dados de alta en nuestra aplicación.

RF10.- El sistema permitirá al usuario ver, editar y eliminar cualquier cita que tenga errores.

RF11.- El sistema permitirá al usuario añadir en cualquier momento un cliente nuevo.

RF12.- El sistema permitirá al usuario borrar cualquier cliente en cualquier momento.

RF13.- El sistema actualizará nuestra agenda y borrará las citas pendientes de aquel cliente que se haya borrado.

RF14.- El sistema permitirá al usuario modificar la ficha del cliente en cualquier momento.

RF15.- El sistema permitirá al usuario consultar en cualquier momento el historial de citas de un cliente.

RF16.- El sistema hará un ranking de los servicios más usados en medida de los servicios consumidos de los clientes.

RF17.- El sistema nos calculará y mostrará en un gráfico los ingresos de cada mes.

RF18.- El sistema permitirá al usuario registrar un nuevo gasto con su descripción y precio indicado.

RF19.- El sistema nos calculará y mostrará en un gráfico los gastos de cada mes.

RF20.- El usuario podrá modificar en cualquier momento los datos sobre la empresa.

RF21.- El usuario podrá salir de la aplicación pulsando el botón de cerrar

2.2. Requisitos no funcionales

Los requisitos no funcionales son características de funcionamiento, especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

A continuación, los requisitos no funcionales de nuestra empresa:

RNF1.- Todos los datos se almacenarán en la base de datos de Firebase.

RNF2.- La aplicación se desarrollará en C# siguiendo el patrón MVVM y estará disponible para dispositivos Windows.

2.3. Requisitos de información

En los requisitos de información vamos a representar entidades e información.

Ingreso: representa la cantidad ingresada en nuestra empresa

- idIngreso
- fecha
- cantidad
- idCita

Empresa: representa el resumen y la información de nuestra empresa

- idEmpresa
- nombre
- dirección
- teléfono
- correo

Servicio: Representa y recoge los servicios que nuestro cliente quiere demandar en nuestra tienda, destacamos los siguientes campos:

- idServicio
- idSector
- nombre
- descripción
- precio

Sector: representa en sectores el tipo de servicios que ofrecemos en nuestro local, destacamos los siguientes campos:

- idSector
- nombre

Cita: recoge todos los datos necesarios para informarnos sobre dicha cita, destacamos los siguientes campos:

- idCitas
- idCliente
- servicios
- fecha
- duracion
- precioCita
- pagado

Cliente: representa a la persona que visita el local de forma usual o no, destacamos los siguientes campos:

- idCliente
- nombre
- apellidos
- telefono
- correo
- citasAcumuladas
- idEmpresa

Gastos: representa los costos que tiene la empresa, los añade el usuario con una descripción y un precio. Destacamos los siguientes campos:

- idPedido
- descripcion
- fecha
- importe
- idEmpresa

3. Planificación de tareas y estimación de costes.

3.1. Planificación y organización de tareas

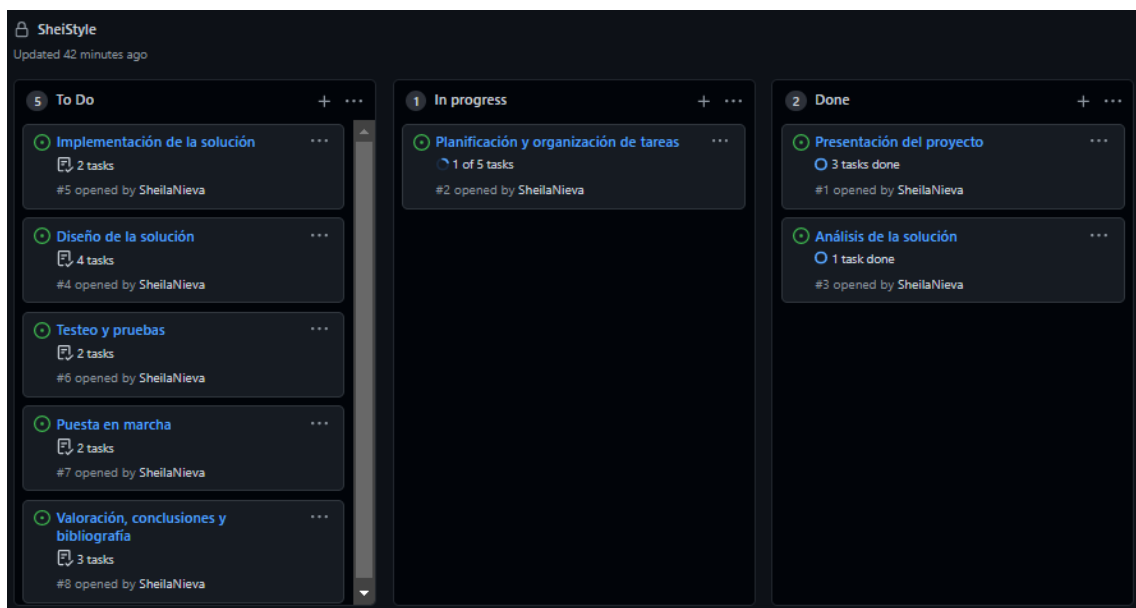
Todo proyecto requiere una organización previa, para poder tener un esquema claro de los pasos a seguir y el tiempo estimado a cada uno de ellos. La improvisación no suele ser un buen aliado en mucho de los casos si queremos cumplir con unos tiempos específicos.

Para la siguiente imagen, hemos utilizado la herramienta de Git 'TimeLine', la cual nos mostrará en una línea del tiempo las tareas (y subtareas que las componen) que hay que entregar y su fecha correspondiente de estimación



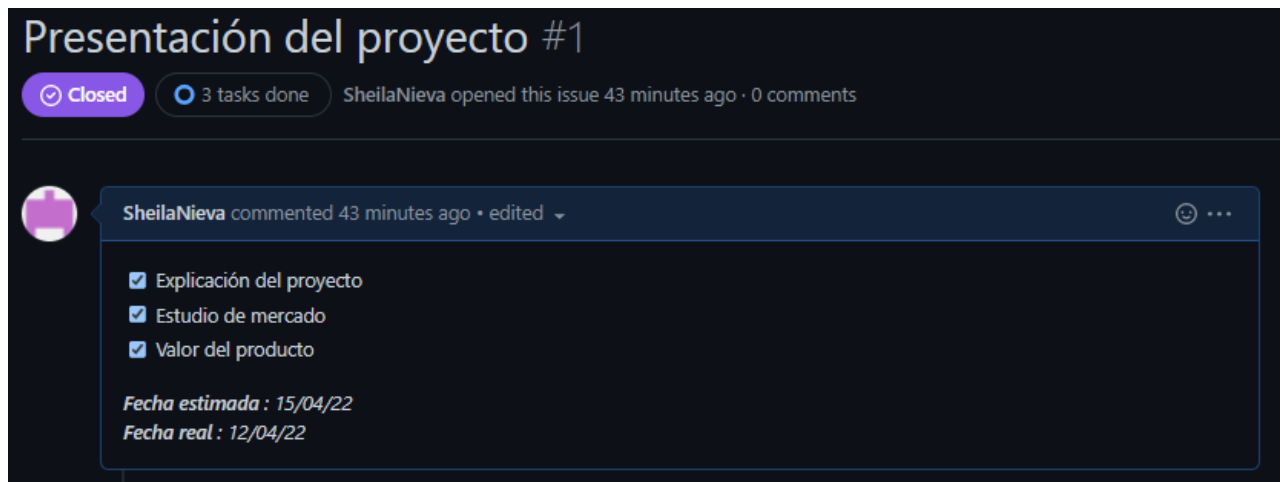
A continuación, vamos a mostrar cómo hemos organizado nuestras tareas en un Kanban.

De esta manera, podemos ver de una forma más rápida todas las tareas que debemos realizar y en qué estado se encuentran actualmente.



En el siguiente paso vamos a explicar las subtareas que componen dichas issues:

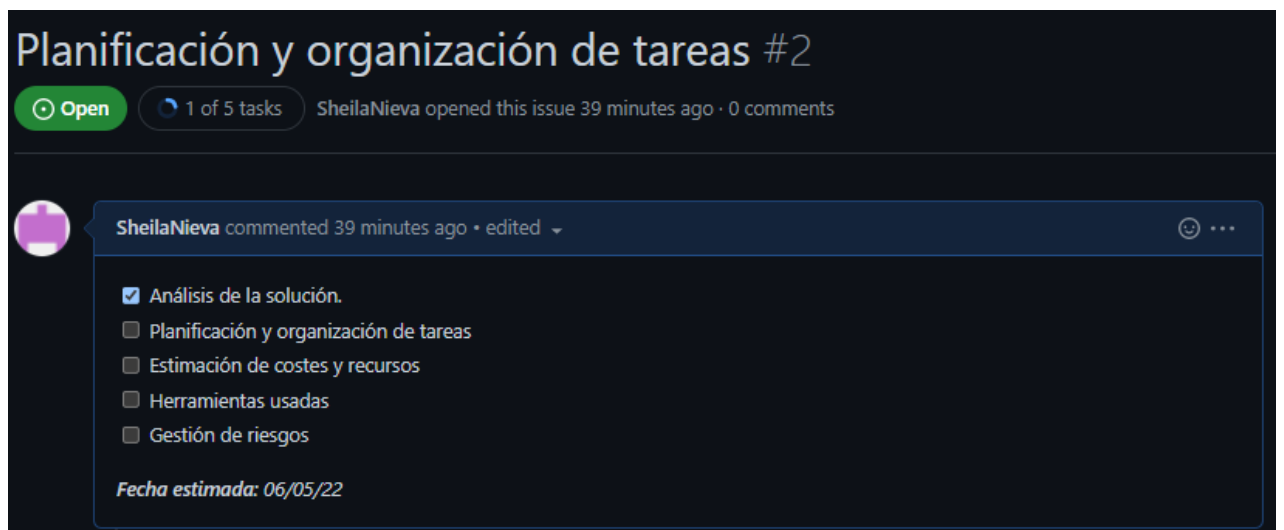
1. Presentación del proyecto



The screenshot shows a Jira issue titled "Presentación del proyecto #1". The status is "Closed" (indicated by a purple circle with a checkmark). It shows "3 tasks done" and "SheilaNieva opened this issue 43 minutes ago · 0 comments". The issue content, commented by SheilaNieva 43 minutes ago, includes a checklist of three items: "Explicación del proyecto", "Estudio de mercado", and "Valor del producto", all of which are checked. Below the checklist, the estimated date is "15/04/22" and the actual date is "12/04/22".

En esta tarea vamos a realizar una breve introducción de lo que tratará nuestra aplicación de escritorio, haremos un estudio de mercado y el sacaremos una conclusión sobre el valor del producto

2. Planificación de tareas y estimación de costes




The screenshot shows a Jira issue titled "Planificación y organización de tareas #2". The status is "Open" (indicated by a green circle with a checkmark). It shows "1 of 5 tasks" and "SheilaNieva opened this issue 39 minutes ago · 0 comments". The issue content, commented by SheilaNieva 39 minutes ago, includes a checklist of five items: "Análisis de la solución." (checked), "Planificación y organización de tareas" (unchecked), "Estimación de costes y recursos" (unchecked), "Herramientas usadas" (unchecked), and "Gestión de riesgos" (unchecked). Below the checklist, the estimated date is "06/05/22".

En esta tarea vamos a realizar un análisis de la solución, la planificación y organización de tareas, la estimación de los costes y recursos. A parte de esto, nombraremos las herramientas usadas y hablaremos de la gestión de riesgos.

3. Análisis de la solución

Análisis de la solución #3

Closed 1 task done SheilaNieva opened this issue 35 minutes ago · 0 comments



SheilaNieva commented 35 minutes ago · edited


☒ Análisis de escenario; casos de uso
Fecha estimada: 22/04/22
Fecha real: 20/04/22

En esta tarea vamos a realizar un análisis de los casos de uso.

4. Diseño de la solución

Diseño de la solución #4

Open 4 tasks SheilaNieva opened this issue 29 minutes ago · 0 comments

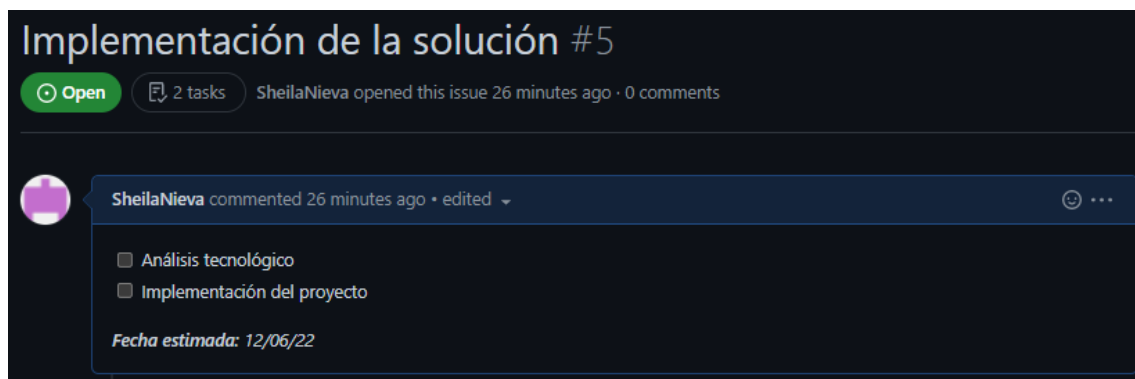


SheilaNieva commented 29 minutes ago

☐ Diseño de la interfaz, usuarios y prototipos
☐ Diagrama de clases
☐ Diseño persistencia de datos
☐ Arquitectura del sistema
Fecha estimada: 23/05/22

En esta tarea vamos a realizar el diseño de la interfaz de usuario, el diagrama de clases, el diseño de la persistencia de datos y el diseño de la arquitectura del sistema

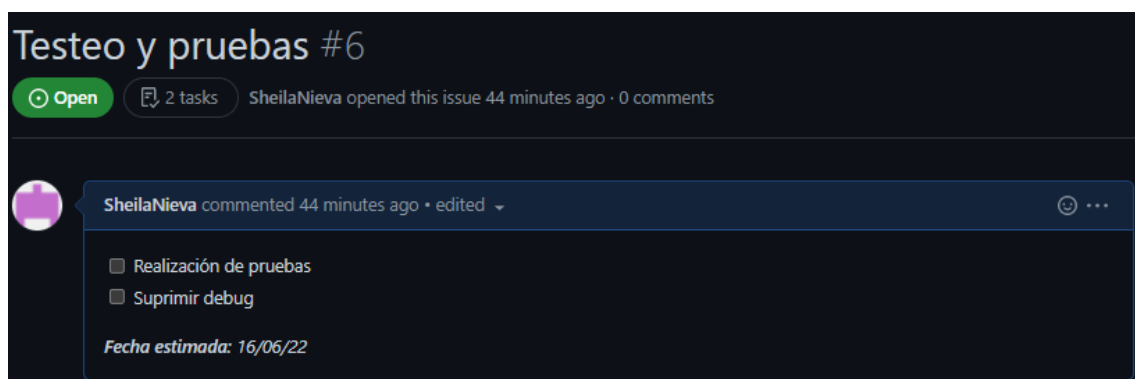
5. Implementación de la solución



The screenshot shows a GitHub issue titled "Implementación de la solución #5". At the top, it says "Open" in a green circle, "2 tasks", and "SheilaNieva opened this issue 26 minutes ago · 0 comments". Below this is a comment by SheilaNieva, made 26 minutes ago, which includes a checklist with two items: "Análisis tecnológico" and "Implementación del proyecto". At the bottom of the comment, it says "Fecha estimada: 12/06/22".

Aquí vamos hablar del análisis tecnológico y procederemos a la implementación del proyecto.

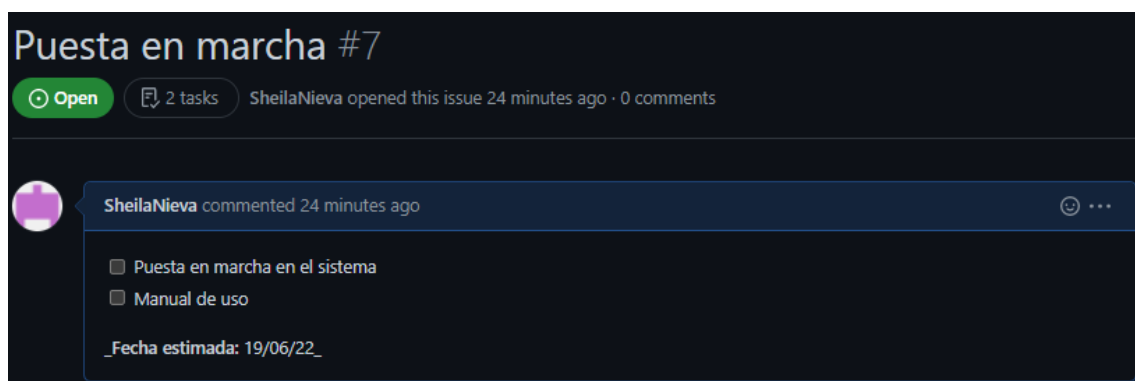
6. Testeo y pruebas



The screenshot shows a GitHub issue titled "Testeo y pruebas #6". At the top, it says "Open" in a green circle, "2 tasks", and "SheilaNieva opened this issue 44 minutes ago · 0 comments". Below this is a comment by SheilaNieva, made 44 minutes ago, which includes a checklist with two items: "Realización de pruebas" and "Suprimir debug". At the bottom of the comment, it says "Fecha estimada: 16/06/22".

Realizaremos una clase de pruebas que nos ayudarán a depurar el proyecto y buscaremos solución a todo tipo de fallos que podamos encontrar.

7. Puesta en marcha



The screenshot shows a GitHub issue titled "Puesta en marcha #7". At the top, it says "Open" in a green circle, "2 tasks", and "SheilaNieva opened this issue 24 minutes ago · 0 comments". Below this is a comment by SheilaNieva, made 24 minutes ago, which includes a checklist with two items: "Puesta en marcha en el sistema" and "Manual de uso". At the bottom of the comment, it says "Fecha estimada: 19/06/22".

Lanzaremos el proyecto con su manual de uso

8. Valoración, conclusiones y bibliografía

Valoración, conclusiones y bibliografía #8

Open 3 tasks SheilaNieva opened this issue 23 minutes ago · 0 comments

SheilaNieva commented 23 minutes ago

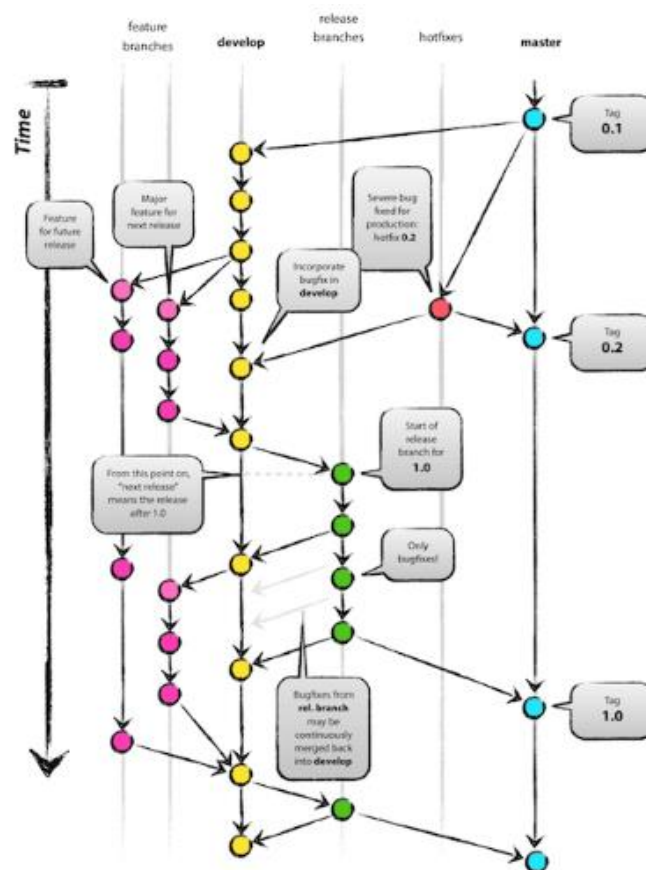
- ☐ Valorar aspectos de la aplicación
- ☐ Conclusión final del proyecto
- ☐ Bibliografía

Fecha estimada: 19/06/22

Esta será nuestra última tarea, donde en ella daremos una valoración y una conclusión general a todo el proyecto y realizaremos la bibliografía.

❖ Uso de GitFlow como herramienta de trabajo

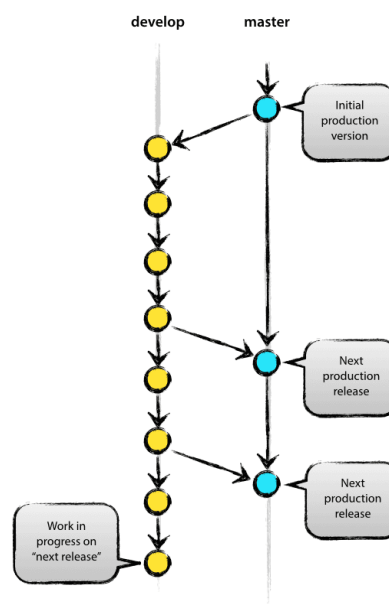
GitFlow es una estrategia creada para mejorar la organización de **Branches** (ramificaciones) dentro del repositorio y, de esta forma, dar más fluidez al proceso de nuevos Features y Releases. Vamos a tener dos tipos de ramas, las ramas principales y las ramas de apoyo.



- **Las ramas principales:**

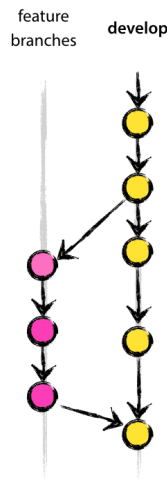
Todo proyecto, por defecto, debería tener al menos dos ramas infinitas para su desarrollo. Esta metodología define que deben existir dos ramas principales: master y develop.

- o Rama master: contiene cada una de las versiones estables del proyecto. Cualquier *commit* que subamos en esta rama debe estar preparado para que se pueda incluir en producción.
- o Rama develop: contiene el código de desarrollo de la siguiente versión planificada del proyecto. En ella se incluirán cada una de las nuevas características que se desarrollen. Esta rama puede incorporarse tanto en una rama *release* (que veremos más adelante) como en la rama *master*, para su despliegue en producción.



- **Las ramas de apoyo:**

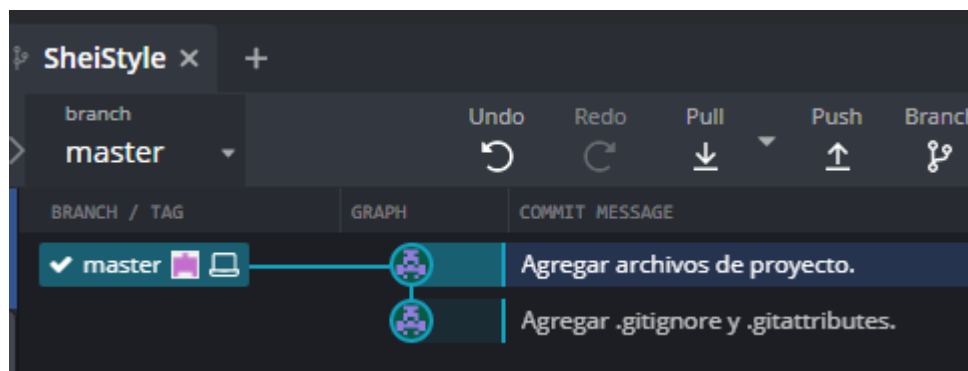
Su objetivo es el permitir el desarrollo en paralelo entre los miembros del equipo, la resolución de problemas en producción de forma rápida, etc. A diferencia de las ramas principales, estas están limitadas en tiempo. Serán eliminadas eventualmente. Un ejemplo de rama de apoyo es la rama *feature*:



En nuestro caso, usaremos la herramienta de GitKraken para hacer todo de manera visual en vez de por comandos.



En la siguiente imagen se muestra el árbol (vacío porque aún no hemos empezado con el código de nuestra aplicación), que poco a poco irá tomando forma con sus ramas y commit correspondientes y quedará algo parecido al que hemos mostrado anteriormente como ejemplo.



❖ Wiki del proyecto

Para finalizar, como herramienta para documentar el proyecto, utilizaré la Wiki de GitHub que nos permite añadir información acerca del proyecto en el propio repositorio. Esta herramienta que ofrece GitHub se trata de un espacio en el que podemos alejar la documentación, por lo que toda la documentación hecha y por hacer estará reflejada ahí.

<https://github.com/SheilaNieva/SheiStyleSNL/wiki>

3.2. Estimación de costes y recursos

La **estimación de costes** en gestión de proyectos es el proceso de prever los recursos financieros y otros necesarios para completar un proyecto dentro de un alcance definido. La estimación de costes tiene en cuenta cada elemento requerido para el proyecto, desde los materiales hasta la mano de obra, y calcula una cantidad total que determina el presupuesto de un proyecto. Una estimación de costes inicial puede determinar si una empresa da luz verde a un proyecto, y si el proyecto avanza, la estimación puede ser un factor en la definición del alcance del proyecto. Si la estimación de costes es demasiado alta, la empresa puede decidir reducir el proyecto en función de lo que pueda pagar. (También es necesario empezar a obtener financiación para el proyecto.) Una vez que el proyecto está en marcha, la estimación de costes se utiliza para gestionar todos sus costes asociados, a fin de que el proyecto se ajuste al presupuesto.

Para llevar a cabo **la estimación de recursos**, es necesario determinar las personas, equipos y/o materiales necesarios. Qué cantidad de cada recurso se utilizará y cuándo estarán disponibles dichos recursos.

Debido a que nuestra aplicación SheiStyle será para escritorio, si hablamos de la parte hardware, necesitaremos única y principalmente un ordenador para su uso.

Por otra parte, hablando del software, no incluiremos en la estimación de costes, gastos por la utilización de Firebase. Dado que Firebase, la versión gratuita, nos permite hacer una gran cantidad de peticiones y bastante espacio de almacenamiento. Es por eso, que teniendo en cuenta nuestro volumen de usuarios iniciales, nos decantaremos por esta opción gratuita que cumple con creces nuestra demanda. Tampoco incluiremos los gastos por el uso de Visual Studio, ya que también es una herramienta gratuita.



Ahora vamos a proceder a estimar los costes de nuestra aplicación. Para ello, solo necesitamos poner un precio a nuestra hora de trabajo y calcular las horas que le dedicaremos a dicho proyecto.

El precio de un programador junior en España oscila entre los 15€/h y los 35€/h. La media sería unos 25€/h. En mi caso el precio será 20€/h. Mi estimación de tiempo para el desarrollo de la aplicación son 90h, y sumándole las 30h dedicadas a documentación hacen un total de 120h.

ESTIMACION DE COSTES		
HORAS DEDICADAS	PRECIO X HORA	TOTAL
120h	20€	2400€
TOTAL CON IVA (21%)		2904€

3.3. Herramientas usadas

Las herramientas que voy a utilizar son las siguientes:

Para el desarrollo de código como bien hemos nombrado antes, vamos a usar **Visual Studio**, que es un entorno de desarrollo integrado (IDE). Incluye depuración, control integrado con Git y se puede extender con diversas extensiones para soportar los lenguajes que necesitemos.



Como también hemos nombrado antes en la estimación de recursos, para el almacenamiento y alojamiento de datos usaremos **Firebase**



Para todas las tareas de documentación utilizaremos **Office Professional Plus 2016**, un conjunto de programas que incluyen Word, Excel, Outlook y demás herramientas de sobra conocidas.



Para el control de código fuente utilizaremos **Git y GitHub**. Para gestionar estas herramientas utilizaremos **GitKraken**, que además incorpora GitKraken boards, que es un gestor de incidencias a través de un tablero Kanban. Esto nos ayudará a descomponer el proyecto en pequeñas tareas y poder planificarlas todas de forma muy ordenada.



Además, utilizaré **draw.io** para realizar los diagramas necesarios y **Figma** para el prototipado de la interfaz de la aplicación (ambas herramientas gratuitas).



3.4. Gestión de riesgos

Riesgos laborales

Se entiende como riesgo laboral a los peligros existentes en una profesión y tarea profesional concreta, así como en el entorno o lugar de trabajo, susceptibles de originar accidentes o cualquier tipo de siniestros que puedan provocar algún daño o problema de salud tanto físico como psicológico. El riesgo laboral se denominará grave o inminente cuando la posibilidad de que se materialice en un accidente de trabajo es alta y las consecuencias presumiblemente severas o importantes.

Para estos, existe una medida denominada Prevención de Riesgos Laborales, consiste en un conjunto de medidas y actividades que se realizan en las empresas para detectar las situaciones de riesgos e implementar las medidas necesarias para eliminarlas o minimizar sus efectos. Se trata también de un conjunto de técnicas orientadas a reconocer, evaluar y controlar los riesgos ambientales que pueden ocasionar accidentes y/o enfermedades profesionales.

En nuestro caso, existen algunos riesgos que podrían darse más a menudo debido a nuestro sector de trabajo como es la informática.

- **Fatiga visual:** Pueden aparecer molestias oculares por el uso de pantallas de visualización.

Solución: tener una colocación ergonómica de la pantalla (mínimo 40cm y debe estar inclinada ligeramente para que el enfoque sea perpendicular a nuestro ángulo de visión) y contar con una luz adecuada (preferiblemente natural)

- **Fatiga muscular:** es producida por posturas incorrectas y por mantener una misma posición en un tiempo prolongado.

Solución: prevenir posturas forzadas o inadecuadas

- **Caídas de personal o golpe contra objetos**

Solución: mantener los lugares de trabajos limpios y ordenados, dejando libres de obstáculos las zonas de paso.

- **Contacto eléctrico**

Solución: respetar normas de seguridad y proceder a revisar el estado de cada equipo antes de su uso. En caso de detectar alguna anomalía, comunicarlo cuanto antes.

- **Carga mental:** producida por el estrés del trabajo o la desmotivación.

Solución: realizar tareas variadas, paradas periódicas para prevenir la fatiga, seguir hábitos de vida saludable y realizar ejercicio de forma habitual.

- **Incendio en nuestra área de trabajo**

Solución: mantener limpio de papeles y material de cartón nuestra mesa de trabajo, ser organizados y apagar los equipos al final de cada jornada de trabajo.

Riesgos específicos de proyectos software

La gestión de riesgos es una actividad de protección dentro de la gestión de proyectos, encargada de identificar, mitigar y monitorizar los riesgos que pudieran afectar a la ejecución y viabilidad del proyecto. Un riesgo es cualquier acontecimiento futuro que pueda afectar de forma negativa o positiva a un proyecto. No todos los riesgos tienen la misma probabilidad de ocurrir, ni el mismo impacto, por lo que a la hora de tratarlos nos preocuparemos de solucionar primero los más problemáticos.

Existen varios tipos de riesgo como son:

- Los riesgos del proyecto. Ponen en peligro al plan. Si estos se cumplen, el proyecto requerirá mayor esfuerzo y dinero. Algunos de ellos pueden ser: el presupuesto, la planificación, el personal, los recursos y los requisitos.

- Los riesgos técnicos. Ponen en peligro la calidad resultante. Si estos se cumplen, el proyecto es más complejo de lo estimado. Algunos de ellos pueden ser: los requisitos, el diseño, la implementación, la interfaz, el mantenimiento, las tecnologías desconocidas...

- Los riesgos del negocio. Ponen en peligro la realización del proyecto. Si estos se cumplen, el proyecto se cancelará. Algunos de ellos pueden ser: la utilidad, la dificultad de venta, el presupuesto...

A continuación, vamos a nombrar de manera organizada algunos riesgos y a implementar una posible solución a ellos:

- o **Elaboración de la planificación**

- Planificación optimista en vez de realista

Solución: revisar de nuevo la planificación y hacer de manera realista

- Presión excesiva en la planificación reduce la productividad.

Solución: dar un poco más de margen de tiempo en las entregas para ir más desahogados, pero sin dejar de lado nuestra organización.

- Un retraso en una tarea produce retrasos en cascada en las tareas dependientes

Solución: Se priorizará sobre aquellas partes funcionales con tal de dar cabida al funcionamiento principal de la aplicación.

- **Organización y gestión**

- El proyecto cuenta con un solo empleado lo que dificulta su desempeño

Solución: el empleado priorizará las tareas fundamentales para el funcionamiento básico de la aplicación, y en caso de que se pueda se añadirá un nuevo miembro que le servirá de apoyo a la hora de desarrollar.

- La falta de experiencia organizando proyectos de envergadura puede dar lugar a una falla en la organización de este.

Solución: apoyarnos en todas las herramientas posibles para evitar que esto ocurra.

- **Ambiente/Infraestructura de desarrollo**

- Las herramientas de desarrollo no funcionan como se esperaba
- El personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas
- La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado

Solución conjunta: organizarse de la mejor manera posible, dándole prioridad a las tareas fundamentales para el funcionamiento básico de la aplicación y conseguir que lo que se entrega tenga funcionalidad.

- **Producto**

- Utilizar lo último en informática alarga la planificación de forma impredecible.
- Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación.
- El trabajo con un entorno software desconocido causa problemas no previstos.

Solución conjunta: priorizar siempre las tareas fundamentales para el funcionamiento básico de la aplicación.

- **Personal**

- La falta de conocimiento de la metodología.

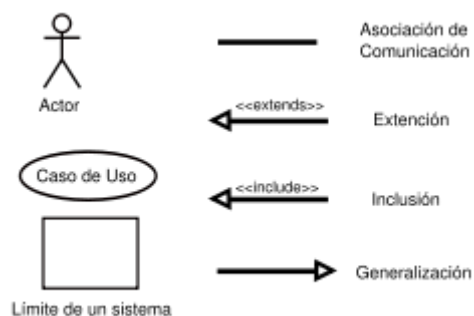
Solución: informarnos bien antes del proyecto sobre las herramientas a implementar.

- Pérdida de información: puede darse el caso de que se pierda código o algún archivo importante de nuestro proyecto.

Solución: utilizar herramientas como GitHub para ir subiendo todo el código poco a poco y organizarnos por tareas de pequeñas.

4. Análisis de escenario

4.1. Casos de uso



Un caso de uso es la descripción de una acción o actividad. Un diagrama de caso de uso es una descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso. Los personajes o entidades que participarán en un diagrama de caso de uso se denominan actores. En el contexto de ingeniería del software, un diagrama de caso de uso representa a un sistema o subsistema como un conjunto de interacciones que se desarrollarán entre casos de uso y entre estos y sus actores en respuesta a un evento que inicia un actor principal.

Existen tres elementos principales:

- **Actor:** en nuestro caso será el administrador que interactúa con nuestra aplicación. Se representa con una figura humana esquemática.

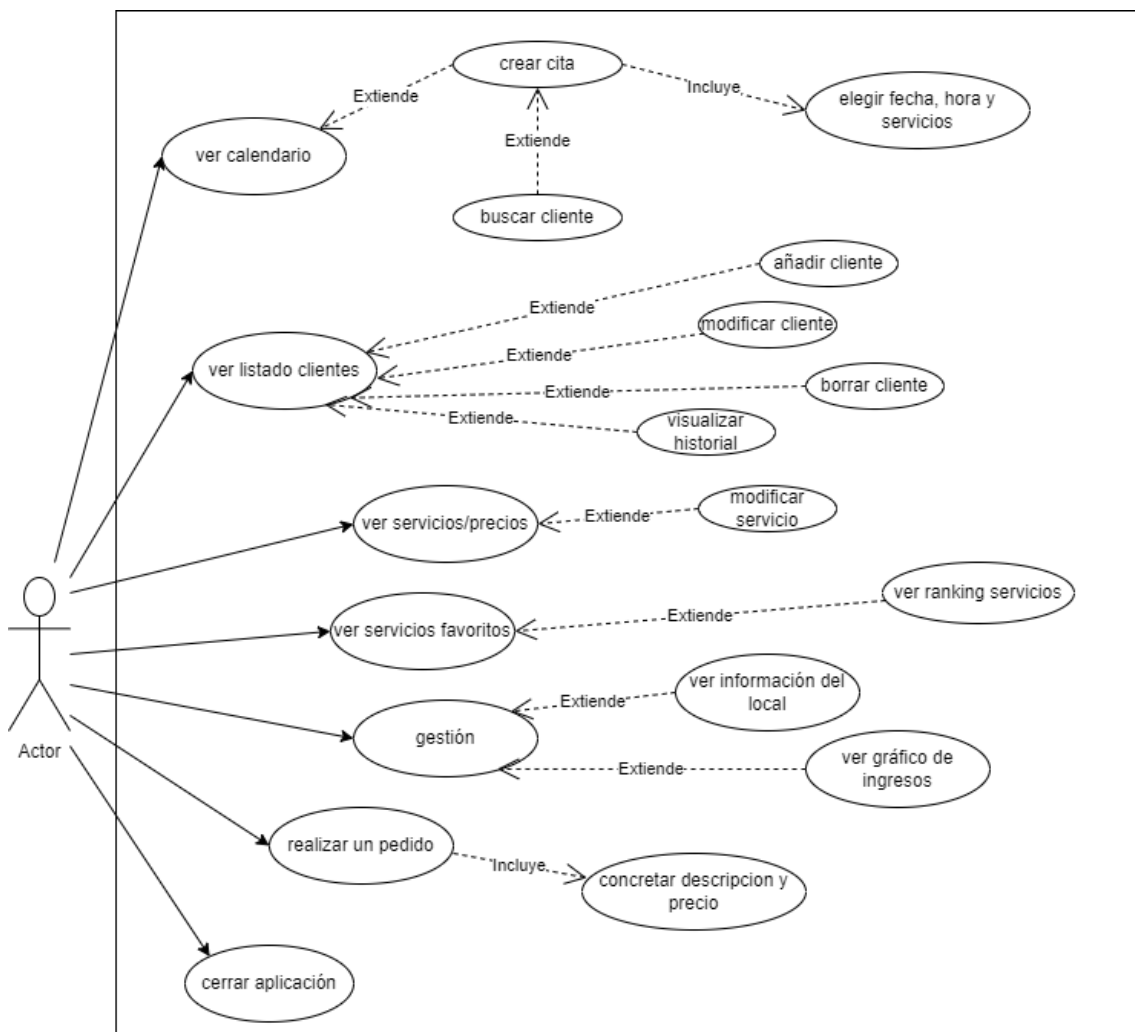
- **Sistema:** en nuestro caso será el Software de nuestro local. Se representa con un rectángulo

- **Caso de uso:** constituye una tarea o proceso dentro de nuestra aplicación. Se muestra en forma de elipse

La relación de estos elementos, que formarán nuestros casos de uso a continuación, se representa con unas líneas de conexión llamadas asociaciones.

- Una **línea recta** entre el actor y el caso de uso evidencia que el actor y el caso de uso descrito están relacionados.

- Una **línea discontinua** establece una relación entre diferentes casos de uso



4.2. Diseño de la interfaz y prototipos

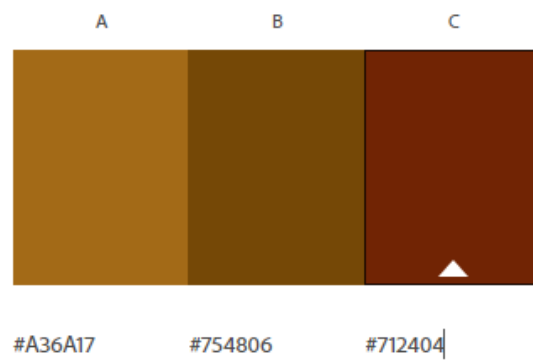
La interfaz de usuario es el medio con el que el usuario puede comunicarse con nuestra aplicación. Normalmente la interfaz de usuario destaca por ser fácil de entender e intuitiva, es por eso que hemos utilizado en la aplicación de SheiStyle, una sección principal en la que hemos expuesto las diferentes acciones de la aplicación en una misma pantalla. También hemos añadido un menú en la parte de arriba a la derecha. Por último, la opción de cerrar la aplicación aparecerá abajo a la derecha.

Cada acción la hemos asociado a un botón, que se mostrará con el título y un icono referente a la acción para que sea altamente intuitiva para todo tipo de usuarios. A continuación, iremos detallando las características de cada una de las pantallas que vamos a ir encontrando con el uso de la aplicación.

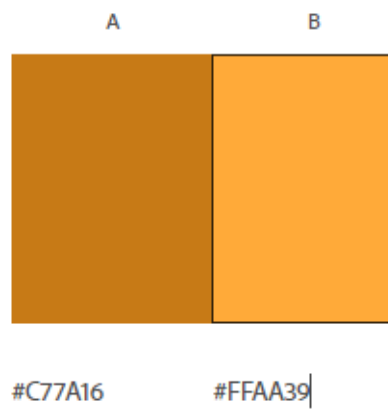
Otro aspecto fundamental en el desarrollo de las interfaces de usuario son los colores a utilizar. Con el objetivo de generar una aplicación amigable hemos utilizado dos colores principales; un naranja amarillento y un marrón oscuro. También aparecerán colores secundarios que serán complementarios a los principales y los básicos como son el blanco y el negro.

Aquí mostramos los tonos y sus referencias:

El marrón que usaremos como principal tiene de referencia: #712404

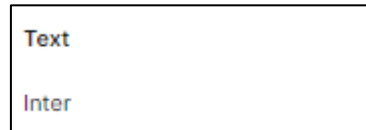


El naranja que usaremos como principal tiene de referencia: #FFAA39



Ahora vamos a proceder a mostrar las pantallas que hemos generado en el prototipado (usando figma) y explicar su función en cada una de ellas.

Por concretar algo más del diseño de la aplicación, hemos usado el estilo de fuente Inter.

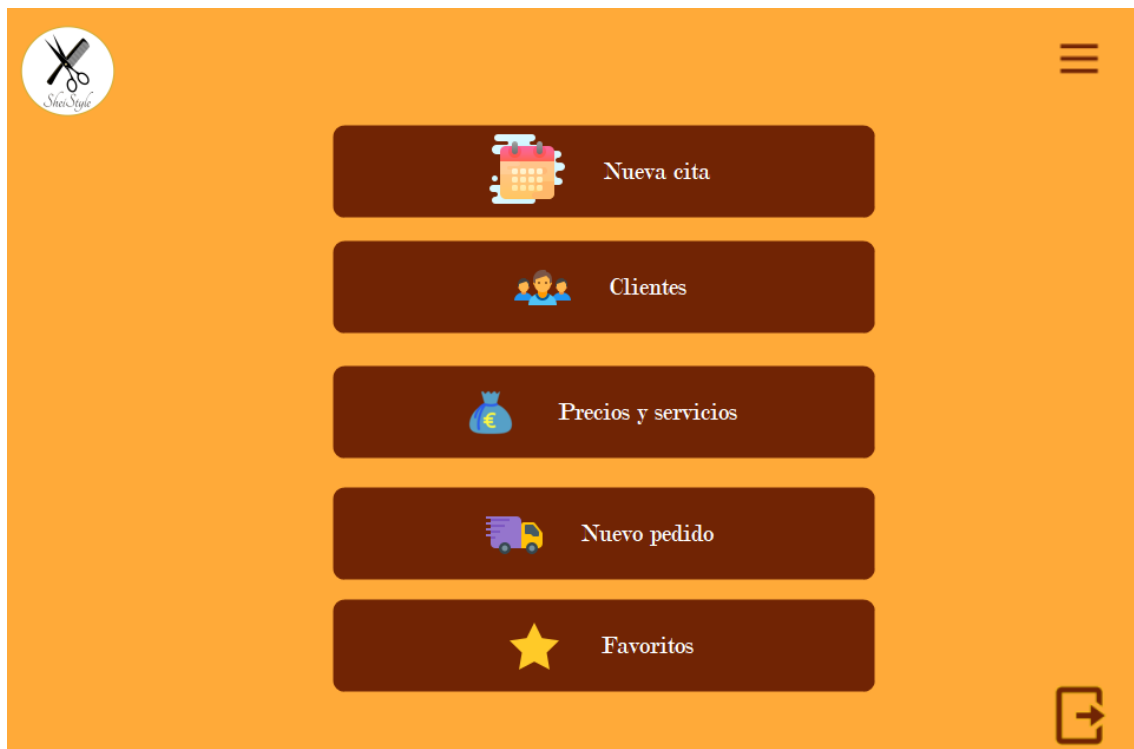


Como nuestra aplicación está instalada en el equipo del salón de belleza y contamos con que solo pueda acceder quien tenga permiso para la gestión de citas, nuestra aplicación no va a necesitar ningún tipo de identificación ni logueo.

Lo primero que se nos mostrará, será una pantalla con nuestro logo con una duración concreta:



Nuestra pantalla principal será la siguiente:



Consta de 5 botones principales como son el calendario, clientes, servicios y precios, nuevo pedido y favoritos.

También contaremos con un menú donde podremos ver información sobre la empresa.

Y por último la opción de salir de la aplicación.

Si le damos al botón de calendario, nos llevará a la siguiente ventana:



Aquí deberemos elegir un mes y un día en concreto para poder coger una cita, después le daremos al botón aceptar y nos llevará a otra ventana para poder elegir la hora.

Agenda

Elija una hora

Fecha: 17 de Mayo de 2022

08.00
08.30
09.00 Laura Gómez
09.30
10.00
10.30
11.00 Eva Jimenez
11.30
12.00 Luci...
12.30
13.00
13.30
14.00
14.30

Aceptar

En esta pantalla se nos mostrará un tipo de agenda en la cual podremos seleccionar alguna hora que esté libre y apuntar el nombre del cliente.

En caso que el cliente sea nuevo, debemos darle de alta antes en la sección de clientes.

Si el usuario ya tiene ficha en nuestro local porque ya haya visitado nuestro local, podremos buscarlo en el buscador, seleccionar su nombre y dale al botón aceptar para que nos lleve a la siguiente ventana.

Calendario

Peluquerías

- ☒ Lavado
- ☐ Decoloración
- ☒ Corte
- ☐ Alisado brasileño
- ☐ Tinte
- ☐ Permanente
- ☒ Mechas
- ☐ Recogido

Uñas

- ☒ Permanentes
- ☐ Pedicura
- ☐ Acrílicas
- ☐ De gel

Depilación con cera caliente

- ☐ Bigote
- ☐ Cejas

Láser

- ☐ Piernas
- ☐ Axilas
- ☐ Brazos
- ☐ Zona íntima

Barbería

- ☐ Afeitado
- ☐ Recorte

Fecha: 17 de Mayo de 2022 Hora: 12:00

Aceptar

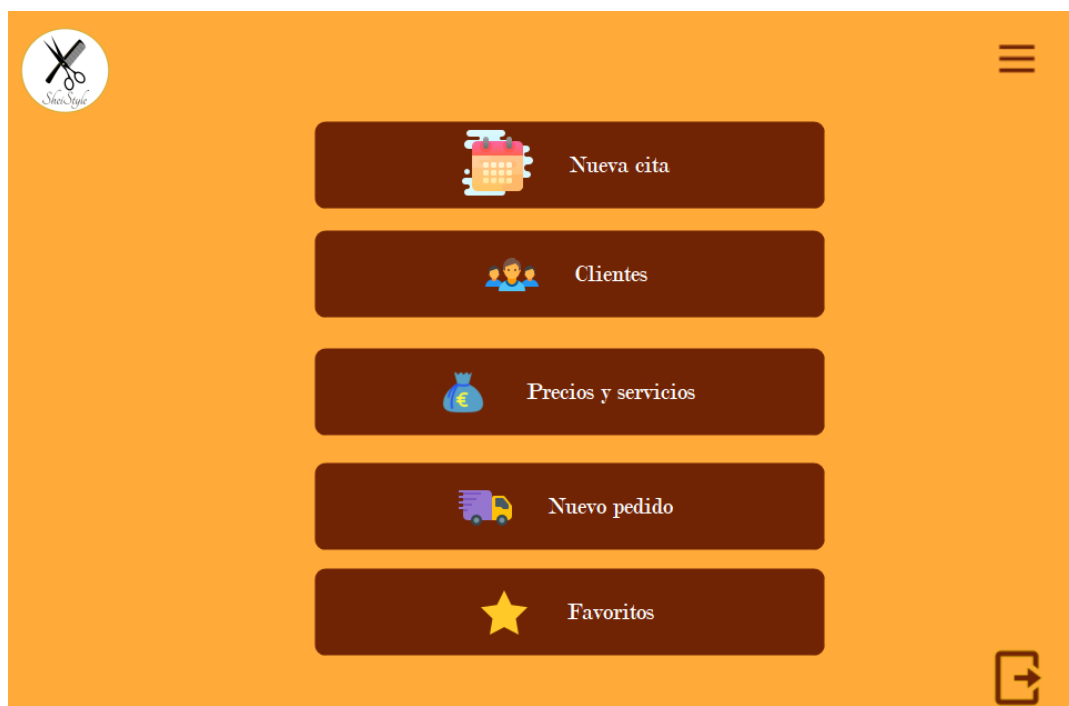
Esta ventana es el siguiente paso para reservar una cita. En ella se nos mostrará todos los servicios que puede consumir en nuestro local distribuidos por sectores. También se nos mostrará los datos recogidos anteriormente, como son la fecha y la hora de la cita que queremos reservar. Una vez elegidos los servicios, pulsaremos el botón aceptar para que nos lleve a la última página y poder guardar nuestra cita.



Esta pantalla nos mostrará el precio de la cita, sumando automáticamente el precio establecido a cada servicio que deseamos consumir. La primera parte nos calcula el precio real, la segunda nos indicará si contamos con algún porcentaje de descuento y la tercera el total a pagar, disminuyendo el presupuesto en caso de contar con un algún tipo de descuento. Esto se conseguirá cada vez que un cliente, registre un múltiplo de 3 citas en nuestro local.


Por último, para concluir con los pasos de la cita, deberemos pulsar el botón de reservar cita.

Si volvemos a la pantalla principal, la segunda opción que hay es la de clientes



Nos llevará a la siguiente pantalla:

← **Listado clientes** 

 Buscar por nombre ▼ **Buscar**

Nombre: Eva	Apellidos: Jimenez Montoya	Móvil: 666777555

Añadir cliente **Editar cliente** **Eliminar cliente** **Mostrar citas**

Aquí podremos ver un listado de todos los clientes registrados en nuestro local. Contamos también con un filtro, como es la búsqueda por nombre.

Tenemos 4 opciones principales:

-**Eliminar cliente:** seleccionando un cliente y dándole a dicho botón comprobaremos si tiene citas pendientes y procederemos a borrarlo en caso de que así lo deseemos.

-**Añadir cliente:** nos mostrará la siguiente pantalla:

← **Nuevo Cliente** 

Introduzca los siguientes datos para registrar un nuevo cliente:

Nombre

Apellidos

Teléfono

Correo

Añadir

Aquí debemos introducir los datos del nuevo cliente, pudiéndolo así registrar en nuestra base de datos.

-Editar cliente: si seleccionamos un cliente y pulsamos este botón, nos mostrara la siguiente ventana:



Nos cargará los datos del cliente que hayamos seleccionado anteriormente y podremos editar alguno de ellos. Para guardar los cambios simplemente tendremos que pulsar el botón de actualizar cliente.

-Mostrar citas: si seleccionamos un cliente y después pulsamos este botón, podremos observar el historial de citas en nuestro local.



Fecha	Hora	Servicios	Precio
17/05/22	12:00	Lavado, corte, mechas, permanentes	38€

Esta ventana de listado de citas, a su vez, contará con otras 2 opciones principales:

-**Eliminar cita:** podremos eliminar una cita seleccionada.

-**Modificar servicios:** podremos editar los servicios de una cita en caso de que el cliente haya cambiado de opinión y desee consumir más o menos de estos. Para ello, la aplicación nos llevará a la siguiente ventana:

Modificar servicios

Peluquería

- ☒ Lavado
- ☐ Decoloración
- ☒ Corte
- ☐ Alisado brasileño
- ☐ Tinte
- ☐ Permanente
- ☒ Mechas
- ☐ Recogido

Depilación con cera caliente

- ☐ Bigote
- ☐ Cejas

Láser

- ☐ Piernas
- ☐ Axilas
- ☐ Brazos
- ☐ Zona íntima

Uñas

- ☒ Permanentes
- ☐ Pedicura
- ☐ Acrílicas
- ☐ De gel

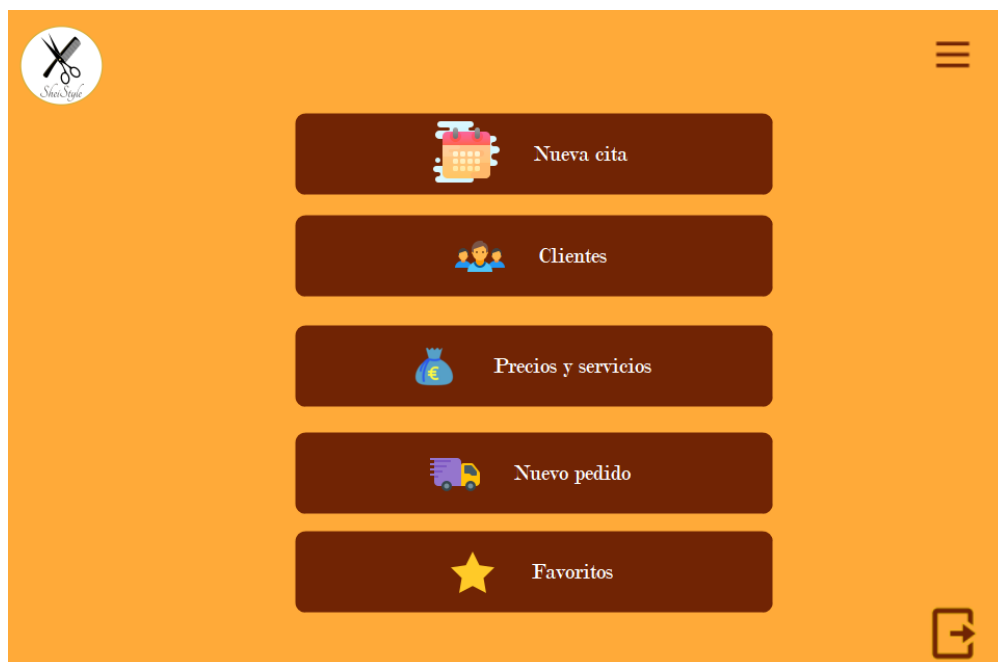
Barbería

- ☐ Afeitado
- ☐ Recorte

Fecha: 17 de Mayo de 2022 Hora: 12:00 Actualizar

Aquí podremos seleccionar de nuevo los servicios deseados y guardarlos en la misma cita (fecha y hora) que teníamos reservada anteriormente. Para guardar los cambios solo tenemos que pulsar el botón aceptar.

Volviendo a la pantalla principal:



Como tercera opción, tenemos el apartado de Servicios y precios que nos llevará a la siguiente ventana:

Peluquería			
Lavado	10€	Decoloración	25€
Corte	12€	Alisado brasileño	96€
Tinte	28€	Permanente	30€
Mechas	25€	Recogido	18€

Depilación con cera caliente	
Bigote	3€
Cejas	18€

Láser (por cada sesión)	
Piernas	30€
Axilas	15€
Brazos	25€
Zona íntima	30€

Uñas	
Permanentes	15€
Pedicura	15€
Acrílicas	22€
De gel	20€

Barbería	
Afeitado	8€
Recorte	10€

Modificar precio

En esta se nos mostrará el listado de servicios que tenemos disponibles en nuestro local con su precio correspondiente, distribuidos por sectores.

Podremos modificar el precio de cada uno de ellos en caso de que sea necesario. Para ello, debemos seleccionar un servicio y pinchar el botón modificar precio, que nos llevará a la siguiente ventana:

Modificar precio

Servicio: Corte

Sector: Peluquería

Precio: 12 €

Añadir

Esta recogerá el nombre del servicio que hayamos seleccionado y el sector. Podremos volver a establecer un precio y guardarlo dándole simplemente al botón añadir.

De nuevo, si volvemos a la pantalla principal:



La última opción es la de favoritos, llevándonos a la siguiente ventana:



En esta ventana, podremos ver de una manera muy clara los servicios de nuestra empresa y el porcentaje que presentan según las citas de nuestros clientes. Por decirlo de otra manera, se muestra como un ranking de los servicios más consumidos.

En nuestra ventana principal, por último, tenemos la opción del menú:



Pinchando en el menú nos llevará a la siguiente ventana:



Esta, a su vez, contará con 2 opciones que se cargarán en el cuadro de su derecha según las pinchemos.

-**Datos SheiStyle:** mostrará la información más importante de nuestro local, como es el nombre, la dirección, el teléfono y el correo.



Nombre	Dirección	Teléfono	Correo
SheiStyle	Avenida Andalucía, nº 0603	900-56-89-24	sheistyle@atencioncliente.es

-**Ingresos:** mostrará una gráfica de barras con los ingresos de cada mes. Como por ejemplo esta:

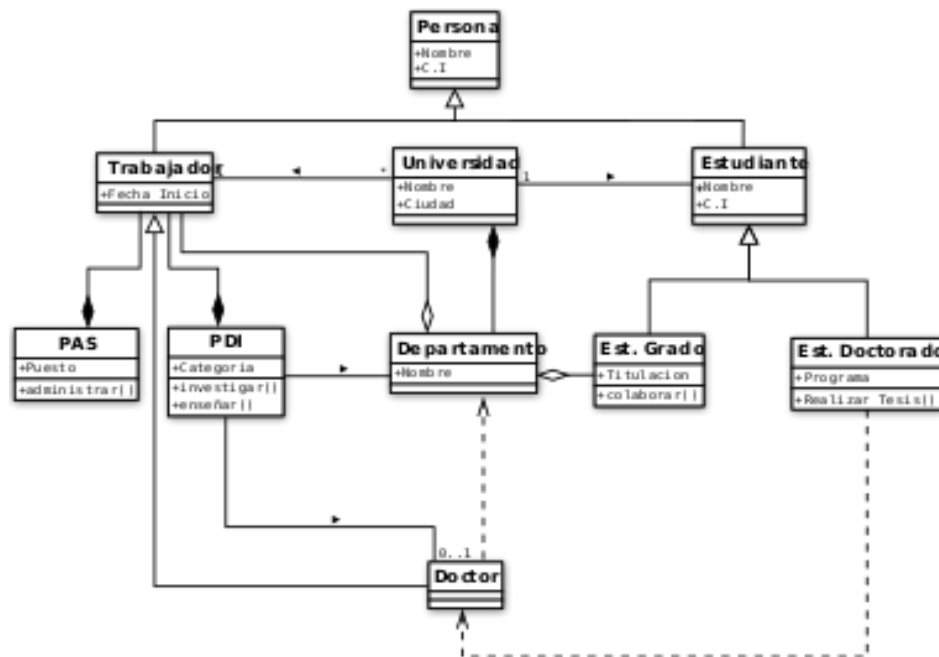


Para ver el funcionamiento de una manera más dinámica y rápida, podemos visitar el link donde explico en un video de youtube el prototipado:

<https://youtu.be/jq4KrEpmV3E>

4.3. Diagrama de clases

Un diagrama de clases en Lenguaje Unificado de Modelado (UML) es un tipo de estructura estática que describe la estructura de un sistema mostrando las clases de este, sus atributos, operaciones (o métodos) y las relaciones entre los objetos.



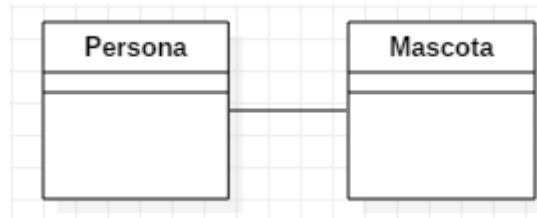
Ejemplo de diagrama de clases

En primer lugar, vamos a describir estos conceptos:

- Una **clase** es una plantilla para la creación de un objeto de datos según un modelo predefinido. Estas se utilizan para representar las entidades o conceptos de la aplicación.
- Los **objetos** son las instancias de una clase.
- Los **atributos** serán las características individuales que diferencian a un objeto de otro y determinan sus cualidades.
- Un **método** es una subrutina cuya operación está definida en una clase y puede pertenecer tanto a una clase, como a un objeto.

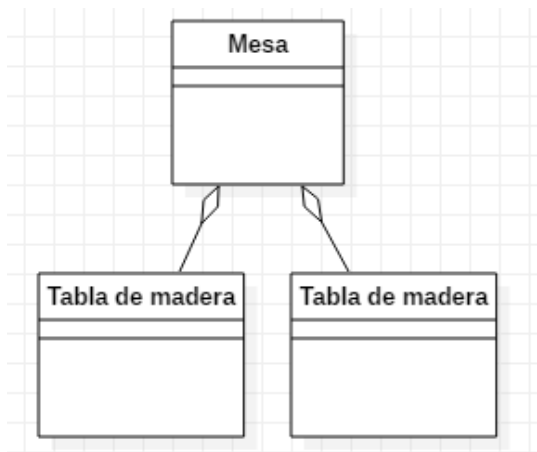
Existen varios tipos de relaciones:

-**Asociación:** es una relación estructural que describe una conexión entre objetos.



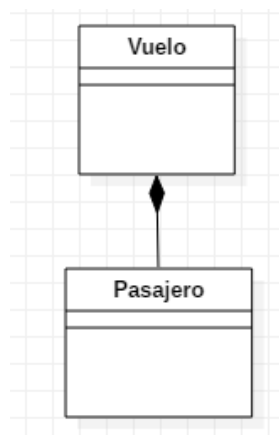
Ejemplo de asociación

-**Agregación:** indica a un objeto y las partes que lo componen. (Un objeto es parte de otro)



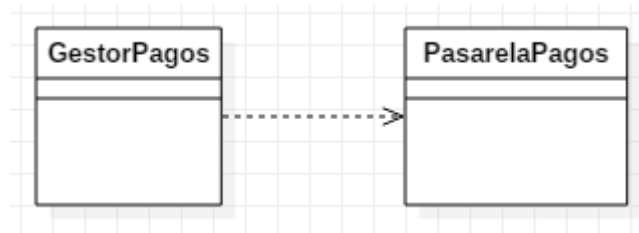
Ejemplo de agregación

-**Composición:** agregación disjunta y estricta: las partes solo existen asociadas al compuesto, se accede a ellas a través del compuesto.



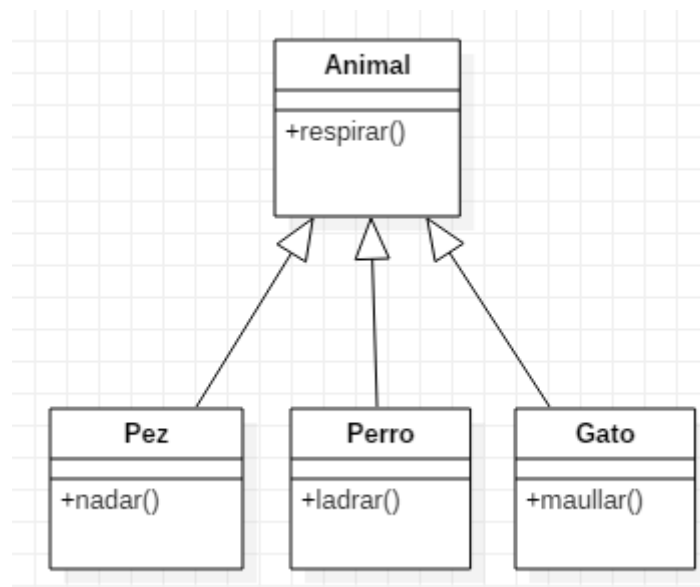
Ejemplo de composición

-**Dependencia:** representa que una clase requiere de otra para ofrecer sus funcionalidades.



Ejemplo de dependencia

-**Herencia:** permite que una clase (clase hija o subclase) reciba los atributos y métodos de otra clase (clase padre o superclase). Estos atributos y métodos recibidos se suman a los que la clase tiene por sí misma.



Ejemplo de herencia

En la siguiente imagen se muestra el diagrama de clases y las relaciones entre ellas de SheiStyle.



Como podemos observar, las relaciones que tiene la aplicación de SheiStyle son tanto de composición como de asociación.

Ahora procedemos a explicar una por una la naturaleza de estas relaciones.

Gastos ↔ Empresa

En SheiStyle, el tipo de relación que unen Gastos y Empresa es del tipo composición.

Los gastos pertenecen a una empresa y una empresa puede tener 0 o varios gastos.

Empresa ↔ Cliente

Es de tipo composición la relación que une Empresa y Cliente.

Una empresa puede tener 0 o varios clientes; y un cliente pertenece a una sola empresa.

Cliente ↔ Cita

El tipo de relación que une Cliente con Cita es de tipo composición.

Un cliente puede tener 1 o varias citas; y una cita solo puede pertenecer a un cliente.

Cita ↔ Servicio

El tipo de relación entre Servicio y Cita es de tipo asociación.

Debido a que, si un servicio es eliminado, la cita sigue estando almacenada en la base de datos. Lo mismo pasaría en caso contrario, si una cita se elimina, el servicio sigue existiendo.

Servicio ↔ Sector

El tipo de relación entre Servicio y Sector es de tipo composición.

Un servicio pertenece únicamente a un sector, pero un sector puede tener 1 o varios servicios a su vez.

Ingresos ↔ Cita

El tipo de relación entre Ingresos y Cita es de tipo composición.

Un ingreso pertenece únicamente a una cita.

4.4. Diseño de la persistencia de la información

Podemos entender 'persistencia' desde varios enfoques. En informática, la persistencia se refiere a la propiedad de los datos para que estos sobrevivan de alguna manera.

En programación, la persistencia es la acción de preservar la información de un objeto de forma permanente (guardado), pero a su vez también se refiere a poder recuperar la información del mismo (leerlo) para que pueda ser nuevamente utilizado.

De forma sencilla, puede entenderse que los datos tienen una duración efímera; desde el momento en que estos cambian de valor se considera que no hay persistencia de los mismos. Sin embargo, en informática hay varios ámbitos donde se aplica y se entiende la persistencia.

Para el desarrollo de SheiStyle, distintos factores me han llevado a elegir la opción de utilizar la **base de datos NoSQL** de Google Firebase.

❖ Base de datos NoSQL

Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala.

NoSQL: (Not Only SQL) plantea datos específicos de esquemas flexibles que se adaptan a los requisitos de las aplicaciones más modernas.

Las bases de datos NoSQL tienen muchas ventajas que nos interesan, entre ellas:

- Estos sistemas responden a las necesidades de escalabilidad horizontal que tienen cada vez más empresas.³
- Pueden manejar enormes cantidades de datos.
- No generan cuellos de botella.
- Escalamiento sencillo.
- Diferentes DBs NoSQL para diferentes proyectos.
- Se ejecutan en clusters de máquinas baratas.

Para terminar de entender bien el concepto de base de datos NoSQL, debemos atender a sus cuatro conceptos más importantes:

■ Bases de datos orientadas a documentos

Los datos se almacenan directamente en documentos de diferentes longitudes, sin ser necesario que los datos estén estructurados. Se les asigna distintos atributos sobre los cuales se puede rastrear los contenidos de los documentos. Son especialmente indicados para gestión de contenidos y

blogs. Utilizan JSON como formato de archivo ya que permite el intercambio de datos más rápido entre aplicaciones.

■ Bases de datos de grafos

Una base de datos de grafos forma relaciones entre datos mediante nodos. Así con contenidos de información muy entrelazada, ofrecen mejor rendimiento que las SQL relacionales. Se usan sobre todo en el ámbito de las redes sociales para representar relaciones entre seguidores.

■ Bases de datos clave-valor

Guardan los datos en pares clave-valor. Los valores individuales están asignados a claves específicas y el propio juego de datos funciona como clave y representa un valor. La clave crea un índice que se puede usar para buscar en la base de datos. Estas claves son unívocas y se pueden comparar a las Primary Key de las bases de datos relacionales.

■ Base de datos orientadas a columnas

A diferencia de SQL, no se guardan los registros en líneas, sino en columnas. Este cambio implica procesos de lectura mucho más cortos y una mayor capacidad de rendimiento. Este modelo se utiliza para minería y análisis de datos.

❖ Conclusión

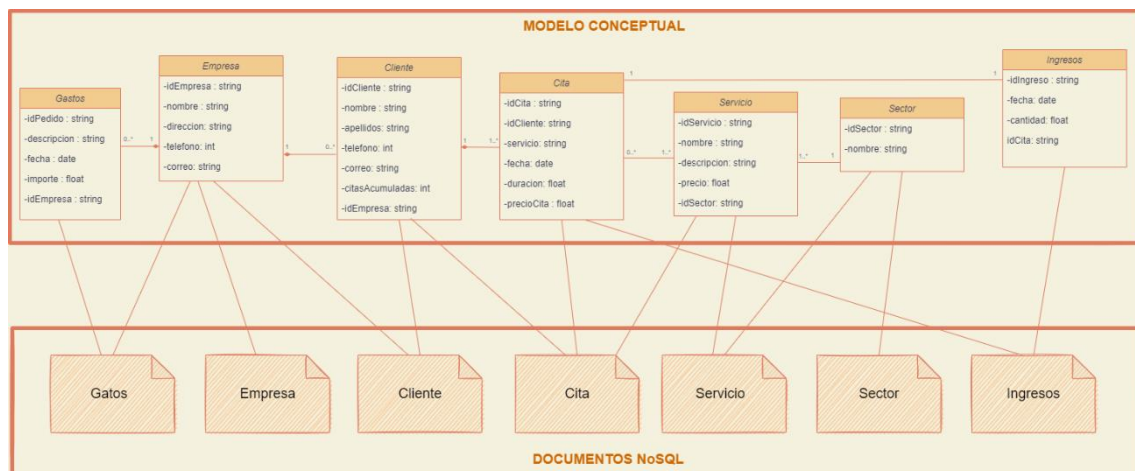
Una vez estudiadas las ventajas que trae consigo las bases de datos NoSQL y teniendo en cuenta los requisitos del propio proyecto que tenemos que llevar a cabo. Dado que puede estar sujeto tanto a cambios, como a un gran crecimiento si obtenemos una gran demanda, no queremos dejar de ofrecer calidad, en el sentido de no tener que parar la base de datos para ningún cambio y que la escritura y lectura sea lo más rápida y optima posible. Es por todo esto que nos hemos decantado por una base de datos NoSQL. En nuestro caso, como ya hemos referenciado anteriormente varias veces, haremos uso de Firebase.

4.5. Diagrama de la persistencia de la información

A continuación, mostramos el diagrama de la base de datos NoSQL que utilizaremos en el proyecto SheiStyle.

En la parte superior mostramos las clases, y sus relaciones, a la que los documentos de la base de datos hacen referencia.

Y en la parte inferior mostramos los documentos que contendrá la base de datos. Las clases con las que están relacionadas serán las clases de las que extraigan datos para su composición.



Una vez visto de manera visual, procederemos a explicarlo un poco de izquierda a derecha.

- Gastos:** Contendrá los elementos de la clase gastos y empresa. Debido a que un gasto es generado por la empresa.
- Empresa:** Contendrá los datos de la clase empresa únicamente.
- Cliente:** Contendrá los datos de las clases empresa y cliente. Debido a que el cliente está registrado en una empresa.
- Cita:** Contendrá los datos de cita, de cliente y de servicio. Debido a que una cita es asociada a un cliente con los servicios que desee.
- Servicio:** Contendrá los elementos de la clase servicio y sector. Debido a que un servicio pertenece a un sector en concreto.
- Sector:** Contendrá los elementos de su clase sector.
- Ingreso:** Contendrá los elementos de la clase ingreso y cita. Debido a que un ingreso viene dado por una cita.

4.6. Arquitectura del sistema

En los inicios de la informática, la programación era bastante difícil para la mayoría de las personas, pero con el tiempo se han ido descubriendo y desarrollando formas y guías generales para resolver problemas. A las mismas se les ha denominado arquitectura de software, por la semejanza con los planos de un edificio, debido a que la arquitectura indica la estructura, funcionamiento e interacción entre las distintas partes del software.

El diseño de una arquitectura de software utiliza los conocimientos de programación para planear el diseño general del software de modo que puedan agregarse detalles más adelante, lo cual permite a los equipos de software delimitar el panorama general y comenzar a elaborar un prototipo.

Algunos puntos clave:

- La arquitectura ayuda a definir una solución para cumplir con todos los requisitos técnicos y operativos, con el objetivo común de optimizar el rendimiento y la seguridad.
- El diseño de la arquitectura implica la intersección de las necesidades de la organización y las necesidades del equipo de desarrollo. Cada decisión puede tener un impacto considerable en la calidad, el mantenimiento, el rendimiento, etc.

Existen varios tipos de patrones de arquitectura del software, como son:

- Cliente-Servidor (Client-Server)
- Red entre iguales (Peer-to-Peer)
- Modelo-Vista-Controlador (MVC)
- Microservicios
- Event-driven
- En capas
- Hexagonal

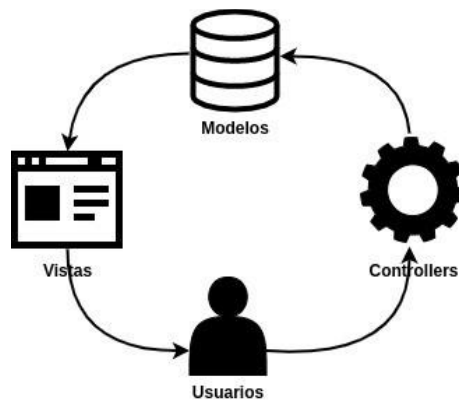
A continuación, vamos a explicar un poco el **patrón MVC** que es el que vamos a utilizar en nuestra aplicación SheiStyle.

La **arquitectura MVC** es un patrón arquitectónico de software en el que la lógica de la aplicación se divide en tres componentes sobre la base de la funcionalidad. Estos componentes se llaman:

-**Models o Modelos:** representan cómo se almacenan los datos en la base de datos.

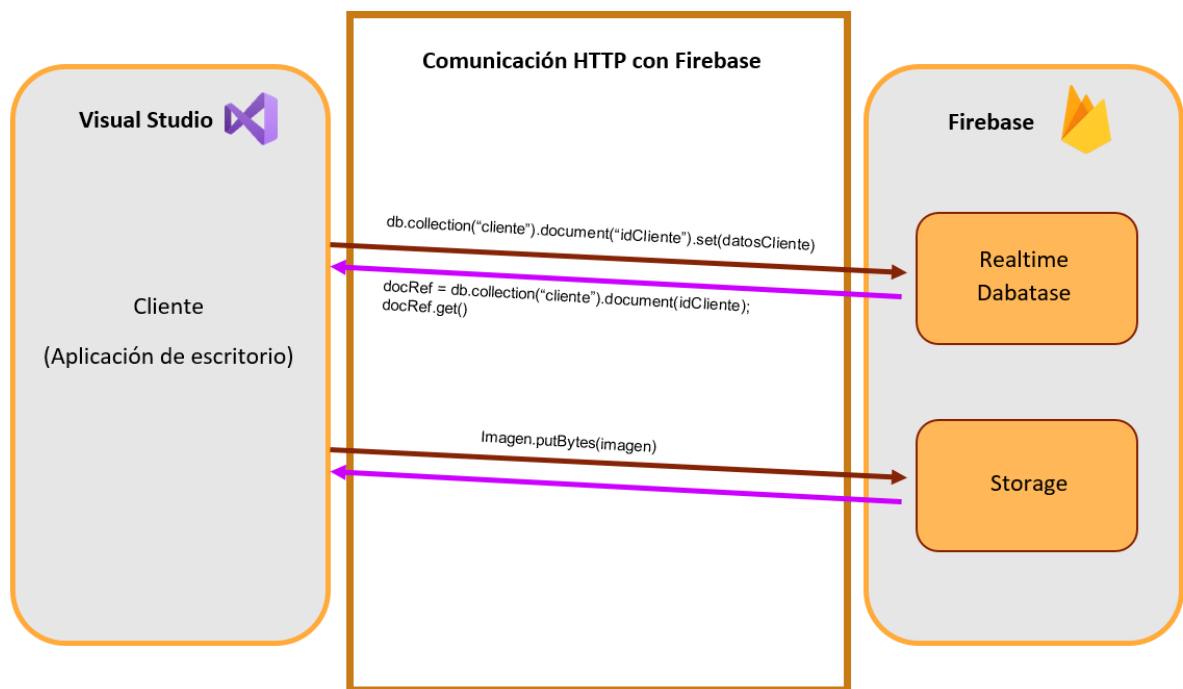
-**Views o Vistas:** los componentes que son visibles para el usuario, como una interfaz de usuario.

-Controllers o Controladores: los componentes que actúan como una interfaz entre los modelos y las vistas.



La arquitectura MVC se utiliza no solo para aplicaciones de escritorio, sino también para aplicaciones móviles y web. Aunque en nuestro caso, SheiStyle será una aplicación de escritorio donde podrá utilizarla el propio personal del local.

A continuación, mostramos el diagrama donde se podrán distinguir de manera clara el cliente (situado a la izquierda), el servidor (situado a la derecha) y las comunicaciones que se realizan entre ambos.



Nuestro servidor FIREBASE cuenta con servicios específicos dentro de él, de los cuales haremos uso.

- **RealTime Database:** este servicio nos permitirá el almacenamiento y manejo de datos desde nuestra aplicación, llevando consigo la mayor cantidad de datos interesantes de nuestra aplicación. En el ejemplo tratamos de obtener todos los animales que se encuentran dentro del documento animal de nuestra base de datos.
- **Storage:** este servicio nos permitirá almacenar imágenes en Firebase, subirlas y gestionarlas desde dentro. En el ejemplo tratamos de poner dentro de la carpeta de imágenes una imagen dada.

5.Implementación de la solución

En este apartado nos vamos a dedicar a analizar las tecnologías que podríamos usar, compararlas y definitivamente elegir una explicando el porqué de esta elección.

5.1. Análisis tecnológico

❖ Tecnología del cliente

- **Java**

Java es un lenguaje de programación y una plataforma informática nacida en 1995 por Sun Microsystems. Es una tecnología utilizada para el desarrollo de aplicaciones y uno de los lenguajes más populares y utilizados a nivel mundial, lo que crea un gran soporte contra posibles errores.

Algunas de sus características son:

- Simplicidad
- Lenguaje orientado a objetos
- Distribuido e independiente de la plataforma
- Seguro y multihilo



- **Kotlin**

Es un lenguaje de programación de tipado estático que corre sobre la máquina virtual de Java y que también puede ser compilado a código fuente de JavaScript.



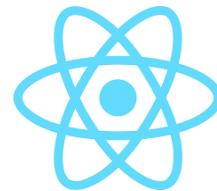
Kotlin destaca por varias características que no solo simplifican la lectura del código sino el propio desarrollo de este. Es por ello que las ventajas de Kotlin respecto a Java a la hora de desarrollar aplicaciones móviles ha hecho que este lenguaje se haga cada vez más popular. La simpleza de la sintaxis permite una curva de aprendizaje muy sencilla, ideal para aprender tu primer lenguaje de programación.

- **React Native**

React Native es un framework de programación de aplicaciones nativas multiplataforma que está basado en JavaScript y ReactJS y diseñado por Facebook. Por lo tanto, podremos crear aplicaciones nativas tanto para Android como para iOS.

Algunas características a destacar de React Native:

- Compatibilidad Cross-Platform
- Funcionalidad nativa
- Actualizaciones instantáneas
- Sencilla curva de aprendizaje
- Experiencia positiva para el desarrollador



❖ Tecnología del servidor

En el lado del servidor vamos a explicar las siguientes opciones:

○ **Spring Boot**

Spring Boot es una herramienta que nace con la finalidad de simplificar aún más el desarrollo de aplicaciones basadas en el ya popular framework Spring Core. Spring Boot busca que el desarrollador solo se centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.



El éxito de Spring Boot se centra en unas características que lo hacen extremadamente fácil de usar:

- Configuración: cuenta con un complejo módulo que autoconfigura todos los aspectos de nuestra aplicación para poder simplemente ejecutar la aplicación sin tener que definir absolutamente nada.
- Resolución de dependencias: solo hay que determinar qué tipo de proyecto estaremos utilizando y él se encarga de resolver todas las librerías para que la aplicación funcione.
- Despliegue: es posible desplegar las aplicaciones mediante un servidor web integrado, como es el caso de Tomcat, Jetty o Undertow.
- Métricas: Spring Boot cuenta con servicios que permite consultar el estado de salud de la aplicación, permitiendo saber si está prendida o apagada, memoria utilizada y disponible, ...
- Extensible: permite la creación de complementos, los cuales ayudan que la comunidad de Software Libre cree nuevos módulos que faciliten aún más el desarrollo.

Spring Boot permite compilar aplicaciones web como un archivo .jar que podemos ejecutar como una aplicación Java normal. Esto lo consigue integrando el servidor de aplicaciones en el propio .jar y levantándolo cuando arrancamos la aplicación. De esta forma podemos distribuir nuestras aplicaciones de una forma mucho más sencilla, al poder configurar el servidor junto con la aplicación. Esto también es muy útil en arquitectura de microservicios, puesto que permite distribuir nuestras aplicaciones como imágenes Docker y podemos escalar horizontalmente.

○ **Node.js**

Node.js es un entorno en tiempo de ejecución multiplataforma para la capa del servidor basado en JavaScript.

Es un entorno controlador por eventos diseñados para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo.



Algunas características principales de Node.js son:

- Velocidad: su biblioteca es muy rápida en la ejecución de código.
- Sin búfer: nunca almacenan los datos en búfer.
- Asíncrono y controlado por eventos: el servidor pasa a la siguiente API después de llamarla, y un mecanismo de notificación de eventos ayuda al servidor a obtener una respuesta de llamada anterior de la API.
- Un subproceso escalable: utiliza un modelo de un solo subproceso con bucle de eventos.

Node.js puede utilizarse para diferentes tipos de aplicaciones como, por ejemplo: aplicaciones streaming, intensivas de datos en tiempo real, vinculadas a E/S, de página única...

○ **Firebase**

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles lanzada en 2011.

Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.



Nosotros, como desarrolladores debemos tener en cuenta algunas de las siguientes ventajas:

- Sincronizar fácilmente los datos de sus proyectos sin tener que administrar conexiones o escribir lógica de sincronización compleja.
- Usa un conjunto de herramientas multiplataforma: se integra fácilmente para plataformas web como en aplicaciones móviles. Es compatible con grandes plataformas, como iOS, Android, aplicaciones web, Unity y C++.
- Usa la infraestructura de Google y escala automáticamente para cualquier tipo de aplicación, desde las más pequeñas hasta las más potentes.
- Crea proyectos sin necesidad de un servidor: Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.

Algunos de los servicios que nos ofrece Firebase son también:

- Realtime Database / Cloud Firestore

Son las bases de datos en tiempo real, se alojan en la nube y son NoSQL. Cloud Firestore es la nueva opción. Firebase envía automáticamente eventos a las aplicaciones cuando los datos cambian.

- Autenticación de usuario

Firebase ofrece un sistema de autenticación que permite el registro, como el acceso utilizando otras plataformas externas (como Google o Twitter)

- Almacenamiento en la nube

Firebase cuenta con un sistema de almacenamiento, donde los desarrolladores pueden guardar ficheros y sincronizarlos. Es personalizable mediante reglas. Es muy útil para guardar fotografías, por ejemplo.

- Cloud Messaging

Servicio de envío de notificaciones y mensajes a diversos usuarios en tiempo real y a través de varias plataformas.

Conclusión tecnológica para mi proyecto

Teniendo en cuenta el análisis anterior realizado para la parte del cliente y del servidor, SheiStyle va a desarrollarse en C# y contando con el servidor de Firebase.

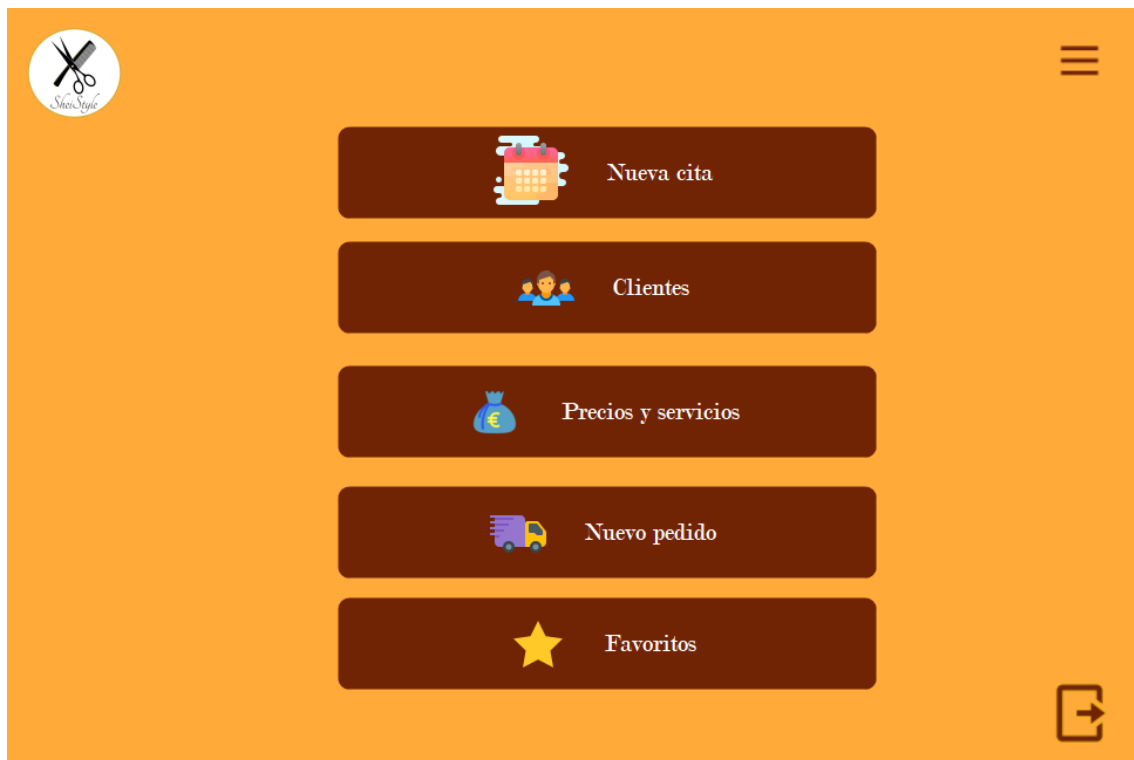
Utilizaremos Firebase como servidor, ya que proporciona servicios muy interesantes y cumple con todos los requisitos establecidos en mi aplicación. Para la subida de datos, podremos usar RealTime o Cloud Firestore. En nuestro

caso, serían válidas las dos; pero nos decantamos por usar RealTime por su eficacia al trabajar con datos en tiempo real. Además, en un futuro, con idea de evolucionar el proyecto, podríamos añadir el servicio de autenticación; ya que Firebase nos lo pone muy fácil para poder acceder mediante Google o Twitter.

Si hablamos de la parte del cliente, hemos decidido utilizar C# como lenguaje. Esto se debe a que sus características se ajustan mayoritariamente a la idea principal del proyecto. A esta conclusión, debemos sumarle mis conocimientos en este lenguaje. Debido a esto, creo que el desarrollo de la aplicación será más óptimo.

5.2 Elementos a implementar

Mi aplicación se iniciará con unas acciones básicas que se indicarán en la ventana principal, tales como:



En este menú podremos ver las funcionalidades por implementar, serán:

- Añadir una cita
- Gestionar los clientes
- Gestionar los precios y servicios
- Realizar pedidos de productos
- Visualizar gráfico según la demanda de los servicios

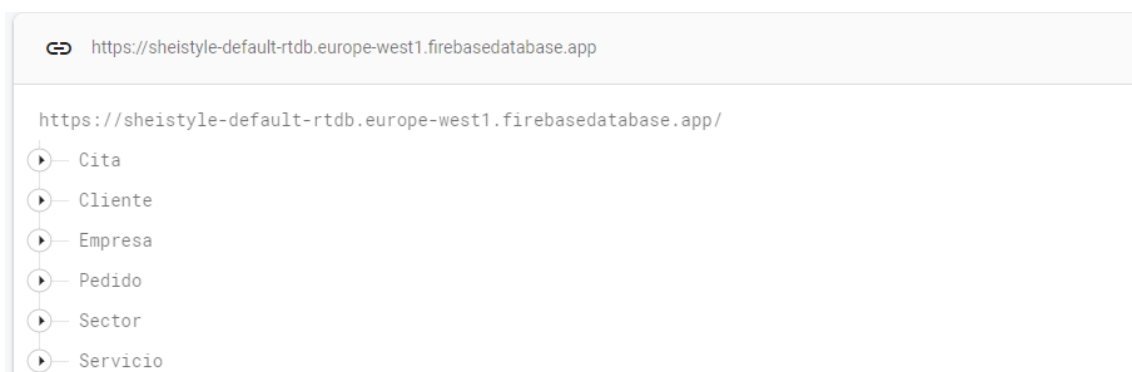
Antes de nada, hacemos la conexión a la base de datos:

```
IFirebaseConfig ifc = new FirebaseConfig()
{
    AuthSecret = "6vyUUSqV0nzcwGXgtzkyPvZDBe8ykvBXH6UV53I",
    BasePath = "https://sheistyle-default-rtdb.europe-west1.firebaseio.com/"
};

IFirebaseClient clien;

1 referencia
private void cargar()
{
    try
    {
        clien = new FireSharp.FirebaseClient(ifc);
    }
    catch (Exception)
    {
        throw; //no internet connection
    }
}
```

Esto es fundamental para poder tener acceso a nuestra base de datos de Firebase. Lo usaremos cada vez que necesitemos consultar algo de la base de datos.



En esta imagen, se muestra en link de mi base de datos y las tablas que lo contienen.

Base de datos	Secreto
sheistyle-default-rtdb	6vyUU5qV0nzcwGXgtzkyPvZDBe8ykvBXH6UV53I

Y este es el secreto de mi base de datos.

En la primera opción, podremos registrar una **nueva cita** en nuestra base de datos.

La siguiente pantalla nos mostrará un calendario, donde podremos seleccionar un día deseado.



En esta parte se ha controlado que no se pueda elegir una fecha anterior a la actual, y que por defecto cojamos el día de 'hoy' (día actual).

```

1 referencia
private void monthCalendar1_DateSelected(object sender, DateRangeEventArgs e)
{
    fechaSeleccionada = monthCalendar1.SelectionRange.Start;
}

1 referencia
private void FormCalendario_Load(object sender, EventArgs e)
{
    monthCalendar1.MinDate = fechaHoyDate;
}

1 referencia
private void btnAceptar_Click(object sender, EventArgs e)
{
    this.Hide();

    FormAgenda frmAgenda = new FormAgenda(fechaSeleccionada);
    frmAgenda.ShowDialog();
}

```


Al pulsar el botón aceptar, nos llevará al apartado de agenda donde le pasamos la fecha seleccionada para poder continuar con la cita.

En la siguiente pantalla se nos mostrará un listado con las citas reservadas del día que nosotros hayamos seleccionado. Ahí podemos ver a grandes rasgos, si la hora que la clienta nos pide para su cita está libre o no. Si queremos seguir con la reserva de nuestra cita, nos llevará a la ventana de servicios.

```
1 referencia
private void cargarTabla()
{
    String nombreC;
    FirebaseResponse res = clien.Get(@"Cita");
    Dictionary<string, Cita> data = JsonConvert.DeserializeObject<Dictionary<string, Cita>>(res.Body.ToString());

    FirebaseResponse res1 = clien.Get(@"Cliente");
    Dictionary<string, Cliente> data1 = JsonConvert.DeserializeObject<Dictionary<string, Cliente>>(res1.Body.ToString());

    //RELLENAR TABLAA
    dgvAgenda.Rows.Clear();
    dgvAgenda.Columns.Clear();
    dgvAgenda.Columns.Add("Fecha", "Fecha");
    dgvAgenda.Columns.Add("Hora", "Hora");
    dgvAgenda.Columns.Add("Cliente", "Cliente");

    foreach (var item in data)
    {
        if(item.Value.fecha.ToShortDateString().Equals(fecha.ToShortDateString()))
        {
            nombreC = nombrecliente(data1, item.Value.idCliente);
            dgvAgenda.Rows.Add(item.Value.fecha, "", nombreC);
        }
    }
}
```

Esta pantalla nos mostrará todos los servicios que ofrecemos en nuestro local, para poder seleccionar los deseados. También debemos seleccionar el nombre del cliente que solicita la cita (anteriormente debe estar dado de alta en nuestra base de datos), y por último la hora (que aparecerán solo las que estén libres).

Servicios

Peluquería

- ☐ Lavado
- ☐ Decoloracion
- ☐ Corte
- ☐ Alisado
- ☐ Tinte
- ☐ Permanente
- ☐ Mechas
- ☐ Recogido

Uñas

- ☐ Permanentes
- ☐ Pedicura
- ☐ Acrílicas
- ☐ De gel

Depilación con cera caliente

- ☐ Bigote
- ☐ Cejas

Láser

- ☐ Piernas
- ☐ Axilas
- ☐ Brazos
- ☐ Zona intima

Baños

- ☐ Recorte

Fecha: 08/06/2022 **Añadir**

Time selection dropdown: 8:00, 8:30, 9:00, 9:30, 10:00, 10:30, 11:00, 11:30, 12:00, 12:30, 13:00, 13:30, 14:00

Client selection dropdown: e d

Para crear la cita, debemos controlar que mínimo haya algún servicio seleccionado, que hayamos elegido una hora, y por último que seleccionemos un cliente.

Creamos un objeto cita de la siguiente manera:

```
1 referencia
private void btnAnadir_Click(object sender, EventArgs e)
{
    String servicios = serviciosSeleccionados();

    if(cbHoras.SelectedIndex == -1)
    {
        MessageBox.Show("Debes seleccionar una hora para reservar tu cita");
    }else if (string.IsNullOrEmpty(servicios))
    {
        MessageBox.Show("Debes seleccionar algún servicio para reservar tu cita");
    }
    else
    {
        FirebaseResponse res = clien.Get(@"Cliente");
        Dictionary<string, Cliente> data = JsonConvert.DeserializeObject<Dictionary<string, Cliente>>(res.Body.ToString());

        // Construimos la cita que vamos a registrar en base de datos
        Cita cita = new Cita();
        Guid UUID = Guid.NewGuid();
        cita.idCita = UUID.ToString();
        cita.idCliente = listaIdCliente[cbCliente.SelectedIndex].ToString();
        cita.servicio = servicios;
        cita.fecha = calcularHorasMinutos(cbHoras.SelectedItem.ToString());
        cita.duracion = dur;
        cita.precioCita = prec;

        MessageBox.Show("Servicios seleccionados : " + cita.servicio + " precio estimado: " + cita.precioCita + " duracion: " + cita.duracion);

        FormPresupuesto frmPresupuesto = new FormPresupuesto(cita);
        frmPresupuesto.ShowDialog();
    }
}
```

Este objeto cita se lo pasamos a la siguiente pantalla, que será la última para reservar la cita. En ella se nos mostrará el importe de ésta, y su descuento en caso de tenerlo.

Presupuesto:	20	€
Descuento:	-	%
Total:	20	€

Reservar cita

El descuento lo comprobamos de la siguiente manera:

```
1 referencia
private void comprobarDescuento()
{
    String idCliente = cita.idCliente;
    // Cliente cliente = new Cliente(idCliente, nombre, apellidos, telefono, correo);
    var res = clien.Get("Cliente/" + idCliente);
    resCliente = res.ResultAs<Cliente>();

    int citas = resCliente.citasAcumuladas;
    MessageBox.Show(citas.ToString());
    if(citas % 3 == 0)
    {
        lblDescuento.Text = "30";
        float total = cita.precioCita - (cita.precioCita * 0.3f);
        lblTotal.Text = total.ToString();
    }
    else
    {
        lblDescuento.Text = "-";
        lblTotal.Text = cita.precioCita.ToString(); ;
    }
}
```

Se le generará el descuento del 30% al usuario en caso de que sus citas acumuladas sean múltiplo de 3.

El último paso para guardar la cita será pulsar el botón 'reservar cita'.

```
1 referencia
private void btnReservarCita_Click(object sender, EventArgs e)
{
    SetResponse resCita = clien.Set(@"Cita/" + cita.idCita, cita);
    MessageBox.Show("Cita añadida con éxito");

    int citasAcumuladas = resCliente.citasAcumuladas + 1;
    Cliente clienteModificado = new Cliente(resCliente.idCliente, resCliente.nombre, resCliente.apellidos, resCliente.telefono,
        resCliente.correo, citasAcumuladas);
    var res = clien.Update("Cliente/" + resCliente.idCliente, clienteModificado);
    this.Close();
}
```

En este botón, añadimos una nueva cita a la tabla 'Cita' de la base de datos y actualizamos las citas acumuladas del cliente, para así poder realizarle el descuento cuando corresponda.

La segunda opción del menú principal es la gestión de **clientes**.



Nos mostrará un listado de todos los clientes registrados en la base de datos, que los cargaremos de la siguiente manera:

```
3 referencias
private void dibujarColumnasLista()
{
    listVClientes.Clear();
    listVClientes.Columns.Clear();

    listVClientes.Columns.Add("Nombre", 150);
    listVClientes.Columns.Add("Apellidos", 150);
    listVClientes.Columns.Add("Teléfono", 150);
    listVClientes.Columns.Add("Correo", 150);
    listVClientes.Columns.Add("IDCliente", 0);
}
```

Con esto dibujamos las columnas de la lista.

```
2 referencias
private void cargarListado()
{
    FirebaseResponse res = clien.Get(@"Cliente");
    Dictionary<string, Cliente> data = JsonConvert.DeserializeObject<Dictionary<string, Cliente>>(res.Body.ToString());
    rellenarListado(data);
}

1 referencia
private void rellenarListado(Dictionary<string, Cliente> data)
{
    dibujarColumnasLista();

    foreach (var item in data)
    {
        //En un array de String introducimos los datos de cada cliente
        String[] row = {item.Value.nombre, item.Value.apellidos, item.Value.telefono, item.Value.correo, item.Value.idCliente};
        //Introducimos los items (clientes) en el listado
        var itemListView = new ListViewItem(row);
        listVClientes.Items.Add(itemListView);
    }
}
```

Y con esto, recogemos los datos de la tabla clientes de la bbdd, los recorremos uno a uno y lo introducimos en las filas de la lista para mostrarlo posteriormente.

En la parte de arriba de esta pantalla, tenemos un filtro. En esta parte podemos buscar por nombre de los clientes. Lo hacemos de la siguiente manera:

```
1 referencia
private void tbFiltroNombre_TextChanged(object sender, EventArgs e)
{
    String nombreBuscado = tbFiltroNombre.Text;

    dibujarColumnasLista();

    FirebaseResponse res = clien.Get(@"Cliente");
    Dictionary<string, Cliente> data = JsonConvert.DeserializeObject<Dictionary<string, Cliente>>(res.Body.ToString());
    foreach (var item in data)
    {
        if (item.Value.nombre.Contains(nombreBuscado))
        {
            //En un array de String introducimos los datos de cada cliente
            String[] row = { item.Value.nombre, item.Value.apellidos, item.Value.telefono, item.Value.correo, item.Value.idCliente };
            //Introducimos los items (clientes) en el listado
            var itemListView = new ListViewItem(row);
            listvClientes.Items.Add(itemListView);
        }
    }
}
```

Según vayamos cambiando el campo del filtro, irá buscando clientes que contengan ese texto que hayamos introducido como nombre.

También contamos con un botón 'buscar', que tendrá la misma función.

```
1 referencia
private void btnBuscar_Click(object sender, EventArgs e)
{
    String nombreBuscado = tbFiltroNombre.Text;
    bool encontrado = false;

    dibujarColumnasLista();

    FirebaseResponse res = clien.Get(@"Cliente");
    Dictionary<string, Cliente> data = JsonConvert.DeserializeObject<Dictionary<string, Cliente>>(res.Body.ToString());
    foreach (var item in data)
    {
        if (item.Value.nombre.Contains(nombreBuscado))
        {
            //En un array de String introducimos los datos de cada cliente
            String[] row = { item.Value.nombre, item.Value.apellidos, item.Value.telefono, item.Value.correo, item.Value.idCliente };
            //Introducimos los items (clientes) en el listado
            var itemListView = new ListViewItem(row);
            listvClientes.Items.Add(itemListView);
            encontrado = true;
        }
    }
    if (!encontrado)
    {
        MessageBox.Show("No se han encontrado clientes con ese nombre");
    }
}
```

Los siguientes botones llevan a cabo la gestión de los clientes, y para ello es necesario que haya alguno seleccionado. Lo controlamos de esta manera:

```
private bool comprobarClienteSeleccionado()
{
    if (listVClientes.SelectedItems.Count == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Si queremos eliminar, una vez hayamos seleccionado un cliente, hacemos clic en el botón correspondiente:

```
//comprueba que hay un cliente seleccionado y lo elimina
1 referencia
private void btnEliminarCliente_Click(object sender, EventArgs e)
{
    if (comprobarClienteSeleccionado())
    {
        Cliente resCliente = recuperarDatosClienteSeleccionado();
        var eliminar = clien.Delete("Cliente/" + resCliente.idCliente);
        MessageBox.Show("Cliente " + resCliente.nombre + " se ha eliminado con éxito");
        cargarListado();
    }
    else
    {
        MessageBox.Show("Debes seleccionar un cliente");
    }
}
```

Para ello debemos recuperar los datos del cliente seleccionado:

```
//Recuperamos los datos del cliente que hemos seleccionado
2 referencias
private Cliente recuperarDatosClienteSeleccionado()
{
    //Obtenemos la fila que pulsamos
    int fila = listVClientes.FocusedItem.Index;
    //Obtenemos el idCliente de la fila que hemos seleccionado
    String idCliente = listVClientes.Items[fila].SubItems[4].Text;

    //Obtenemos el cliente que tenfa ese id y lo devolvemos
    FirebaseResponse res = clien.Get(@"Cliente/" + idCliente);
    Cliente resCliente = res.ResultAs<Cliente>();
    return resCliente;
}
```

Para editar un cliente, abriremos una nueva venta donde podremos modificar algunos datos del cliente.

Cliente

Modifica los campos que desee:

Nombre: a

Apellidos: a

Teléfono: 455-555-555

Correo: a

Actualizar cliente

```
1 referencia
private void btnActualizarCliente_Click(object sender, EventArgs e)
{
    bool correcto;

    string idCliente = resCliente.idCliente;
    string nombre = tbNombre.Text;
    string apellidos = tbApellidos.Text;
    string telefono = tbTlfn.Text;
    string correo = tbCorreo.Text;

    correcto = comprobarDatosCliente(nombre, apellidos, telefono, correo);

    if (correcto)
    {
        Cliente clienteModificado = new Cliente(idCliente, nombre, apellidos, telefono, correo);
        var res = clien.Update("cliente/" + idCliente, clienteModificado);

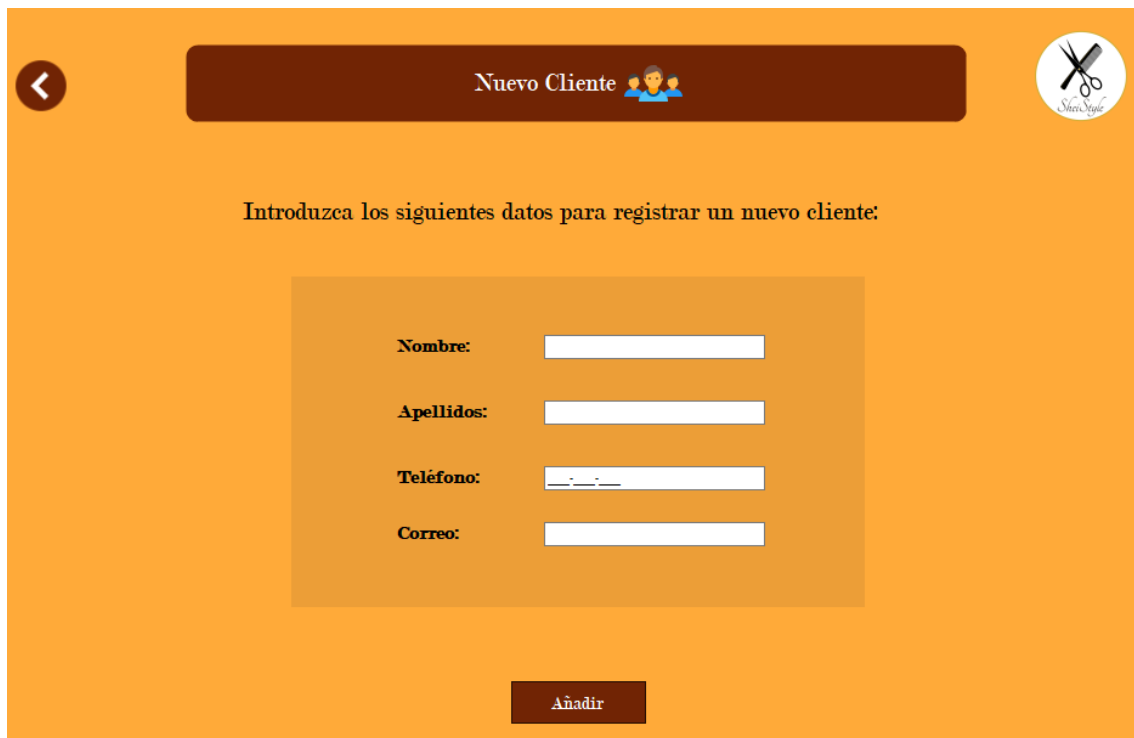
        MessageBox.Show("Cliente " + nombre + " se ha modificado con éxito");

        this.Close();
        FormListadoClientes frmListadoCliente = new FormListadoClientes();
        frmListadoCliente.Show();
    }
}
```

Para ello, es necesario comprobar que no hay ningún campo vacío y todos tienen su formato correspondiente, lo haremos desde el siguiente método:

```
private bool comprobarDatosCliente(string nombre, string apellidos, string telefono, string correo)
{
    if (string.IsNullOrEmpty(nombre) || string.IsNullOrEmpty(apellidos) || string.IsNullOrEmpty(telefono)
        || string.IsNullOrEmpty(correo))
    {
        MessageBox.Show("Debes rellenar todos los campos");
        return false;
    }
    else
    {
        if (!tbTlfn.MaskCompleted) //comprueba que el tlfn tenga 9 digitos
        {
            MessageBox.Show("El teléfono debe contener 9 dígitos");
            return false;
        }
        else if (!correoValido(correo))
        {
            MessageBox.Show("Formato del correo erróneo");
            return false;
        }
        else
        {
            return true;
        }
    }
}
```

Para registrar un nuevo cliente, también se nos mostrará una nueva ventana para poder rellenarla con sus datos.



The screenshot shows a mobile application interface with an orange background. At the top, there is a dark brown header bar containing a back arrow icon on the left, the text "Nuevo Cliente" with a family icon in the center, and a circular logo with scissors on the right. Below the header, the text "Introduzca los siguientes datos para registrar un nuevo cliente:" is displayed. In the center, there is a white rectangular box with rounded corners containing four input fields: "Nombre:", "Apellidos:", "Teléfono:" (with a mask "____-____-____"), and "Correo:". At the bottom center, there is a dark brown button with the text "Añadir".


```

1 referencia
private void btnAnadirCliente_Click(object sender, EventArgs e)
{
    bool correcto;
    Guid UUID = Guid.NewGuid();
    String idCliente = UUID.ToString();
    String nombre = tbNombre.Text;
    String apellidos = tbApellidos.Text;
    String telefono = tbTlfn.Text;
    String correo = tbCorreo.Text;

    correcto = comprobarDatosCliente(nombre, apellidos, telefono, correo);

    if (correcto)
    {
        Cliente cliente = new Cliente(idCliente, nombre, apellidos, telefono, correo);
        SetResponse res = clien.Set(@"Cliente/" + idCliente, cliente);

        MessageBox.Show("Cliente " + nombre + " añadido con éxito");

        tbNombre.Text = "";
        tbApellidos.Text = "";
        tbCorreo.Text = "";
        tbTlfn.Text = "";
    }
}

```

En este caso como en el anterior, también tenemos controlado que no haya ningún campo vacío y también que los campos 'teléfono' y 'correo' tengan sus formatos correspondientes.

Por último, el botón 'Mostrar citas', nos muestra una lista con todas las citas que ha reservado ese cliente en nuestro local.

La tercera opción del menú principal es la gestión de **servicios**. En ella mostraremos una tabla por cada sector que incluirá los servicios asociados.

Precios y servicios

Peluquería

Servicio	Precio
Permanente	35
Mechas	18
Lavado	5
Corte	15.2
Recogido	18
Decoloracion	30
Tinte	22
Alisado	55

Uñas

Servicio	Precio
Acrilicas	18
Permanentes	12
Pedicura	10.5
De gel	16

Depilación cera caliente

Servicio	Precio
Bigote	7
Cejas	5

Barbería

Servicio	Precio
Recorte	7
Afeitado	7

Depilación láser

Servicio	Precio
Piernas	20
Zona íntima	20
Axilas	12
Brazos	13

Modificar precio

Para cargar las tablas, consultamos por cada sector, para así poder recuperar los servicios que pertenezcan a ese sector:

```

1 referencia
private void FormPrecios_Load(object sender, EventArgs e)
{
    crearTablas();

    FirebaseResponse res = clien.Get(@"Sector");
    Dictionary<string, Sector> data = JsonConvert.DeserializeObject<Dictionary<string, Sector>>(res.Body.ToString());

    foreach (var item in data)
    {
        array.Add(item.Value.idSector);
    }

    FirebaseResponse res1 = clien.Get(@"Servicio");
    Dictionary<string, Servicio> data1 = JsonConvert.DeserializeObject<Dictionary<string, Servicio>>(res1.Body.ToString());

    foreach (var item in data1)
    {
        if (array[0].ToString() == item.Value.idSector) //sector CERA
        {
            dgvCera.Rows.Add(item.Value.nombre, item.Value.precio, item.Value.idServicio);
        }

        if (array[1].ToString() == item.Value.idSector) //sector UÑAS
        {
            dgvUnas.Rows.Add(item.Value.nombre, item.Value.precio, item.Value.idServicio);
        }

        if (array[2].ToString() == item.Value.idSector) //sector BARBERIA
    }
}

```

En esta ventana también tenemos la opción de modificar el precio de cada servicio en caso de que sea necesario.

```
1referencia
private void btnModificarPrecio_Click(object sender, EventArgs e)
{
    if(dgvPeluqueria.SelectedRows.Count>0)
    {
        seleccionItem(dgvPeluqueria);
    } else if (dgvCera.SelectedRows.Count > 0)
    {
        seleccionItem(dgvCera);
    }
    else if (dgvBarberia.SelectedRows.Count > 0)
    {
        seleccionItem(dgvBarberia);
    }
    else if (dgvLaser.SelectedRows.Count > 0)
    {
        seleccionItem(dgvLaser);
    }
    else if (dgvUnas.SelectedRows.Count > 0)
    {
        seleccionItem(dgvUnas);
    }
    else if (dgvCera.SelectedRows.Count > 0)
    {
        seleccionItem(dgvCera);
    }
}
```

Para ello es necesario comprobar que hay alguno seleccionado, y después abrimos una nueva ventana de edición, donde mostrará los datos de este, y solo nos dejará editar el campo del precio.



```

1 referencia
private void FormModificarPrecio_Load(object sender, EventArgs e)
{
    FirebaseResponse res = clien.Get(@"Servicio");
    Dictionary<string, Servicio> data = JsonConvert.DeserializeObject<Dictionary<string, Servicio>>(res.Body.ToString());

    foreach (var item in data)
    {
        if(item.Value.idServicio == idServicio)
        {
            tbServicio.Text = item.Value.nombre;
            tbSector.Text = obtenerNombreSector(item.Value.idSector);
            tbPrecio.Text = item.Value.precio.ToString();
        }
    }
}

```

Los datos de ese servicio los rescatamos de esta manera. Esto podemos hacerlo porque a esta pantalla le hemos pasado el id del servicio seleccionado, con lo cual podemos consultar en la base de datos por ese servicio.

A la hora de guardar el cambio del precio, debemos actualizar ese servicio de la siguiente manera:

```

1 referencia
private void btnActualizar_Click(object sender, EventArgs e)
{
    if (tbPrecio.Text!=null)
    {
        FirebaseResponse res = clien.Get(@"Servicio/" + idServicio);
        Servicio resServicio = res.ResultAs<Servicio>();

        Servicio servicioModificado = new Servicio(resServicio.idServicio, resServicio.nombre,
            resServicio.descripcion, float.Parse(tbPrecio.Text), resServicio.duracion, resServicio.idSector);
        var res2 = clien.Update("Servicio/" + idServicio, servicioModificado);

        MessageBox.Show("El precio de " + resServicio.nombre + " ha sido modificado con éxito");

        this.Close();
        FormPrecios formPrecios = new FormPrecios();
        formPrecios.Show();
    }
    else
    {
        MessageBox.Show("Completa el campo de precio");
    }
}

```

La cuarta opción del menú principal es **nuevo pedido**.



En esta pantalla podemos registrar un nuevo pedido, lo que hará que nuestra empresa tenga unos gastos.

Para esto tan solo es necesario, concretar en la descripción la información del pedido (productos pedidos) y el precio que nos costará.

```
private void btnActualizarCliente_Click(object sender, EventArgs e)
{
    String idEmpresa="";

    FirebaseResponse res1 = clien.Get(@"Empresa");
    Dictionary<string, Empresa> data1 = JsonConvert.DeserializeObject<Dictionary<string, Empresa>>(res1.Body.ToString());

    foreach (var item in data1)
    {
        idEmpresa = item.Value.idEmpresa;
    }

    Guid UUID = Guid.NewGuid();
    String idPedido = UUID.ToString();
    String descripcion = tbDescripcion.Text;
    float importe = float.Parse(tbImporte.Text);
    DateTime fecha = DateTime.Now;

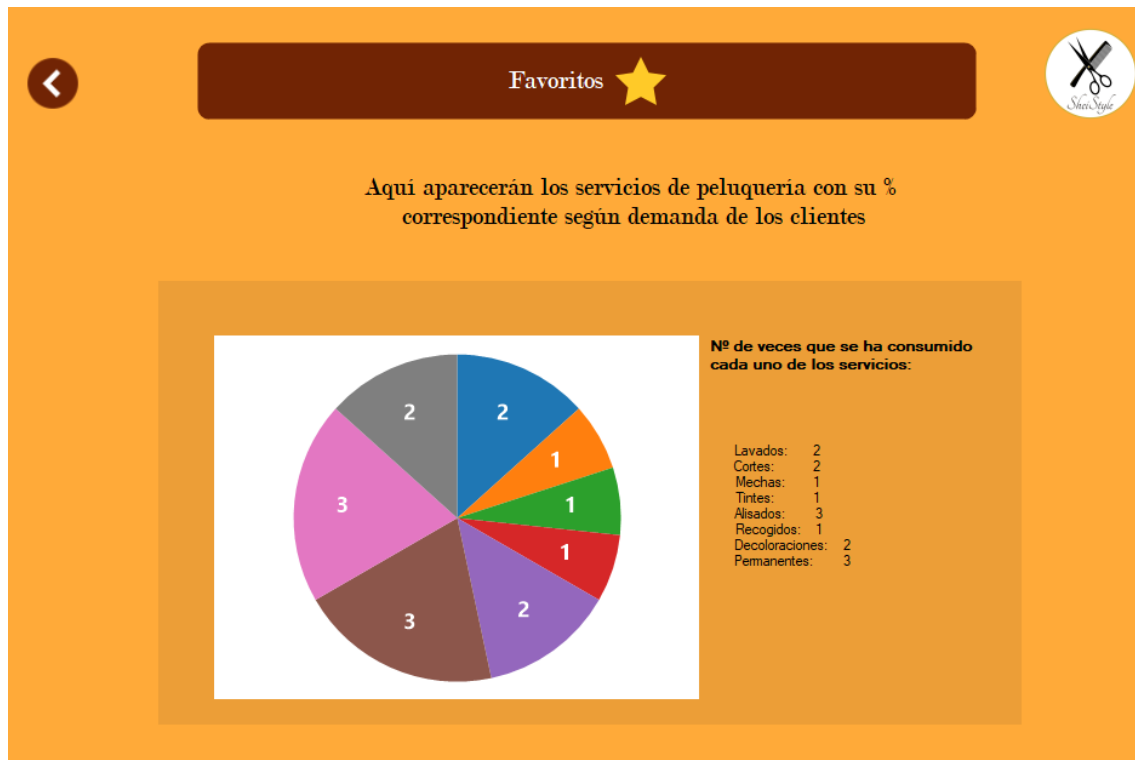
    if(string.IsNullOrEmpty(descripcion) || string.IsNullOrEmpty(importe.ToString())){
        MessageBox.Show("Debes rellenar todos los campos");
    }
    else
    {
        Pedido pedido = new Pedido(idPedido, descripcion, importe, fecha, idEmpresa);
        SetResponse res = clien.Set(@"Pedido/" + idPedido, pedido);

        MessageBox.Show("El pedido se ha generado con éxito");

        tbDescripcion.Text = "";
        tbImporte.Text = "";
    }
}
```

Recogemos los datos de la pantalla, controlamos que no estén vacíos y procedemos a hacer un Insert en la base de datos guardando así nuestro nuevo pedido (gasto).

La quinta opción de la pantalla principal, es la de **favoritos**. Donde vamos a analizar un poco nuestra situación de demanda relacionados con nuestros servicios de peluquería. Hemos usado un gráfico para representarlo de manera más visual:



Para ello, hemos necesitado consultar en la tabla Citas de la base de datos:

```
int permanente = 0;
FirebaseResponse res = clien.Get(@"Cita");
Dictionary<string, Cita> data = JsonConvert.DeserializeObject<Dictionary<string, Cita>>(res.Body.ToString());
foreach (var item in data)
```

Una vez hemos rescatado los datos que deseemos, los metemos en el gráfico circular de la siguiente manera:

```
double[] servGrafico = {lavado, recogido, mechas, tinte, corte, alisado, permanente, decoloracion };

var pie = grafServicios.Plot.AddPie(servGrafico);
grafServicios.Refresh();
pie.ShowValues = true;
```

A parte de los botones principales que muestra la ventana principal, también contamos con un menú que tendrá varias secciones. Una de ellas, es mostrar los **datos de la empresa** en una ventana.



Datos de la empresa

Ingresos

Gastos

Datos empresa

Nombre:	SheiStyle
Dirección:	Av. Andalucía, nº0603
Teléfono:	901-568-924
Correo:	sheistyle@atencioncliente.es

Para ello, hemos consultado la base de datos para rescatar los datos que corresponden a la empresa. Lo hemos conseguido desde la tabla 'empresa' de la siguiente manera:

```
1 referencia
private void FrmDatosEmpresa_Load(object sender, EventArgs e)
{
    FirebaseResponse res = clien.Get(@"Empresa");
    Dictionary<string, Empresa> data = JsonConvert.DeserializeObject<Dictionary<string, Empresa>>(res.Body.ToString());

    foreach (var item in data)
    {
        tbNombre.Text = item.Value.nombre;
        tbTlfn.Text = item.Value.telefono;
        tbDireccion.Text = item.Value.direccion;
        tbCorreo.Text = item.Value.correo;
    }
}
```

Como se ve, en el menú de la izquierda, hay más opciones.

Tenemos los apartados de ingresos y gastos, que llevarán la cuenta del movimiento de dinero que se produce en nuestro local. Esto lo hemos mostrado de manera visual en gráficos. Para ello hemos realizado consultas a sus respectivas tablas de la base de datos.

En la parte de ingresos:

1 referencia

```
private void consultaIngresos()
{
    ArrayList listaCantidadIngreso = new ArrayList();
    float cantIngreso = 0;
    FirebaseResponse res = clien.Get(@"Ingreso");
    Dictionary<string, Ingreso> data = JsonConvert.DeserializeObject<Dictionary<string, Ingreso>>(res.Body.ToString());
    foreach (var item in data)
    {
        cantIngreso = item.Value.cantidad + cantIngreso;
        listaCantidadIngreso.Add(cantIngreso);
    }
    double[] cantGrafico = new double[listaCantidadIngreso.Count];
    double[] tiempoGrafico = new double[listaCantidadIngreso.Count];

    for (int i = 0; i < listaCantidadIngreso.Count; i++)
    {
        cantGrafico[i] = double.Parse(listaCantidadIngreso[i].ToString());
        tiempoGrafico[i] = i;
    }

    lblIngresos.Text = cantIngreso.ToString();

    grafIngresos.Plot.AddScatter(tiempoGrafico, cantGrafico);

    grafIngresos.Refresh();
}
```

En la parte de gastos:

2. Testnet testnet

```
private void consultarGastos()
{
    ArrayList listaCantidadGasto = new ArrayList();
    float cantGasto = 0;
    FirebaseResponse res1 = clien.Get(@"Pedido");
    Dictionary<string, Pedido> data1 = JsonConvert.DeserializeObject<Dictionary<string, Pedido>>(res1.Body.ToString());
    foreach (var item in data1)
    {
        cantGasto = item.Value.importe + cantGasto;
        listaCantidadGasto.Add(cantGasto);
    }
    double[] cantGraficoGasto = new double[listaCantidadGasto.Count];
    double[] tiempoGrafico = new double[listaCantidadGasto.Count];

    for (int i = 0; i < listaCantidadGasto.Count; i++)
    {
        cantGraficoGasto[i] = double.Parse(listaCantidadGasto[i].ToString());
        tiempoGrafico[i] = i;
    }

    lblGastos.Text = cantGasto.ToString();

    grafGastos.Plot.AddScatter(tiempoGrafico, cantGraficoGasto);

    grafGastos.Refresh();
}
```


6. Testeo y pruebas de la solución

6.1. Plan de pruebas

Es el producto formal que define los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para elaborar una planificación paso a paso de las actividades de prueba. Se plantean los siguientes niveles de prueba:

- **Pruebas unitarias**

Es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, si el estado inicial es válido, entonces el estado final es válido también.

Los pasos necesarios para llevar a cabo pruebas unitarias son:

- Ejecutar los casos de uso, esperando que los casos de prueba contemplen todas las condiciones válidas y esperadas como las inválidas e inesperadas.
- Corregir los errores o defectos encontrados y repetir las pruebas que los detectaron.

La prueba unitaria está finalizada cuando se cumplan todas las verificaciones y no se encuentre ningún defecto.

- **Pruebas de integración**

se realizan una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo. Se centra principalmente en probar la comunicación entre los componentes y sus comunicaciones ya sea hardware o software.

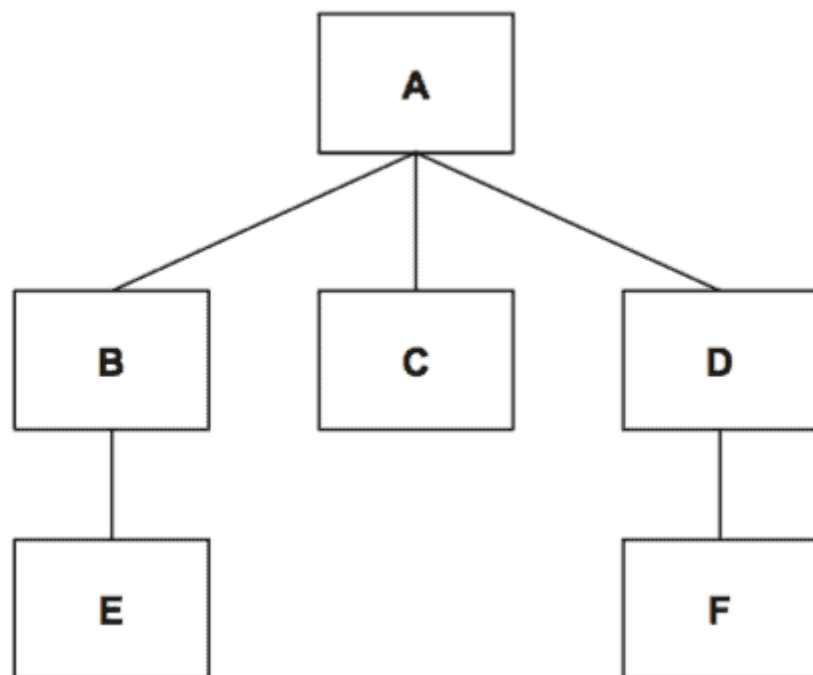
Se distinguen las siguientes estrategias de integración:

- **De arriba abajo (top-down).** El primer componente que se desarrolla y prueba es el primero de la jerarquía (A). Los componentes de nivel más bajo se sustituyen por componentes auxiliares para simular a los componentes invocados. En este caso no son necesarios componentes conductores. Una de las ventajas de aplicar esta estrategia es que las interfaces entre los distintos componentes se prueban en una fase temprana y con frecuencia.

- **De abajo arriba (*bottom-up*)**. En este caso se crean primero los componentes de más bajo nivel (E, F) y se crean componentes conductores para simular a los componentes que los llaman. A continuación, se desarrollan los componentes de más alto nivel (B, C, D) y se prueban. Por último, dichos componentes se combinan con el que los llama (A). Los componentes auxiliares son necesarios en raras ocasiones.

Este tipo de enfoque permite un desarrollo más en paralelo que el enfoque de arriba abajo, pero presenta mayores dificultades a la hora de planificar y de gestionar.

- **Estrategias combinadas**. A menudo es útil aplicar las estrategias anteriores conjuntamente. De este modo, se desarrollan partes del sistema con un enfoque «*top-down*», mientras que los componentes más críticos en el nivel más bajo se desarrollan siguiendo un enfoque «*bottom-up*». En este caso es necesaria una planificación cuidadosa y coordinada de modo que los componentes individuales se «encuentren» en el centro.



- **Pruebas del sistema**

Tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

Son pruebas de integración del sistema de información completo, y permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen. Dan una visión muy similar a su comportamiento en el entorno de producción.

En esta etapa pueden distinguirse los siguientes tipos de pruebas, cada uno con un objetivo claramente diferenciado:

- **Pruebas funcionales.** Dirigidas a asegurar que el sistema de información realiza correctamente todas las funciones que se han detallado en las especificaciones dadas por el usuario del sistema.
- **Pruebas de comunicaciones.** Determinan que las interfaces entre los componentes del sistema funcionan adecuadamente, tanto a través de dispositivos remotos, como locales. Asimismo, se han de probar las interfaces hombre/máquina.
- **Pruebas de rendimiento.** Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.
- **Pruebas de volumen.** Consisten en examinar el funcionamiento del sistema cuando está trabajando con grandes volúmenes de datos, simulando las cargas de trabajo esperadas.
- **Pruebas de sobrecarga.** Consisten en comprobar el funcionamiento del sistema en el umbral límite de los recursos, sometiénolo a cargas masivas. El objetivo es establecer los puntos extremos en los cuales el sistema empieza a operar por debajo de los requisitos establecidos.
- **Pruebas de disponibilidad de datos.** Consisten en demostrar que el sistema puede recuperarse ante fallos, tanto de equipo físico como lógico, sin comprometer la integridad de los datos.
- **Pruebas de facilidad de uso.** Consisten en comprobar la adaptabilidad del sistema a las necesidades de los usuarios, tanto para asegurar que se acomoda a su modo habitual de trabajo, como para determinar las facilidades que aporta al introducir datos en el sistema y obtener los resultados.
- **Pruebas de operación.** Consisten en comprobar la correcta implementación de los procedimientos de operación, incluyendo la planificación y control de trabajos, arranque y re arranque del sistema, etc.

- **Pruebas de entorno.** Consisten en verificar las interacciones del sistema con otros sistemas dentro del mismo entorno.
- **Pruebas de seguridad.** Consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.

- **Pruebas de implantación**

El objetivo de las pruebas de implantación es comprobar el funcionamiento correcto del sistema integrado de hardware y software en el entorno de operación, y permitir al usuario que, desde el punto de vista de operación, realice la aceptación del sistema una vez instalado en su entorno real y en base al cumplimiento de los requisitos no funcionales especificados.

- **Pruebas de aceptación**

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

Estas pruebas van dirigidas a comprobar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catálogo de requisitos y en los criterios de aceptación del sistema de información, y conseguir así la aceptación final del sistema por parte del usuario.

- **Pruebas de regresión**

El objetivo de las pruebas de regresión es eliminar el efecto onda, es decir, comprobar que los cambios sobre un componente de un sistema de información, no introducen un comportamiento no deseado o errores adicionales en otros componentes no modificados.

Las pruebas de regresión pueden incluir:

- La repetición de los casos de pruebas que se han realizado anteriormente y están directamente relacionados con la parte del sistema modificada.
- La revisión de los procedimientos manuales preparados antes del cambio, para asegurar que permanecen correctamente.
- La obtención impresa del diccionario de datos de forma que se compruebe que los elementos de datos que han sufrido algún cambio son correctos.

6.2. Realización de pruebas

❖ **Prueba 1:** Cogemos una fecha para una nueva cita

- Comprobamos que no se pueda coger una fecha anterior a la actual



Esto lo conseguimos de la siguiente manera:

```
1 referencia
private void monthCalendar1_DateSelected(object sender, DateRangeEventArgs e)
{
    fechaSeleccionada = monthCalendar1.SelectionRange.Start;
}

1 referencia
private void FormCalendario_Load(object sender, EventArgs e)
{
    monthCalendar1.MinDate = fechaHoyDate;
}
```

Al cargar el formulario, le decimos que la fecha mínima sea la actual (que ya la hemos sacado anteriormente). Y por defecto, decimos que se seleccione el día actual para así asegurarnos de haber escogido un día y que sea válido.

Con esta comprobación no damos lugar a coger una cita en el pasado.

- ❖ **Prueba 2:** al hacer doble clic sobre una cita en la agenda de un día en concreto, podemos ver la duración y el teléfono de contacto del cliente, se comprueba:

- Solo ocurra algo cuando haya una fila de la tabla seleccionada.

```
/*
 * Metodo que se produce cuando hacemos doble click sobre un registro de la tabla
 * */
1 referencia
private void dgvAgenda_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    String mensaje = "";
    //Si alguna fila de la tabla está seleccionada
    if (dgvAgenda.SelectedCells.Count > 0)
    {
```

Esto lo controlamos metiendo el código que queramos ejecutar dentro de este bloque de código.

En la siguiente pantalla podemos ver varias pruebas.

The screenshot shows a mobile application interface for a hair salon. At the top, there is a navigation bar with a back arrow, the title "Servicios", a calendar icon, and a logo. Below the navigation bar, the services are organized into five categories, each with a list of options and checkboxes:

- Peluquería:**
 - ☐ Lavado
 - ☐ Decoloracion
 - ☐ Corte
 - ☐ Alisado
 - ☐ Tinte
 - ☐ Permanente
 - ☐ Mechas
 - ☐ Recogido
- Uñas:**
 - ☐ Permanentes
 - ☐ Pedicura
 - ☐ Acrílicas
 - ☐ De gel
- Depilación con cera caliente:**
 - ☐ Bigote
 - ☐ Cejas
- Láser:**
 - ☐ Piernas
 - ☐ Axilas
 - ☐ Brazos
 - ☐ Zona íntima
- Barbería:**
 - ☐ Afeitado
 - ☐ Recorte

At the bottom of the screen, there is a form to add a new appointment. It includes a date field labeled "Fecha:" with the value "14/06/2022", a button labeled "Añadir", a time selection dropdown labeled "Seleccione la hora deseada:" with the value "12:30", and a client selection dropdown labeled "Elija un cliente:" with the value "david clemente".

❖ **Prueba 3:** elegir un cliente para la cita.

- Por defecto seleccionamos el primer cliente de la lista, para no permitir que se guarde una cita sin cliente.

Lo conseguimos de la siguiente manera:

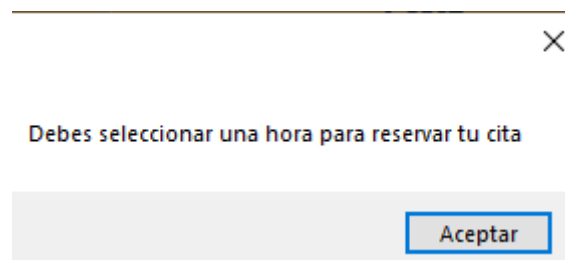
```
//Por defecto, seleccionamos el primer cliente del combobox  
cbCliente.SelectedIndex = 0;
```

❖ **Prueba 4:** elegir hora para la cita.

- En el combobox solo mostramos las horas disponibles que nos quedan para ese día. Borrando de esta manera las que estén ocupadas por otras citas
- Comprobamos que se seleccione una hora para reservar la cita

```
if(cbHoras.SelectedIndex == -1)  
{  
    MessageBox.Show("Debes seleccionar una hora para reservar tu cita");  
}
```

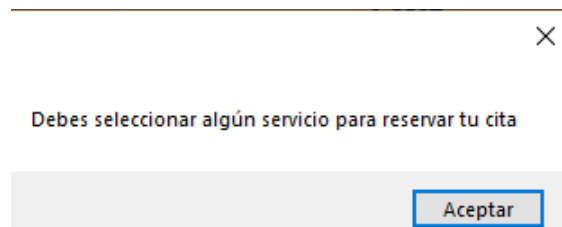
En caso de no ser así, mostramos un mensaje de aviso



❖ **Prueba 5:** elegir los servicios de una cita

- Comprobamos que se haya elegido al menos 1 servicio para poder registrar una cita.

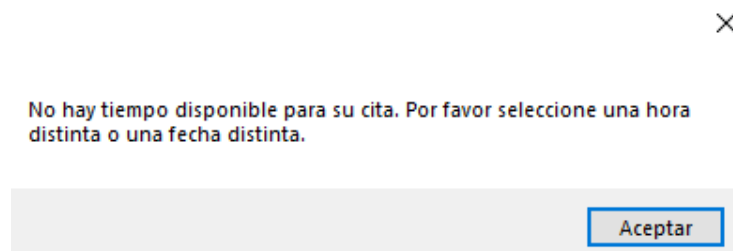
En caso de que esto no se cumpla, se muestra un mensaje de aviso



❖ **Prueba 6:** elegimos unos servicios y una hora concreta

- Comprobamos que nuestra cita tenga el tiempo suficiente para realizarse y no se solape con la siguiente

En caso de que no dé tiempo a realizar esos servicios en esa hora, mostramos un mensaje por pantalla para que se busque una hora acuerdo con esos servicios.



En este formulario también contamos con varias pruebas



The screenshot shows a mobile app interface with an orange background. At the top, there is a dark brown button labeled 'Presupuesto'. Below it, the following information is displayed:

Presupuesto:	32	€
Descuento:	30	%
Total:	22,4	€

At the bottom, there is a dark brown button labeled 'Reservar cita'.

❖ **Prueba 7:** visualizamos el importe de la cita

- Comprobamos si al cliente le corresponde un 30% de descuento al haber reservado 3 citas (o múltiplo de 3) en nuestro local. Lo hacemos de la siguiente manera:

En caso de que le corresponda el descuento, se indica por pantalla y se calcula el precio total.

❖ **Prueba 8:** reservamos la cita

- Comprobamos que no tiene citas sin pagar. De ser así, no dejamos reservar ninguna cita más hasta que no pague las que debe.

Si todo está correcto, procedemos a reservar la cita. Por el contrario, mostramos un mensaje haciendo saber que tiene citas sin pagar.

×

No se puede reservar esta cita, porque tiene citas anteriores sin pagar

Aceptar

En el siguiente formulario, también contamos con varias pruebas



The screenshot shows a web application interface for managing clients. At the top, there is a header bar with a back arrow on the left, the title 'Listado clientes' with a family icon in the center, and a logo on the right. Below the header, there is a search section with the label 'Filtrar por nombre:', a text input field, and a 'Buscar' button. The main content area contains a table with the following data:

Nombre	Apellidos	Teléfono	Correo
david	clemente	555-555-555	d@c.com
pepe	zarcero	562-545-622	p@z.com

Below the table, there are four buttons: 'Añadir cliente', 'Editar cliente', 'Eliminar cliente', and 'Mostrar citas'.

❖ **Prueba 9:** editar un cliente

- Comprobamos que haya un cliente seleccionado para poder editarlo.

Lo conseguimos mirando si en el listado hay algún ítem seleccionado. En caso de no ser así, no podemos continuar y mostramos un mensaje al usuario.



Debes seleccionar un cliente

Aceptar

❖ **Prueba 10:** eliminar un cliente

- Comprobamos que haya un cliente seleccionado para poder eliminarlo.

Lo conseguimos mirando si en el listado hay algún ítem seleccionado. En caso de no ser así, no podemos continuar y mostramos un mensaje al usuario.

×

Debes seleccionar un cliente

Aceptar

- Comprobamos que no tenga citas sin pagar. De ser así, no permitimos borrarlo hasta que no pague.

×

No se puede eliminar el cliente, porque tiene citas sin pagar

Aceptar

❖ **Prueba 11:** mostrar citas de un cliente.

- Comprobamos que haya un cliente seleccionado para poder consultar sus citas:

Lo conseguimos mirando si en el listado hay algún ítem seleccionado. En caso de no ser así, no podemos continuar y mostramos un mensaje al usuario.

×

Debes seleccionar un cliente

Aceptar

❖ **Prueba 12:** buscar un cliente por el filtro

- Comprueba que ese usuario exista en nuestra base de datos

De no ser así, muestra un mensaje indicándolo.



No se han encontrado clientes con ese nombre

Aceptar

En el siguiente formulario también podemos ver alguna prueba

Fecha	Hora	Servicios	Precio	Pagado
02/06/2022	9:00	Lavado, Alisado.	60	No

❖ **Prueba 13:** eliminar una cita

- Comprobamos que se haya seleccionado una cita.

Si no es así, mostramos un mensaje por pantalla



Debes seleccionar una cita para poder eliminarla

Aceptar

- Comprobamos que esa cita no está pagada. Ya que, si está pagada, es una cita archivada y de nada nos valdría eliminarla.


Si es así, mostramos un mensaje indicándolo





Esta cita está pagada y archivada. No se puede eliminar

Aceptar

Pasamos a la pantalla de nuevo cliente.



Nuevo Cliente 



Introduzca los siguientes datos para registrar un nuevo cliente:

Nombre:

Apellidos:

Teléfono:

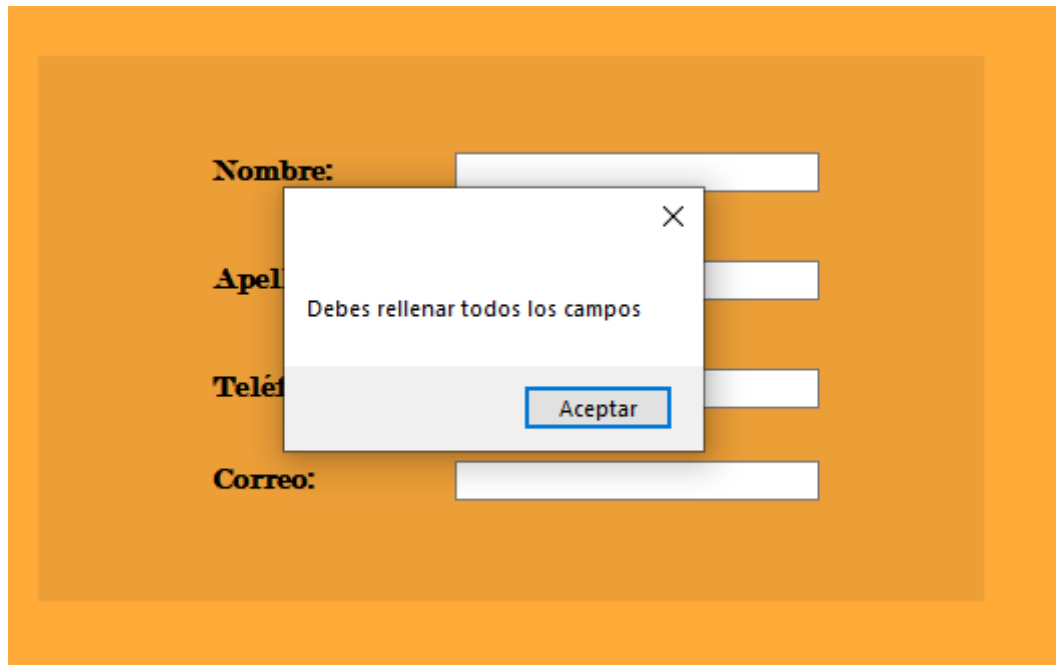
Correo:

Añadir

❖ **Prueba 14:** añadir un cliente

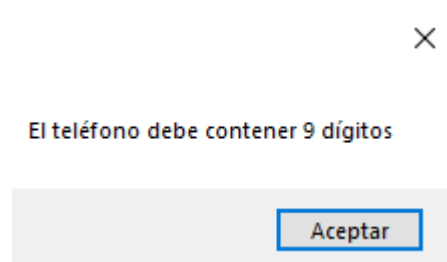
- Comprobamos que todos los campos tengan información.

De no ser así, lo mostramos por pantalla

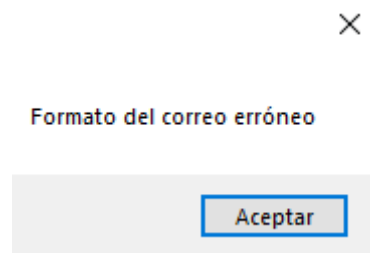


- Comprobamos que el campo teléfono tenga su formato correspondiente

De no ser así, mostramos un mensaje concreto.




- De la misma manera, validamos el formato del correo. Indicamos si no se cumplen los requisitos del email.





El formulario de editar un cliente tiene las mismas comprobaciones que en añadir, ya que es el mismo objeto con el que estamos programando y tiene los mismos requisitos.

- Comprobar que, a la hora de actualizar un cliente, estén todos los campos rellenos
- Comprobar que, el número de teléfono contenga 9 dígitos
- Comprobar que, el campo de email sea una dirección de correo válida.

Ahora pasamos al formulario de precios y servicios.



Precios y servicios 



Peluqueria

Servicio	Precio
Permanente	32
Mechas	18
Lavado	5
Corte	15,2
Recogido	18
Decoloracion	30
Tinte	22
Alisado	55

Uñas

Servicio	Precio
Acrilicas	18
Permanentes	12
Pedicura	10,5
De gel	16

Barbería

Servicio	Precio
Recorte	7
Afeitado	7

Depilación cera caliente

Servicio	Precio
Bigote	7
Cejas	5

Depilación láser

Servicio	Precio
Piernas	20
Zona intima	20
Axilas	12
Brazos	13

Modificar precio

❖ **Prueba 15:** modificamos el precio de un servicio

- Comprobamos que haya seleccionado un servicio.

De no ser así, lo indicamos por pantalla.



Elija un servicio para modificar su precio

Aceptar

Una vez seleccionamos un servicio y procedemos a modificar su precio

Modificar precio

Servicio Afeitado

Sector Barbería

Precio 7.00 €

Actualizar

- Comprobamos que a la hora de actualizar el precio (único campo editable) no quede vacío.

De ser así, informamos al usuario con un mensaje.



Completa el campo de precio

Aceptar

Por último, vamos a realizar pruebas de que todo se ha creado correctamente en nuestra base de datos.

❖ **Prueba 16:** nuevo cliente



Nuevo Cliente

Introduzca los siguientes datos para registrar un nuevo cliente:

Nombre: Sheila

Apellidos: Nieva

Teléfono: 655-965-236

Correo: sheila@gmail.com

Añadir

Aquí mostramos nuestra base de datos, con el cliente que acabamos de crear.

```
▼ — Cliente
  ▼ — 32f21c24-ffd2-4d20-a3b6-76f22cd8143c
    apellidos: "Nieva"
    citasAcumuladas: 1
    correo: "sheila@gmail.com"
    idCliente: "32f21c24-ffd2-4d20-a3b6-76f22cd8143c"
    idEmpresa: "d0f1f34c-6f4f-4dca-834b-f2cac4fc87e0"
    nombre: "Sheila"
    telefono: "655-965-236"
```

❖ **Prueba 17:** Actualizamos el cliente con un nuevo teléfono y correo

Cliente

Modifica los campos que desee:

Nombre: Sheila

Apellido: A

Telefono: T

Correo: sheila_nuevo@gmail.com

Cliente Sheila se ha modificado con éxito

Aceptar

Actualizar cliente

Y vemos como en nuestra base de datos, también se actualizan los cambios.

▼	—	Cliente
	▼	32f21c24-ffd2-4d20-a3b6-76f22cd8143c + 🗑
		— apellidos: "Nieva"
		— citasAcumuladas: 1
		— correo: "sheila_nuevo@gmail.com"
		— idCliente: "32f21c24-ffd2-4d20-a3b6-76f22cd8143c"
		— idEmpresa: "d0f1f34c-6f4f-4dca-834b-f2cac4fc87e0"
		— nombre: "Sheila"
		— telefono: "555-666-222"

Ahora vamos a proceder a registrar una cita para este cliente que acabamos de crear.

❖ Prueba 18: nueva cita

Servicios

Presupuesto

Presupuesto: 57 €

Descuento: 0 %

Total: 57 €

Cita añadida con éxito

Aceptar

Reservar cita

Fecha: 15/06/2022

Añadir

Selección la hora deseada: 9:30

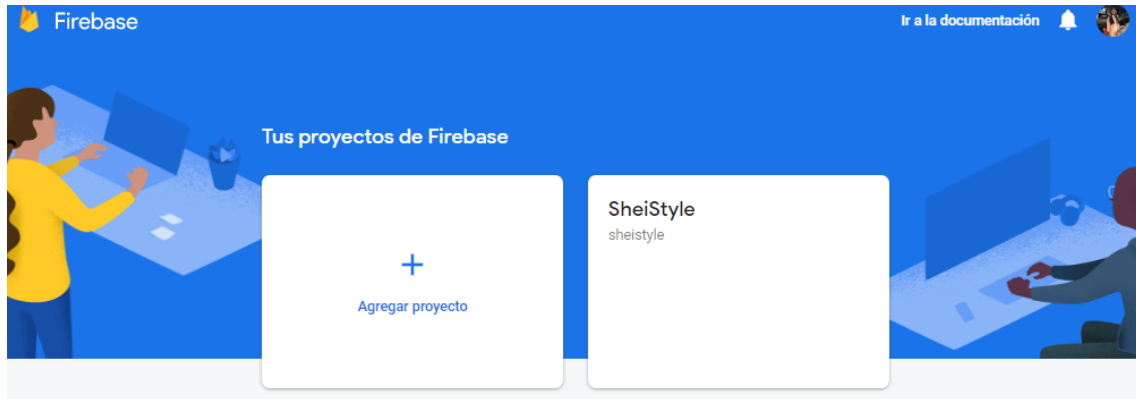
Elija un cliente: Sheila Nieva

Y en nuestra base de datos, como esperábamos, se ha creado la cita con los datos recogidos.

```
▼ Cita
  ▼ 85648f18-1972-4ff9-8157-da8c683305c4
    duracion: 2.5
    fecha: "2022-06-15T09:30:00"
    idCita: "85648f18-1972-4ff9-8157-da8c683305c4"
    idCliente: "32f21c24-ffd2-4d20-a3b6-76f22cd8143c"
    pagado: false
    precioCita: 57
    servicio: "Permanente,Lavado,Piernas,"
```

7. Lanzamiento y puesta en marcha

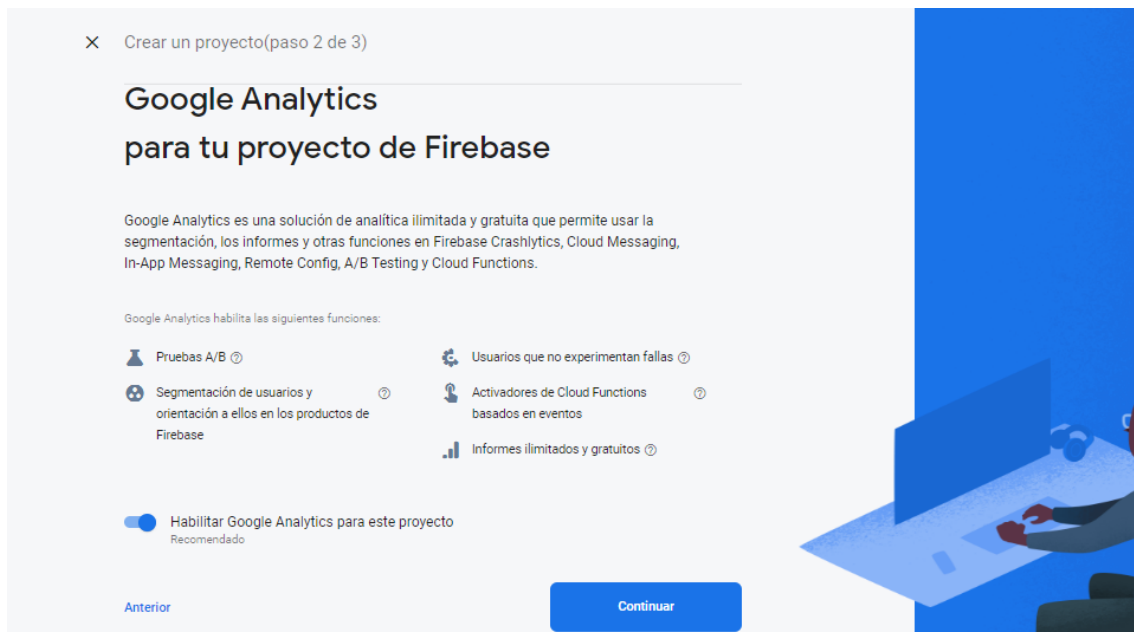
En cuanto al despliegue y puesta en marcha de nuestro sistema, Firebase nos lo pone realmente fácil, ya que solo necesitamos de una cuenta de Google para poder iniciar sesión. Una vez iniciada la sesión podremos crear hasta tres proyectos.



En la creación de un nuevo proyecto debemos introducir en primer lugar el nombre del mismo.

The image shows a dialog box titled 'Crear un proyecto(paso 1 de 3)'. The main heading inside is 'Comencemos con el nombre de tu proyecto' with a question mark icon. Below this is the instruction 'Ingresa el nombre de tu proyecto'. There is a text input field containing 'My Project'. Above the input field, there is a link that says 'Agrega Firebase a uno de tus proyectos de Google Cloud existentes' and another link 'Más información'. At the bottom of the dialog box is a 'Continuar' button.

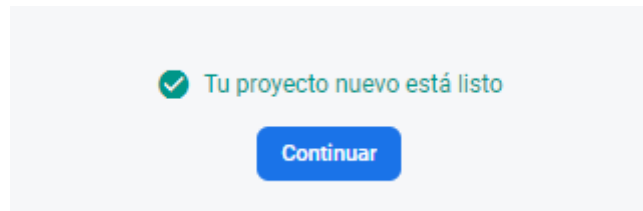
El segundo paso para poder crear nuestro proyecto en Firebase, es configurar google Analytics.



Por último, las últimas configuraciones para Google Analytics



Una vez hayamos completado todos los pasos y aceptado las condiciones de Google Analytics, el proyecto se nos crea adecuadamente.

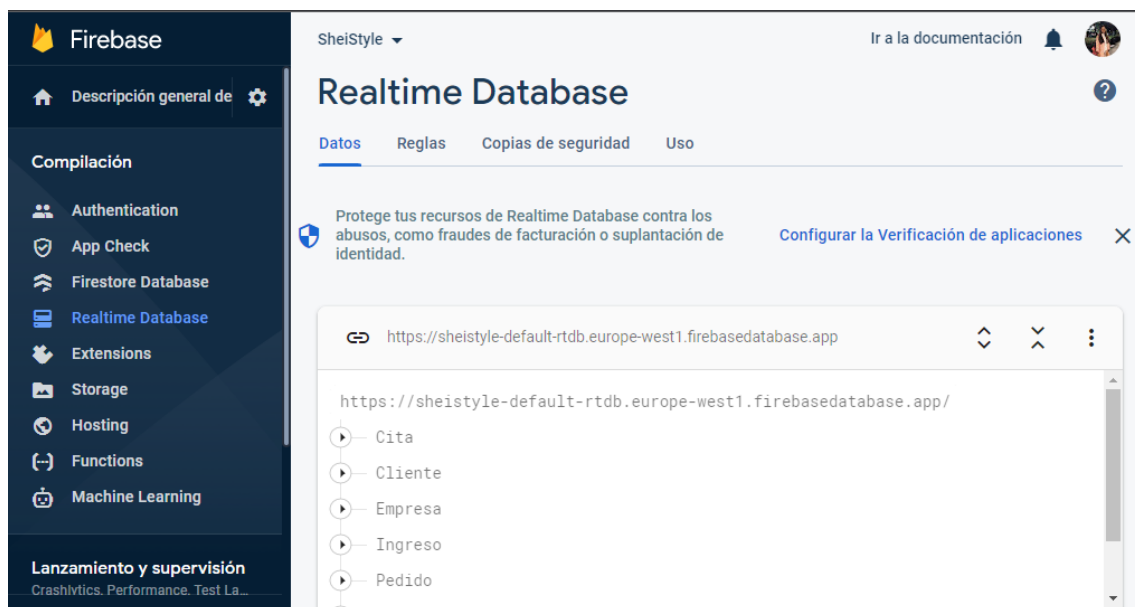


Ahora añadiríamos nuestra base de datos para poder meter nuestras tablas e ir guardando los datos de nuestra aplicación.

En nuestro caso utilizaremos Realtime Database para tener una base de datos en tiempo real.



Nuestra base de datos ya creada con las tablas y algunos datos ya registrados por nuestra aplicación tiene una vista tal que así:



8.Manual de uso

A continuación, les dejo el enlace para el manual de uso:

<https://github.com/SheilaNieva/SheiStyleSNL/blob/master/SheiStyleSNL/Documentacion/Manual%20de%20uso%20SheiStyle.pdf>

9.Valoración y conclusiones

Una vez llegados a este punto, llega la hora de echar la vista atrás y ver hasta donde hemos sido capaces de llegar.

Este trabajo de estos meses, he de decir que ha sido un poco duro y frustrante a la vez que satisfactorio. Cuando llegó el momento de elegir la idea del proyecto, tenía la mente en blanco, pero al cabo de unos días se me vino a la cabeza la idea de llevar la gestión de un salón de belleza. Al principio, esa idea estaba cogida con pinzas y hasta que tomó forma y pude darle color, hubo momentos de mucha tensión y agobio. Pero al final, con esfuerzo todo se consigue. Una vez supe ver más o menos mi aplicación y que mi tutora me diese el visto bueno, se me fueron ocurriendo ideas y puntos para ir añadiendo a mi proyecto. Todo esto, teniendo siempre en cuenta mis capacidades, mi experiencia y sobretodo, el tiempo estimado por la fecha de entrega.

A pesar de que todo esto ha sido un reto como bien he sugerido antes, me ha servido de mucho estos años de aprendizaje en este curso, aunque también he tenido que dedicar tiempo al autoaprendizaje de ciertas tecnologías como por ejemplo Firebase. Y por supuesto, el coger más confianza, soltura y comodidad al usar Git, herramienta con la cual nos han 'machacado' mucho en el curso y que no veíamos las facilidades que nos puede presentar en un proyecto como este.

El estar realizando FCT en el mismo tiempo, ha resultado algo difícil ya que hay días que acabas muy saturada y tienes que seguir con el proyecto. Pero en mi caso, también ha ayudado por ejemplo para aprender cosas de git, y ver su uso en la vida real en un trabajo diario.

A pesar de la ayuda de profesores y resolver dudas en las tutorías creo que este proyecto me ha servido para crecer un poquito más como desarrolladora e investigar y poner en práctica todo lo aprendido en estos años, sin una ayuda continua y 'vigilancia' de personas superiores a mí.

En conclusión, a pesar de todos los sentimientos y frustraciones encontradas a lo largo de estos meses, estoy muy contenta con mi trabajo y valoro muy positivamente esta experiencia.

10. Bibliografía

<https://blog.mailrelay.com/es/2018/09/06/estudio-de-mercado>

<https://dafo.ipyme.org/Home#&&g=en-que-consiste>

<https://www.obsbusiness.school/blog/como-hacer-un-estudio-de-mercado-en-4-pasos>

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

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

<https://docs.github.com/es/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards>

<https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>

<https://timelines.gitkraken.com/>

<https://www.gladysgbegnedji.com/estimar-recursos-de-las-actividades/>

<https://www.gladysgbegnedji.com/estimar-los-costos-del-proyecto/>

<https://firebase.google.com/?hl=es>

<https://www.wrike.com/es/project-management-guide/faq/que-es-la-estimacion-de-costes-en-gestion-de-proyectos/>

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

<https://diagramasuml.com/diagrama-de-clases/>

[https://es.wikipedia.org/wiki/Persistencia_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Persistencia_(inform%C3%A1tica))

<https://es.wikipedia.org/wiki/NoSQL#Ventajas>

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

<https://blog.codmind.com/que-es-spring-boot/>

<https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/#::~:~:text=basado%20en%20JavaScript.-,Node.,procesos%2C%20pues%20no%20hay%20bloqueos>

<https://es.wikipedia.org/wiki/Firebase#Desarrollo>

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

<https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/integracion/>

<https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/sistema/>

<https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/acceptacion/>

<https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/regresion/>