

PROPUESTA DEL PROYECTO

LA CESTA VERDE

MARIO ALONSO NÚÑEZ

JOSE EDUARDO RAIMUNDO FERNANDO

ALEJANDRO LEAD CASTAÑO

Tabla de contenido

1. Propuesta del proyecto	4
1.1 Título	4
1.2 Número de grupo	4
1.3 Participantes.....	4
1.4 Motivación	4
1.5 Objetivos	4
1.6 Análisis del sistema a implementar.....	5
1.7 Aproximación de diseño.....	9
2. Tabla de fechas.....	10
3. Análisis de requisitos.....	11
3.1 Definición de Objetivos	11
3.2 Definición de Actores	13
3.3 Definición de requisitos Funcionales	14
3.4 Definición de requisitos No Funcionales.....	51
4. Desarrollo del proyecto.....	53
4.1 Paso 1: Definición de herramientas y estructura.....	54
4.2 Paso 2: Diseño de la base de datos	56
4.3 Paso 3: Estructura del proyecto Backend.....	58
4.4 Paso 4: Funcionamiento general del Backend	60
4.5 Paso 5: Obtención de la información nutricional.....	64
4.6 Paso 6: Estructura del proyecto Frontend	65
4.7 Paso 7: Creación del Frontend y navegación entre pantallas	67
4.8 Paso 8: Maquetación de escenas del Frontend para sistemas móviles.....	70
4.9 Paso 9: Desarrollo de componentes propios	73
4.10 Paso 10: Maquetación de las escenas del Frontend para el modo Escritorio	76
5. Manual de desarrollador.....	79
5.1 Peticiones del backend.....	79
5.1.1 POST: Registro de usuario	79
5.1.2 POST: Acceso	81
5.1.3 POST: Registro de negocio	83
5.1.4 GET: Obtener negocios registrados.....	84
5.1.5 POST: Registro de categorías	85
5.1.5 GET: Obtener categorías	85
5.1.6: POST: Registro de producto	86
5.1.7: GET: Obtener productos	87

5.1.8: GET: Obtener información nutricional	88
6. Conclusiones.....	89
7. Bibliografía	90

1. Propuesta del proyecto

1.1 Título

La cesta verde

1.2 Número de grupo

Grupo 4

1.3 Participantes

Mario Alonso Núñez

José Eduardo Raimundo Fernando

Alejandro Leal Castaño

1.4 Motivación

Nos encontramos en un mundo cada vez más globalizado, donde el acceso a los productos es cada vez más sencillo y la diversidad de estos es cada vez mayor. Podemos decir sin temor a equivocarnos que nos encontramos en la era de la globalización, pues en la misma ciudad donde vivimos podemos encontrar elementos provenientes desde la otra punta del mundo, esperando para estar en nuestras casas.

Sin embargo, aunque esto trae consigo una gran cantidad de factores positivos, muchas veces podemos quedar cejados por la gran cantidad de luces que vemos en nuestro camino, encontrándonos en la tesitura de que el bosque no nos permite ver el árbol.

Esto es algo que ocurre en nuestro día a día con una gran cantidad de elementos que están y han estado durante mucho más tiempo del que podemos imaginar. No necesitamos pensar a lo grande, sino en las pequeñas cosas que han acompañado la vida de la gente desde siempre y que actualmente se encuentran a punto de desaparecer debido a esa ceguera colectiva.

Esta es la situación de los pequeños negocios locales que se encuentran, sobre todo, en las medianas y grandes ciudades del mundo moderno. Nos referimos a todas aquellas tiendas que venden bienes de consumo de primera necesidad y que lo han hecho durante mucho tiempo, las cuales ahora luchan por intentar sobrevivir debido a carecer de un altavoz lo suficientemente grande como para poder recordarnos que siguen ahí.

Por otra parte, la compra de productos en las grandes superficies ha provocado una gran reducción del consumo de productos locales, los cuales son más ecológicos y fáciles de producir, debido a necesitar de una menor gestión para poder llegar hasta nosotros.

Por estos motivos decidimos plantear *La cesta verde*, la cual será una plataforma orientada a los pequeños comercios locales, donde estos puedan anunciar sus productos y competir con las grandes marcas. De esta manera podremos volver a poner los sanos productos locales al alcance de los consumidores, además de evitar que sigan muriendo los negocios que nos han acompañado durante toda la vida.

1.5 Objetivos

El proyecto será desarrollado utilizando herramientas de desarrollo web, sin embargo, no se pretende desarrollar una simple página de compra de productos, sino que los esfuerzos se

enfocaran en llevar a cabo una aplicación híbrida. Esto conlleva dos aspectos fundamentales en lo que a la interacción con el usuario se refiere:

- El producto debe poder ser ejecutado tanto en un buscador web como en dispositivos inteligentes y equipos de sobremesa.
- La interacción del usuario con la aplicación debe poder ser dinámica dependiendo del tamaño de la pantalla donde se estén visualizando, lo que conllevará diseñar la aplicación para que esta se presente correctamente independientemente de la resolución utilizada por el dispositivo.

Además de esto, como elementos básicos de interacción de usuario se desarrollarán los siguientes aspectos:



- Implementación de tokens para lograr la autenticación automática de ellos usuarios en la aplicación, permitiendo a los mismos poder navegar por esta tranquilamente sin la necesidad de introducir sus credenciales cada vez que se recargue la misma.
- Capacidad del usuario para registrar datos en la aplicación, de modo que se facilite su interacción con la misma a no necesitar volver a registrarlos nuevamente. Esto hace referencia a elementos como: La cesta de la compra, los datos de perfil, direcciones de facturación etc.

En lo que respecta al uso de tecnologías multimedia, se plantean los siguientes aspectos a desarrollar:

- Correcta interacción de usuario con fotos que permitan visualizar los elementos con los que se está tratando.
- Capacidad de los negocios de indicar la dirección donde se encuentra, la cual se mostrará en la aplicación a través de una posición mediante herramienta de mapas.
- Capacidad del sistema para asociar automáticamente la tabla nutricional relacionada con los alimentos que se encuentran a disposición de los usuarios en la plataforma.

1.6 Análisis del sistema a implementar

La funcionalidad del sistema desarrollar se mostrará mediante el siguiente esquema WBS, donde los diferentes nodos del árbol se encuentran organizados siguiendo el siguiente código de colores:

-  Nodo interno del producto mínimo vital.
-  Paquete de trabajo del producto mínimo vital.

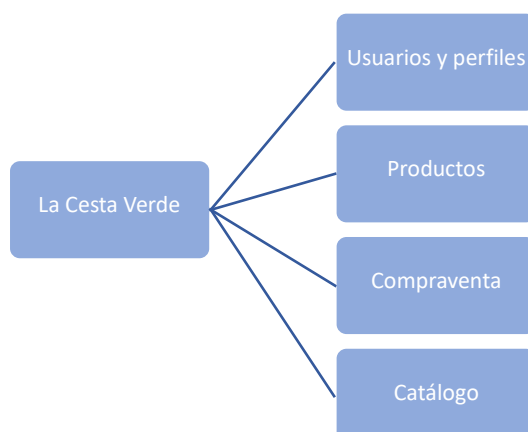


Ilustración 1: WBS del proyecto

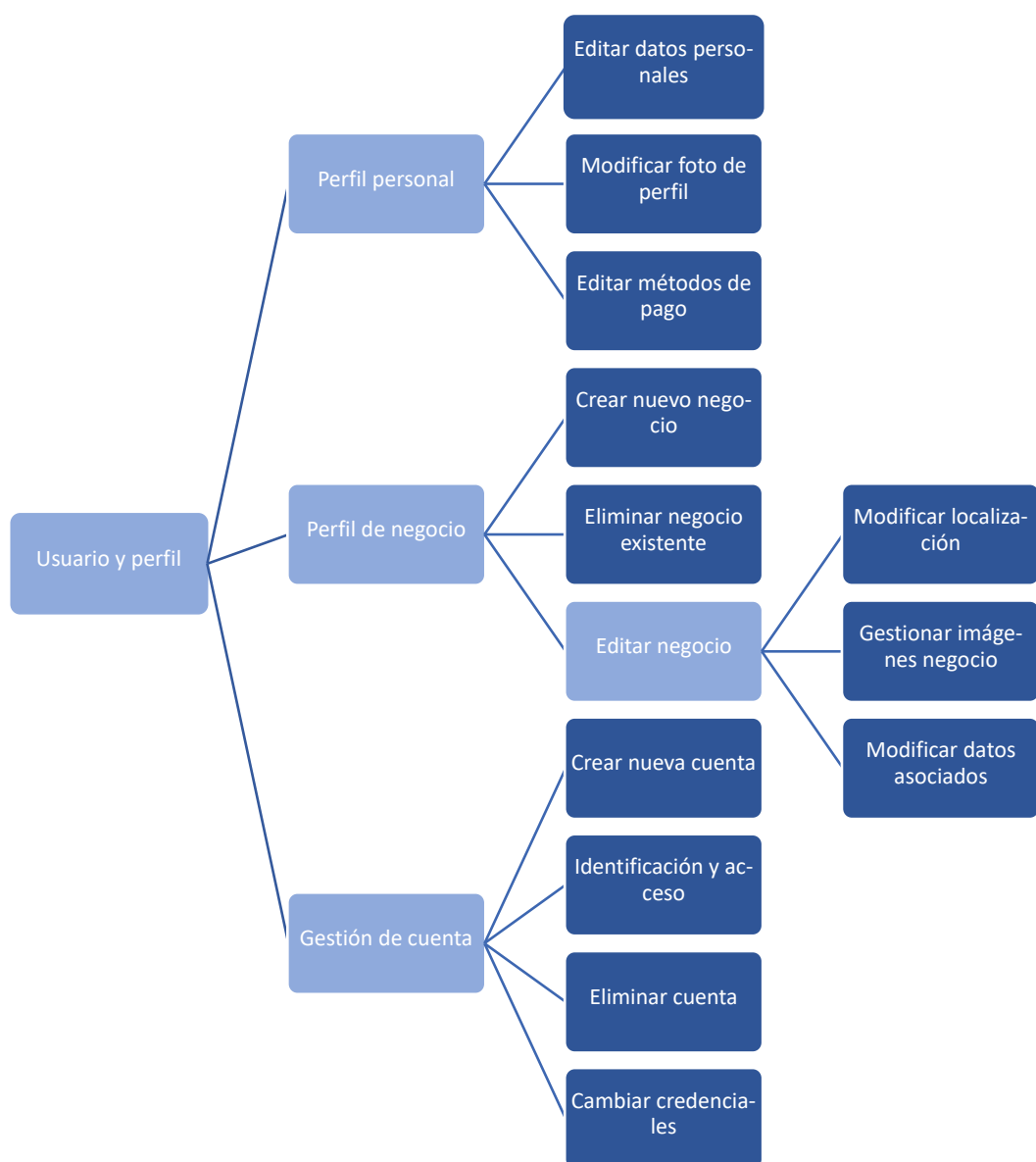


Ilustración 2: Epic 1 - Usuarios y perfiles

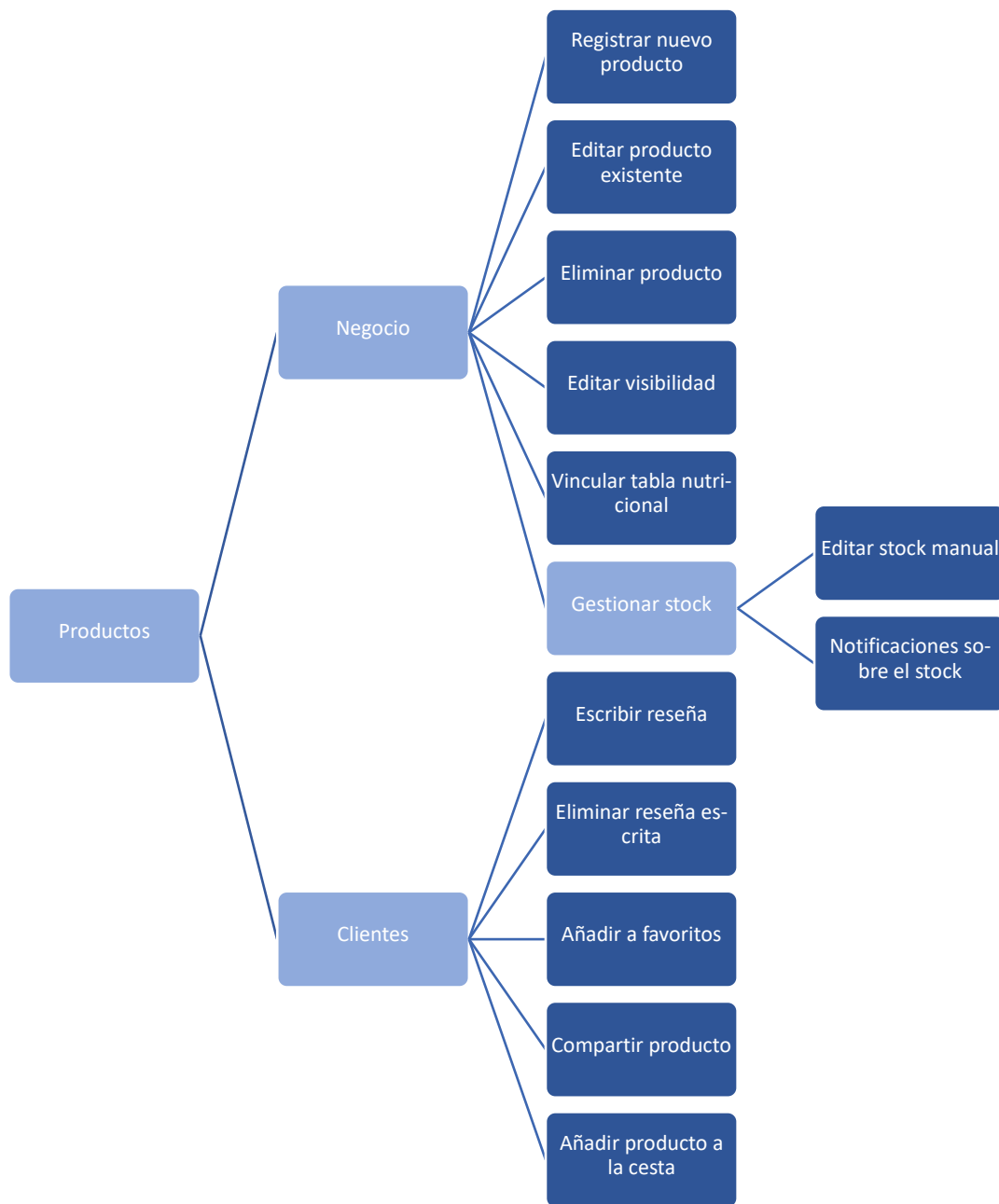


Ilustración 3: Epic 2 - Productos



Ilustración 5: Epic 3 - Compraventa

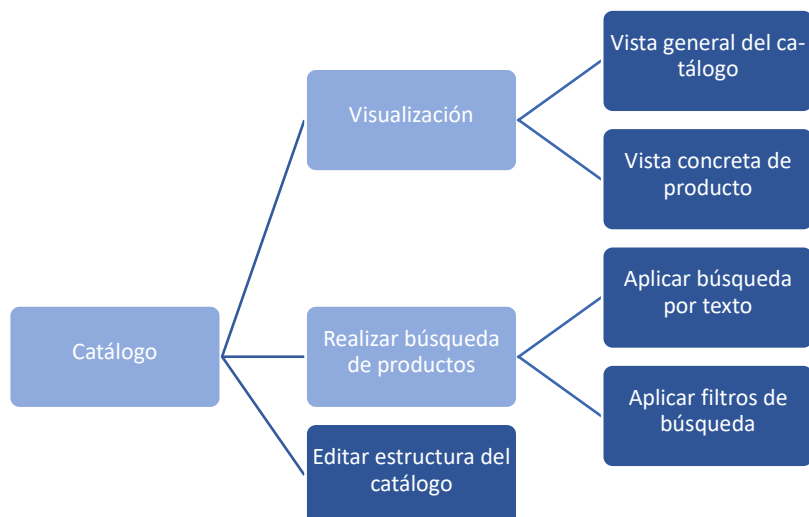


Ilustración 4: Epic 4 - Catálogo

1.7 Aproximación de diseño

Para llevar cabo el desarrollo de la aplicación híbrida se plantea el uso de las siguientes herramientas:

- **Ionic:** Framework de desarrollo que permite la transcripción de aplicaciones desarrolladas mediante diferentes Frameworks webs a otras plataformas como móviles, aplicaciones de escritorio, televisiones, etc.
- **Angular:** Framework para el desarrollo de aplicaciones que web utilizado para crear la estructura de la aplicación.
- **Node.js:** Herramienta empleada en el desarrollo de backend. Dispone de varios Frameworks de desarrollo para la creación y gestión específica de peticiones al servidor.
- **MongoDb:** Herramienta para la definición y gestión de bases de datos no relacionales.
- **JasonWebToken:** Herramienta empleada en el cifrado de tokens para mantener la identificación del usuario mientras este navega dentro de la aplicación.



Ilustración 8: Ionic icono



Ilustración 7: Angular icono

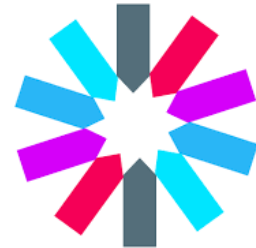


Ilustración 6: JasonWebToken icono



Ilustración 10: MongoDB icono



Ilustración 9: Node.js icono

2. Tabla de fechas

Fecha	Descripción
13-02-2023	Formación del grupo
18-02-2023	Realización de mockup para dispositivos móviles la propuesta
24-02-2023	Presentación de la propuesta del proyecto
28-02-2023	Maquetación de escenas del Frontend mediante los elementos de Ionic
07-03-2023	Diseño de la base de datos del proyecto
10-03-2023	Desarrollo del backend
15-03-2023	Desarrollo de la navegación entre las pantallas del Frontend
18-03-2023	Maquetación de las pantallas del Frontend – 1
21-03-2023	Maquetación de las pantallas del Frontend – 2
22-03-2023	Realización del mockup para dispositivos de escritorio

3. Análisis de requisitos

En la presente sección realizaremos un análisis de los diferentes tipos de requisitos que componen el proyecto desde un marco correspondiente a la fase de análisis.

3.1 Definición de Objetivos

OBJ-001	Gestión de usuarios y perfiles
Versión	Sin numerar
Autores	Cliente que encarga la aplicación
Fuentes	Usuarios
Descripción	El sistema
Subobjetivos	<p>El sistema debe gestionar correctamente la información referente a cada uno de los usuarios de la aplicación, permitiendo la modificación de estos y la correcta adecuación de su presentación al usuario. Esto conlleva la proporción del soporte y la accesibilidad necesarias para la creación de nuevas cuentas por parte de futuros usuarios.</p> <p>Del mismo modo, la plataforma debe asegurar la correspondencia uno a uno de los usuarios con los datos correspondientes a su perfil, asegurando así que estos únicamente puedan ser accesibles por el usuario propietario de la cuenta.</p>
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo
Estabilidad	alta
Comentarios	Ninguno

OBJ-002	Gestión productos
Versión	Sin numerar
Autores	Cliente que encarga la aplicación
Fuentes	Usuarios
Descripción	El sistema
Subobjetivos	<p>El sistema debe gestionar correctamente la información asociada a los diferentes productos que son registrados en la aplicación por parte de los usuarios de esta. Esto incluye acciones como la inclusión de nuevos productos, eliminación de los ya existentes o modificación de los datos asociados.</p> <p>Los productos registrados en la aplicación deben estar asociados a la cuenta de usuario que los ha registrado de manera unitaria. De este modo, se asegura que cada producto únicamente puede ser propiedad de un usuario.</p>
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo

Estabilidad	alta
Comentarios	Ninguno

OBJ-003	Gestión de compraventa
Versión	Sin numerar
Autores	Cliente que encarga la aplicación
Fuentes	Usuarios
Descripción	<p>El sistema debe proporcionar a los usuarios de la aplicación las acciones necesarias para poder llevar a cabo las diferentes acciones de compraventa dentro del sistema. Estas acciones deberán ser intuitiva y mostrarse de manera adecuada al usuario con el fin de minimizar los posibles errores y confusiones derivadas de las mismas.</p> <p>De la misma manera, este proceso debe poder realizarse de manera confidencial, asegurando que los datos no puedan ser accesibles por terceros.</p>
Subobjetivos	El sistema deberá
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo
Estabilidad	alta
Comentarios	Ninguno

OBJ-004	Gestión del catálogo
Versión	Sin numerar
Autores	Cliente que encarga la aplicación
Fuentes	Usuarios
Descripción	<p>El sistema gestionar correctamente la información mostrada al usuario referente a los diferentes productos registrados en la aplicación, dentro de la sección de catálogo.</p> <p>La información visualizada en el catálogo debe de corresponderse con la que pertenece al producto al cual se muestra vinculada, asegurando así una fidelidad a la hora de mostrar dicha información.</p>
Subobjetivos	El sistema deberá
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo
Estabilidad	alta
Comentarios	Ninguno

3.2 Definición de Actores

AC-001	Usuario desconocido
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Personas que no han sido identificadas en la aplicación
Descripción	Persona que no han pasado por el proceso de identificación, ya sean por que han decidido no hacerlo o porque no poseen una cuenta con la que identificarse. Los usuarios desconocidos no pueden navegar por las secciones privadas de la aplicación.
Comentarios	Este actor puede pasar a ser un actor <i>Usuario</i> mediante la realización del proceso de autenticación.

AC-002	Usuario
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Personas pertenecientes a cuerpos de servicios de emergencias
Descripción	Persona que se encuentra identificada en la aplicación mediante el uso de una cuenta registrada previamente en la misma. Los usuarios pueden navegar por las secciones privadas de la aplicación, pero únicamente pueden acceder a las secciones públicas si siguen el proceso de Cerrar Sesión.
Comentarios	Este actor puede pasar a ser un actor <i>Usuario desconocido</i> si realiza el proceso de Cerrar Sesión en la aplicación

3.3 Definición de requisitos Funcionales

UC-001	Editar datos personales
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar los datos personales asociados a su perfil.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar los datos personales en su perfil. El caso de uso comienza. 2. El sistema muestra al usuario el formulario requerido para modificar sus datos personales. 3. El usuario modifica los datos deseados en el formulario. 4. El sistema verifica los datos y actualiza la información asociada a los datos personales del usuario en la aplicación. El caso de uso finaliza.
Postcondición	Los datos personales del usuario deben haber sido actualizados en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos modificados no son correctos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-002	Editar foto de perfil
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar la foto asociada a su perfil.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar la foto en su perfil. El caso de uso comienza. 2. El sistema muestra al usuario el formulario para indicar la nueva foto. 3. El usuario indica la nueva foto en el formulario. 4. El sistema verifica la imagen indicada y la actualiza como nueva foto de perfil. El caso de uso finaliza.
Postcondición	La foto de perfil del usuario debe haber sido actualizada en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si la foto indicada no es aceptable, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-003	Editar métodos de pago
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar los métodos de pago asociados a su perfil.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar los métodos de pago asociados a su perfil. El caso de uso comienza. 2. El sistema muestra al usuario el formulario para modificar los métodos de pago. 3. El usuario modifica los datos indicados en el formulario. 4. El sistema verifica los datos introducidos por el usuario y actualiza los métodos de pago. El caso de uso finaliza.
Postcondición	Los métodos de pago del usuario deben haber sido actualizada en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si la información introducida no es correcta, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-004	Crear nuevo negocio
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere crear un nuevo negocio asociado a su perfil.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar que quiere crear un nuevo negocio asociados a su perfil. El caso de uso comienza. 2. El sistema muestra al usuario el formulario crear un nuevo negocio. 3. El usuario introduce los datos indicados en el formulario. 4. El sistema verifica los datos introducidos y crea un nuevo negocio asociado al perfil del usuario. El caso de uso finaliza.
Postcondición	El nuevo negocio debe crearse y vincularse al perfil del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si la información introducida no es correcta, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-005	Eliminar negocio existente
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere crear un negocio existente y vinculado a su perfil.
Precondición	El usuario debe estar registrado en la aplicación y el negocio debe existir previamente y estar asociado al perfil del usuario que lo quiere eliminar.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá a su perfil e indica que quiere eliminar un negocio existente vinculado al mismo. El caso de uso comienza. 2. El sistema muestra al usuario un mensaje de confirmación. 3. El usuario verifica la eliminación. 4. El sistema elimina el negocio seleccionado por el usuario y todos los productos asociados al mismo. El caso de uso finaliza
Postcondición	El nuevo negocio debe crearse y vincularse al perfil del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 3: Si el usuario cancela la eliminación, el caso de uso finaliza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-006	Modificar localización de negocio
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar la localización de un negocio asociado a su perfil.
Precondición	El usuario debe estar registrado en la aplicación y el negocio debe existir previamente y estar asociado al perfil del usuario que lo quiere eliminar.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar la localización correspondiente a un negocio asociado a su perfil, en la página del propio negocio. El caso de uso comienza. 2. El sistema muestra al usuario el formulario para indicar la nueva dirección. 3. El usuario indica la nueva dirección en el formulario. 4. El sistema verifica la dirección indicada y la actualiza como nueva dirección del negocio en cuestión. El caso de uso finaliza.
Postcondición	La dirección del negocio debe haber sido actualizada en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si la dirección indicada no es aceptable, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-007	Gestionar imágenes de negocio
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere gestionar las imágenes asociadas a un negocio, ya sea para añadir nuevas o eliminar alguna de las existentes.
Precondición	El usuario debe estar registrado en la aplicación y el negocio debe existir previamente y estar asociado al perfil del usuario que lo quiere eliminar. En el caso de que se quiera eliminar una imagen, esta debe haber sido asociada previamente con el negocio en cuestión.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar realizar cambios en las imágenes asociadas a un negocio en concreto asociado a su perfil, en la página de dicho negocio. El caso de uso comienza. 2. El sistema muestra al usuario un panel donde poder realizar las modificaciones en la galería de fotos. 3. El usuario realiza las modificaciones deseadas. 4. El sistema actualiza la galería de imágenes asociadas al negocio. El caso de uso finaliza.
Postcondición	Las imágenes asociadas al negocio deben ser actualizadas siguientes las decisiones del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 3: Si la alguna de las modificaciones no es aceptable, el sistema lo notificará al usuario mientras esta se realiza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-008	Modificar datos asociados al negocio
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar los datos de un negocio asociado a su perfil.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar los datos de un negocio asociado a su perfil en la página del propio negocio. El caso de uso comienza. 2. El sistema muestra al usuario el formulario requerido para modificar los datos del negocio. 3. El usuario modifica los datos deseados en el formulario. 4. El sistema verifica los datos y actualiza la información asociada a los datos del negocio en cuestión. El caso de uso finaliza.
Postcondición	Los datos personales del negocio deben haber sido actualizados en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos modificados no son correctos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-009	Crear nueva cuenta
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario desconocido quiera crear una nueva cuenta en la aplicación.
Precondición	El usuario debe ser desconocido, es decir, no debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario desconocido accede a la página de acceso de la aplicación e indica que quiere crear una nueva cuenta. El caso de uso comienza. 2. El sistema muestra al usuario el formulario requerido para crear la nueva cuenta 3. El usuario introduce los datos necesarios en el formulario. 4. El sistema verifica los datos y crea una nueva cuenta en el sistema. El caso de uso finaliza.
Postcondición	El sistema debe crear una nueva cuenta con los datos proporcionados por el usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos modificados no son correctos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-010	Identificación y acceso
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario desconocido quiera acceder a la aplicación a través de una cuenta registrada previamente.
Precondición	El usuario debe ser desconocido, es decir, no debe estar registrado en la aplicación. La cuenta con la que se realizará el acceso debe haber sido registrada previamente en el sistema.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario desconocido accede a la página de acceso de la aplicación e indica que quiere identificarse y acceder al sistema. El caso de uso comienza. 2. El sistema muestra al usuario el formulario requerido para crear la nueva cuenta 3. El usuario introduce los datos necesarios en el formulario. 4. El sistema verifica los datos y realiza la identificación y acceso del usuario. El caso de uso finaliza.
Postcondición	El usuario estará identificado en el sistema y este permitirá su acceso a las secciones restringidas a usuarios identificados.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos modificados no son correctos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-011	Eliminar cuenta
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere eliminar su cuenta del sistema.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá a su perfil e indica que quiere eliminar su cuenta. El caso de uso comienza. 2. El sistema muestra al usuario un panel de confirmación. 3. El usuario introduce los datos necesarios y verifica la eliminación. 4. El sistema elimina la cuenta del usuario y todos los datos asociados a la misma, incluyendo los negocios y productos existentes. El caso de uso finaliza
Postcondición	La cuenta del usuario ha sido eliminada del sistema y el usuario redireccionando a la sección de acceso.
Excepciones	<ul style="list-style-type: none"> • Paso 3: Si el usuario cancela la eliminación, el caso de uso finaliza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-012	Cambiar credenciales
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-001: Gestión de usuarios y perfiles
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar las credenciales de acceso asociada a su perfil.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar las credenciales de acceso asociadas a su perfil. El caso de uso comienza. 2. El sistema muestra al usuario el formulario para introducir las nuevas credenciales de acceso. 3. El usuario introduce las nuevas credenciales de acceso. 4. El sistema verifica los datos introducidos y la actualiza las credenciales de acceso asociadas al perfil. El caso de uso finaliza.
Postcondición	Las credenciales de acceso asociadas al perfil de usuario deben haber sido actualizada en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos introducidos no son aceptables, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-013	Registrar nuevo producto
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario quiera registrar un nuevo producto en la aplicación, el cual pertenecerá a un negocio que se encuentre vinculado a su cuenta.
Precondición	El usuario debe estar registrado previamente en la aplicación y tener vinculado mínimamente un negocio a su cuenta.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá al perfil vinculado a su negocio donde quiere agregar el nuevo producto, selecciona la opción de añadir un producto nuevo. El caso de uso comienza. 2. El sistema muestra al usuario el formulario requerido para registrar el nuevo producto. 3. El usuario introduce los datos necesarios en el formulario. 4. El sistema verifica los datos y crea el nuevo producto indicado por el usuario. El caso de uso finaliza.
Postcondición	El nuevo producto deberá estar registrado en la aplicación y vinculado al negocio existente seleccionado por el usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos introducidos son erróneos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-014	Editar producto existente
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario quiera editar la información vinculada a un producto ya existente en la aplicación y vinculado a un negocio asociado a dicho usuario.
Precondición	El producto a editar debe haber sido registrado previamente en la aplicación y estar asociado a uno de los negocios vinculados al usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá al perfil vinculado a su negocio donde se encuentra el producto, accede al mismo e indica que quiere editarlo. El caso de uso comienza. 2. El sistema muestra al usuario el formulario requerido para editar los datos asociados al producto. 3. El usuario modifica los datos del formulario. 4. El sistema verifica los datos y modifica la información asociada al producto. El caso de uso finaliza.
Postcondición	Los datos referentes al producto modificado por el usuario deben haber sido actualizados correctamente en la aplicación.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos introducidos son erróneos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-015	Eliminar producto
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario quiera eliminar un producto ya existente en la aplicación y vinculado a un negocio asociado a dicho usuario.
Precondición	El producto a eliminar debe haber sido registrado previamente en la aplicación y estar asociado a uno de los negocios vinculados al usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá al perfil vinculado a su negocio donde se encuentra el producto, accede al mismo e indica que quiere eliminarlo. El caso de uso comienza. 2. El sistema muestra al usuario un mensaje de confirmación. 3. El usuario verifica la eliminación. 4. El sistema elimina el producto seleccionado por el usuario. El caso de uso finaliza.
Postcondición	El producto indicado por el usuario debe haber sido eliminado de la aplicación.
Excepciones	<ol style="list-style-type: none"> 1. Paso 3: Si el usuario no verifica la acción, el caso de uso finalizará.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-016	Editar visibilidad
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario quiera modificar la visibilidad de un producto ya existente en la aplicación y vinculado a un negocio asociado a dicho usuario.
Precondición	El producto en cuestión debe haber sido registrado previamente en la aplicación y estar asociado a uno de los negocios vinculados al usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá al perfil vinculado a su negocio donde se encuentra el producto, accede al mismo y modifica directamente su visibilidad. El caso de uso comienza. 2. El sistema actualiza la visibilidad del producto en cuestión. El caso de uso finaliza.
Postcondición	La visibilidad del producto en cuestión debe haber sido actualizada en la aplicación.
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 1. Tiempo requerido por el usuario. 2. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-017	Vincular tabla nutricional
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario quiera vincular la tabla nutricional de un producto ya existente en la aplicación y vinculado a un negocio asociado a dicho usuario.
Precondición	El producto en cuestión debe haber sido registrado previamente en la aplicación y estar asociado a uno de los negocios vinculados al usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario realiza el proceso para registrar un nuevo producto en la aplicación, rellendo el formulario indicado por el sistema. El caso de uso comienza. 2. El sistema analiza los datos introducidos por el usuario y los utiliza para generar la tabla nutricional vinculado al nuevo producto. EL caso de uso finaliza.
Postcondición	El nuevo producto registrado por el usuario debe tener vinculada la tabla nutricional generada por el sistema.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos introducidos son erróneos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Tiempo requerido por el usuario. 2. Inmediato
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-018	Editar stock manual
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario quiera eliminar un producto ya existente en la aplicación y vinculado a un negocio asociado a dicho usuario.
Precondición	El producto en cuestión debe haber sido registrado previamente en la aplicación y estar asociado a uno de los negocios vinculados al usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario accederá al perfil vinculado a su negocio donde se encuentra el producto, accede al mismo e indica que quiere modificar su stock. El caso de uso comienza. 2. El sistema muestra al usuario formulario. 3. El usuario introduce en el formulario el nuevo stock. 4. El sistema verifica los datos introducidos por el usuario y modifica el stock asociado al producto indicado. El caso de uso finaliza.
Postcondición	El stock del producto en cuestión debe haber sido actualizado en la aplicación.
Excepciones	Paso 4: Si los datos introducidos son erróneos, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Alta
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-019	Notificaciones sobre el stock
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando sea necesario notificar al usuario sobre alteraciones dentro del stock vinculado a sus productos.
Precondición	Los productos sobre los que se quiere informar deben haber sido registrado previamente en la aplicación y estar asociado a uno de los negocios vinculados al usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El sistema identifica un evento relacionado con el stock de un producto, por el cual considera que el usuario debe ser notificado. El caso de uso comienza. 2. El usuario envía una notificación al usuario donde detalla el evento que se ha producido y le muestra el estado actual del stock en referente a los productos afectados. El caso de uso finaliza.
Postcondición	El usuario debe haber recibido la notificación correspondiente al stock.
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-020	Escribir reseña
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere escribir una reseña sobre un producto.
Precondición	El usuario debe estar registrado en la aplicación y el producto sobre el que se quiere escribir la reseña debe contener la visibilidad adecuada como para permitir el acceso de dicho usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario se dirige a la página del producto en cuestión e indica que quiere escribir una reseña. El caso de uso comienza. 2. El sistema verifica que el usuario es adecuado para escribir una reseña en el producto indicado. 3. El sistema muestra al usuario un formulario. 4. El usuario introduce en el formulario los datos necesarios. 5. El sistema verifica los datos introducidos por el usuario y vincula la reseña al producto. El caso de uso finaliza.
Postcondición	La reseña escrita por el usuario debe estar vinculada correctamente al producto en cuestión.
Excepciones	<ul style="list-style-type: none"> • Paso 2: Si el sistema determina que el usuario no es adecuado para escribir la reseña, mostrará un mensaje de notificación. El caso de uso finaliza. • Paso 5: Si los datos introducidos son erróneos, el sistema lo notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Inmediato. 4. Tiempo requerido por el usuario. 5. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-021	Eliminar reseña escrita
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere eliminar una reseña existente sobre un producto.
Precondición	El usuario debe estar registrado en la aplicación y el producto sobre el que se quiere escribir la reseña debe contener la visibilidad adecuada como para permitir el acceso de dicho usuario. La reseña debe haber sido creada por el mismo usuario que la quiere eliminar.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario se dirige a la página del producto en cuestión e indica que quiere eliminar una reseña en concreto. El caso de uso comienza. 2. El sistema verifica que el usuario es el mismo que escribió la reseña y la elimina de la página. El caso de uso finaliza.
Postcondición	La reseña escrita por el usuario debe estar vinculada correctamente al producto en cuestión.
Excepciones	<ul style="list-style-type: none"> • Paso 2: Si el sistema determina que el usuario no es adecuado para escribir la reseña, mostrará un mensaje de notificación. El caso de uso finaliza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Muy baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-022	Añadir a favoritos
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere añadir a favoritos un producto en concreto.
Precondición	El usuario debe estar registrado en la aplicación y el producto tener la visibilidad adecuada para mostrarse a dicho usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere añadir a favoritos un producto en concreto dentro del catálogo. El caso de uso comienza. 2. El sistema verifica que el usuario es adecuado y añade el producto a su lista de favoritos. El caso de uso finaliza.
Postcondición	El producto debe añadirse a la lista de favoritos del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 2: Si el sistema determina que el usuario no es adecuado para añadir el elemento a favoritos, mostrará un mensaje de notificación. El caso de uso finaliza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-023	Compartir producto
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere compartir el producto.
Precondición	El usuario debe estar registrado en la aplicación y el producto tener la visibilidad adecuada para mostrarse a dicho usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere compartir un producto en concreto dentro del catálogo. El caso de uso comienza. 2. El sistema muestra las diferentes opciones para compartir el producto 3. El usuario selecciona la opción deseada y realiza los pasos indicados para llevarlo a cabo. 4. El sistema envía la información del producto por el medio indicado. El caso de uso finaliza.
Postcondición	La información sobre el producto debe ser compartida por el sistema según el medio indicado por el usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 3: Si el usuario indica que quiere cancelar el proceso, el caso de uso finaliza
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-024	Añadir producto a la cesta
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-002: Gestión de productos
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere añadir un producto a la cesta.
Precondición	El usuario debe estar registrado en la aplicación y el producto tener la visibilidad adecuada para mostrarse a dicho usuario.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere añadir a la cesta un producto en concreto dentro del catálogo. El caso de uso comienza. 2. El sistema verifica que se puede realizar el proceso y agrega el producto a la cesta del usuario. El caso de uso finaliza.
Postcondición	El producto indicado debe haber sido agregado a la cesta del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 2: Si el sistema detecta que el artículo no puede ser agregado a la cesta, mostrará una notificación al usuario y el caso de uso finaliza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-025	Eliminar producto de cesta
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere eliminar un producto a la cesta.
Precondición	El usuario debe estar registrado en la aplicación y el producto a eliminar debe haber sido introducido previamente a la cesta.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere eliminar un producto concreto de su cesta en la sección de esta. El caso de uso comienza. 2. El sistema elimina el producto de la cesta del usuario. El caso de uso finaliza.
Postcondición	El producto indicado debe haber sido eliminado de la cesta del usuario.
Excepciones	Ninguna.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-025	Modificar cantidad de un producto
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar la cantidad de un producto a que se encuentra en su cesta.
Precondición	El usuario debe estar registrado en la aplicación y el producto a cuya cantidad se quiere modificar debe haber sido introducido previamente a la cesta.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar la cantidad a comprar asociada a un producto concreto de su cesta en la sección de esta. El caso de uso comienza. 2. El sistema verifica que se puede realizar el proceso y modifica la cantidad de compra asociada a dicho producto. El caso de uso finaliza.
Postcondición	El producto indicado debe haber sido eliminado de la cesta del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 2: En el caso de que el proceso no pueda llevarse a cabo, el sistema notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-027	Vaciar cesta
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere eliminar un producto a la cesta.
Precondición	El usuario debe estar registrado en la aplicación y la cesta no debe estar vacía.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere vaciar su cesta eliminando todos los productos que contiene la sección de esta. El caso de uso comienza. 2. El sistema elimina todos los productos de la cesta del usuario. El caso de uso finaliza.
Postcondición	Todos los productos deben haber sido eliminado de la cesta del usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 2: En el caso de que la cesta ya se encuentre vacía, el sistema no realizará ninguna modificación y el caso de uso finaliza.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-028	Editar dirección de envío
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que editar la dirección de envío dentro del proceso de compra.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en el proceso de compra.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar la dirección de envío asociada a la compra en proceso. El caso de uso comienza. 2. El sistema muestra al usuario el formulario donde modificar la dirección de envío. 3. El usuario modifica los datos necesarios. 4. El sistema verifica que los datos introducidos son correctos y modifica la dirección de envío asociada a la compra en proceso. El caso de uso finaliza.
Postcondición	La dirección de compra asociada a la compra debe haber sido actualizada.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos introducidos son erróneos, el sistema lo notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-029	Editar dirección de facturación
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que editar la dirección de facturación dentro del proceso de compra.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en el proceso de compra.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere modificar la dirección de facturación asociada a la compra en proceso. El caso de uso comienza. 2. El sistema muestra al usuario el formulario donde modificar la dirección de facturación. 3. El usuario modifica los datos necesarios. 4. El sistema verifica que los datos introducidos son correctos y modifica la dirección de facturación asociada a la compra en proceso. El caso de uso finaliza.
Postcondición	La dirección de facturación asociada a la compra debe haber sido actualizada.
Excepciones	<ul style="list-style-type: none"> • Paso 4: Si los datos introducidos son erróneos, el sistema lo notificará al usuario.
Rendimiento	<ol style="list-style-type: none"> 5. Inmediato. 6. Inmediato. 7. Tiempo requerido por el usuario. 8. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-030	Elegir método de pago
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que seleccionar el método de pago a utilizar dentro del proceso de compra.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en el proceso de compra.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere seleccionar el método de pago a utilizar en la compra en proceso. El caso de uso comienza. 2. El sistema muestra al usuario un panel con los diferentes métodos de pago asociados a su cuenta. 3. El usuario indica el método de pago que quiere utilizar. 4. El sistema actualiza el método de pago a utilizar en la compra en proceso. El caso de uso finaliza.
Postcondición	El método de pago asociado a la compra debe haber actualizado.
Excepciones	Ninguna.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-031	Elegir tipo de envío
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere seleccionar el tipo de envío a utilizar dentro del proceso de compra.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en el proceso de compra.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere seleccionar el tipo de envío a utilizar en la compra en proceso. El caso de uso comienza. 2. El sistema muestra al usuario un panel con los diferentes tipos de envío disponibles. 3. El usuario indica el tipo de envío que quiere utilizar. 4. El sistema actualiza el tipo de envío a utilizar en la compra en proceso. El caso de uso finaliza.
Postcondición	El tipo de envío asociado a la compra debe haber actualizado.
Excepciones	Ninguna.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-032	Realizar pago
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-003: Gestión de compraventa
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere realizar el pago de los artículos que tiene en la cesta
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en el proceso de compra, teniendo en la cesta al menos un artículo.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere realizar el proceso de compra mientras se encuentra en la sección de la cesta. El caso de uso comienza. 2. El sistema muestra al usuario un panel con el resumen de la compra y los datos asociados a la misma, como el método de pago y las direcciones de envío y facturación. 3. El usuario afirma que quiere realizar la compra. 4. El sistema redirecciona al usuario a la pasarela de pago. 5. El usuario realiza las acciones pertinentes en el sistema de pago para la realización de la compra. 6. El sistema verifica que la compra ha sido llevada a cabo con éxito y realiza las actualizaciones pertinentes. El caso de uso finaliza.
Postcondición	El sistema queda registrada la compra llevada a cabo por el usuario.
Excepciones	<ul style="list-style-type: none"> • Paso 3: Si el usuario niega que quiere realizar la compra, el caso de uso finaliza. • Paso 6: Si el sistema detecta que ha ocurrido un error durante el proceso de compra llevado a cabo en el sistema de pago, mostrará un mensaje al usuario y el caso de uso finalizará.
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato. 3. Tiempo requerido por el usuario. 4. Inmediato. 5. Tiempo requerido por el usuario. 6. Inmediato.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-033	Vista general del catálogo
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-004: Gestión del catálogo
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere visualizar una vista general del catálogo de la aplicación.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere dirigirse a la sección general del catálogo. El caso de uso comienza. 2. El sistema muestra al usuario una vista general del catálogo. El caso de uso finaliza.
Postcondición	El usuario debe situarse en la pantalla de catálogo.
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Alta
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-034	Vista concreta de producto
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-004: Gestión del catálogo
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere visualizar una vista de un producto concreto dentro del catálogo de la aplicación.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en la vista general del catálogo
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario indica que quiere dirigirse a una página del catálogo correspondiente a un producto concreto, dentro de la página general del propio catálogo. El caso de uso comienza. 2. El sistema muestra al usuario una vista del catálogo en referente a dicho producto en concreto. El caso de uso finaliza
Postcondición	El usuario debe situarse en la pantalla del catálogo referente a dicho producto en concreto.
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Alta
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-035	Aplicar búsqueda por texto
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-004: Gestión del catálogo
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere visualizar los productos relacionados a una búsqueda mediante un texto concreto proporcionado por el mismo.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en una sección referente al catálogo.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario introduce el texto por el que quiere realizar la búsqueda en el elemento dedicado a ello e indica que quiere visualizar los elementos relacionados con la misma. El caso de uso comienza. 2. El sistema muestra al usuario una vista general del catálogo con los elementos relacionados al texto que ha introducido previamente. El caso de uso finaliza
Postcondición	El usuario debe situarse en la pantalla de vista general del catálogo
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Alta
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-036	Aplicar filtros de búsqueda
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-004: Gestión del catálogo
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere visualizar los productos relacionados a el valor indicado por el mismo sobre los filtros proporcionados por el sistema.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en una sección referente al catálogo.
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario selecciona el valor deseado para aquellos filtros que quiere aplicar dentro de los proporcionados por el sistema e indica que quiere visualizar los elementos relacionados con la misma. El caso de uso comienza. 2. El sistema muestra al usuario una vista general del catálogo con los elementos relacionados a los filtros que ha especificado anteriormente. El caso de uso finaliza.
Postcondición	El usuario debe situarse en la pantalla de vista general del catálogo
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 1. Inmediato. 2. Inmediato.
Frecuencia	Alta
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

UC-037	Editar estructura del catálogo
Versión	Sin numerar
Autores	Equipo de desarrollo
Fuentes	Usuario
Dependencias	OBJ-004: Gestión del catálogo
Descripción	El sistema deberá comportarse como se describe en el presente caso de uso cuando un usuario indique que quiere modificar la estructuración de la vista general del catálogo en base a una de las opciones proporcionadas por el sistema.
Precondición	El usuario debe estar registrado en la aplicación y encontrarse en la vista general del catálogo
Secuencia normal	<ol style="list-style-type: none"> 1. El usuario selecciona una opción concreta de las proporcionados por el sistema relativas a la estructura con la que se muestra la vista general del catálogo. El caso de uso comienza. 2. El sistema muestra al usuario una vista general del catálogo con los elementos relacionados a los filtros que ha especificado anteriormente. El caso de uso finaliza.
Postcondición	El usuario debe situarse en la pantalla de vista general del catálogo
Excepciones	Ninguna
Rendimiento	<ol style="list-style-type: none"> 3. Inmediato. 4. Inmediato.
Frecuencia	Alta
Importancia	Media
Urgencia	Media
Estado	En desarrollo
Estabilidad	Alta
Comentarios	Ninguno

3.4 Definición de requisitos No Funcionales

NFR-001	Escalabilidad
Versión	Sin numerar.
Autores	Equipo de desarrollo.
Fuentes	Cliente que encarga la aplicación.
Dependencias	OBJ-001: Gestión de usuarios y perfiles. OBJ-002: Gestión de productos. OBJ-003: Gestión de compraventa.
Descripción	El sistema debe permitir la inclusión de nuevos usuarios en el mismo sin que para ello se sufran problemas de rendimiento ni fiabilidad de los datos. En adición, la inclusión de nuevos usuarios no debe necesitar modificar la información referente a los que ya se encuentran registrados en el sistema.
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo
Estabilidad	alta
Comentarios	Ninguno

NFR-002	Información gráfica y visual
Versión	Sin numerar.
Autores	Equipo de desarrollo.
Fuentes	Cliente que encarga la aplicación.
Dependencias	OBJ-003: Gestión de compraventa. OBJ-004: Gestión del catálogo.
Descripción	El sistema debe mostrar la información de forma gráfica y fácil de interpretar a los usuarios, con el objetivo de que su uso sea accesible e inclusivo para todos aquellos sectores de la población que sufren dificultades en el uso de aplicaciones semejantes.
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo
Estabilidad	alta
Comentarios	Ninguno

NFR-003	Seguridad
Versión	Sin numerar.
Autores	Equipo de desarrollo.
Fuentes	Cliente que encarga la aplicación.

Dependencias	OBJ-001: Gestión de usuarios y perfiles. OBJ-002: Gestión de productos. OBJ-004: Gestión del catálogo.
Descripción	<p>El sistema debe proporcionar una capa de seguridad sobre el uso de los datos referentes a los usuarios que utilizan la aplicación. Esto incluye tanto a la información personal como a la de los negocios y productos vinculados a los mismos.</p> <p>El sistema debe garantizar las medidas necesarias para evitar el acceso no autorizado de terceros a los datos de usuarios o la realización de suplantaciones de identidad.</p> <p>El sistema debe garantizar el uso de conexiones cifradas con las que evitar el uso de técnicas de adquisición de información durante el flujo de transacción de datos entre el usuario y los servidores del sistema.</p>
Importancia	vital
Urgencia	inmediatamente
Estado	En desarrollo
Estabilidad	alta
Comentarios	Ninguno

4. Desarrollo del proyecto

En el presente apartado se detallarán todos los pasos llevados a cabo por el equipo para desarrollar un prototipo del producto especificado en la propuesta detallada anteriormente. Estos pasos se centran en transmitir al lector una visión aproximada de como se ha realizado el desarrollo, sin embargo, estos no tienen por qué expresarse en el mismo orden en el que se han llevado a cabo.

Se ha decidido desarrollar los pasos de esta manera para facilitar el entendimiento del desarrollo. En el caso de que el lector quiera obtener una visión más aproximada a como se ha llevado a cabo el orden de los acontecimientos, puede obtener más información en el apartado denominado *Tabla de fechas*.

- Paso 1:** Definición de herramientas y estructura.
- Paso 2:** Diseño de la base de datos.
- Paso 3:** Estructura del proyecto Backend.
- Paso 4:** Desarrollo del Backend.
- Paso 5:** Estructura del proyecto Frontend.
- Paso 6:** Creación del Frontend y navegación entre pantallas.
- Paso 7:** Maquetación de escenas del Frontend para sistemas móviles.

4.1 Paso 1: Definición de herramientas y estructura

El proyecto se basa en una aplicación web orientada a ser utilizada a través de un conjunto heterogéneo de dispositivos, por lo que resulta necesario emplear en un *framework* que nos permita dotar al *Frontend* de la suficiente versatilidad como para interaccionar con las diferentes interfaces y tamaños de pantalla.



Como hemos visto en la propuesta, para hacer esto utilizaremos la herramienta *Ionic* apoyada en el *framework* de desarrollo *Angular*.

Además de esto, necesitaremos construir un *backend* con el que podamos comunicarnos para rescatar los datos que consideremos necesarios, independientemente del dispositivo en el que nos encontremos. Dicho de otra manera, debemos abstraer la información de los usuarios del dispositivo en el que se encuentran.



Para esto se ha decidido utilizar la herramienta *Nose.js*, debido a que se trata de la más utilizada y la que dispone actualmente de una mayor variedad de recursos. Dentro de esto se ha implementado el *Framework* para servidores web *express*, el cual nos facilita la creación de los distintos mensajes y respuestas que utilizará el cliente para interaccionar con el *Backend*.

Por otra parte, con el objetivo de poder generar imágenes independientes del *Backend* se ha optado por utilizar la herramienta *Docker*, la cual nos permite poder interaccionar con el mismo sin la necesidad de estar ejecutándolo localmente.

Finalmente, para poder desarrollar un buen acceso y almacenamiento de la información necesitaremos diseñar e implementar una base de datos donde ponerlos a buen recaudo. Se ha decidido llevar a cabo el diseño y creación de esta mediante la herramienta de bases de datos no relacionales *MongoDB*.



Se considera necesario separar físicamente el *backend* de la base de datos, para lo que emplearemos el servicio *MongoDB Atlas*, el cual nos permite implementar nuestra base de datos en un servidor lejano y gestionar los datos almacenados en el mismo. Para facilitar el acceso a la base de datos se está utilizando la herramienta *MongoDB Compass*, la cual hace la labor de interfaz gráfica entre los servidores donde se instancia la base de datos y el usuario.

Por otra parte, se ha considerado implementar la interacción de la aplicación con diferentes APIs para proporcionar un extra de funcionalidades que resulten útiles de cara al usuario. Se han elegido las siguientes APIs:

- **FatSecret:** Nos permite obtener información nutricional a cerca de los alimentos subidos a la plataforma. Utilizada para mostrar a los clientes datos importantes con los que pueden tener un mayor conocimiento sobre aquellos productos que pueden comprar.
- **MapBox:** API con la que podemos mostrar mapas mediante una posición geográfica dada. Utilizada para mostrar a los clientes la dirección de los comercios locales de una manera más visual.



De esta manera podemos definir tres partes principales que componen el proyecto, las cuales se comunican entre sí para llevar a cabo el correcto tratamiento de la información por parte del usuario. Estas son:

- **Fontend:** Desarrollado mediante *Ionic – Angular*, implementa diferentes vistas con el objetivo de dar soporte tanto a la versión de aplicaciones móviles como de escritorio. Se comunica con el *Backend* mediante un servicio específico.
- **Backend:** Desarrollado mediante la herramienta *Node.js* e implementado a través del *framework express.js*, realiza el tratamiento necesario de los datos para llevar a cabo las acciones solicitadas por el usuario. También es el responsable de comunicarse con las APIs utilizadas a través de un servicio específico para cada una de ellas.
- **Data Base:** Responsable del correcto almacenamiento de los datos utilizados por los clientes. Esta desarrollada mediante *MongoDB* y se utiliza el servicio *MongoDB Atlas* para su correcta implementación en un servidor lejano.

El desarrollo de la aplicación se describe desde abajo hacia arriba, es decir, desde la base de datos hasta la interacción del usuario mediante el *Frontend*. Esto con el objetivo de facilitar el entendimiento del proceso, aunque no tiene por que corresponderse con el desarrollo real.

En la siguiente imagen podemos visualizar un esquema que muestra la estructura y las diferentes herramientas utilizadas para la implementación de la aplicación.

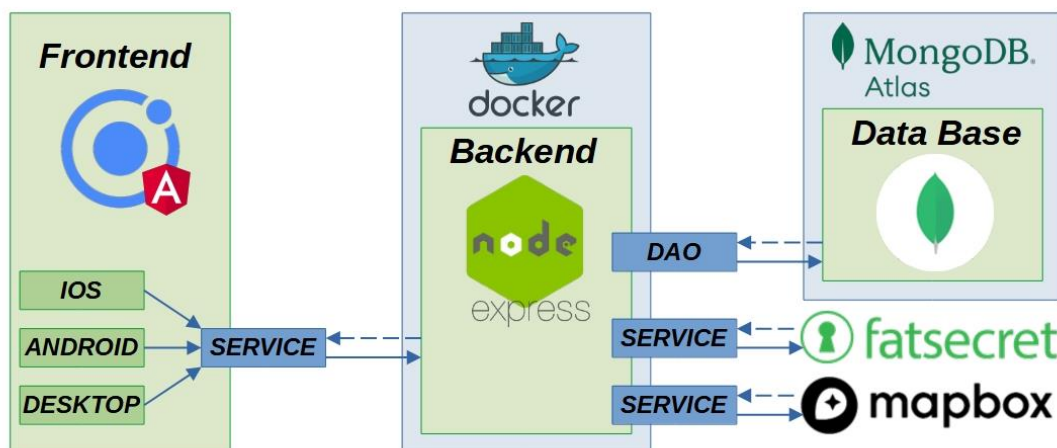


Ilustración 11: Esquema general de la estructura del proyecto

4.2 Paso 2: Diseño de la base de datos

Uno de los primeros pasos llevados a cabo es el diseño de la base de datos, en la cual se almacenará la información sobre la que los usuarios interactuarán. Esto es importante por qué influirá tanto en el tratamiento de los datos llevados a cabo por el *Backend* como la representación de estos en el *Frontend*.

En la siguiente imagen podemos ver el modelo de diseño de la base de datos implementada:

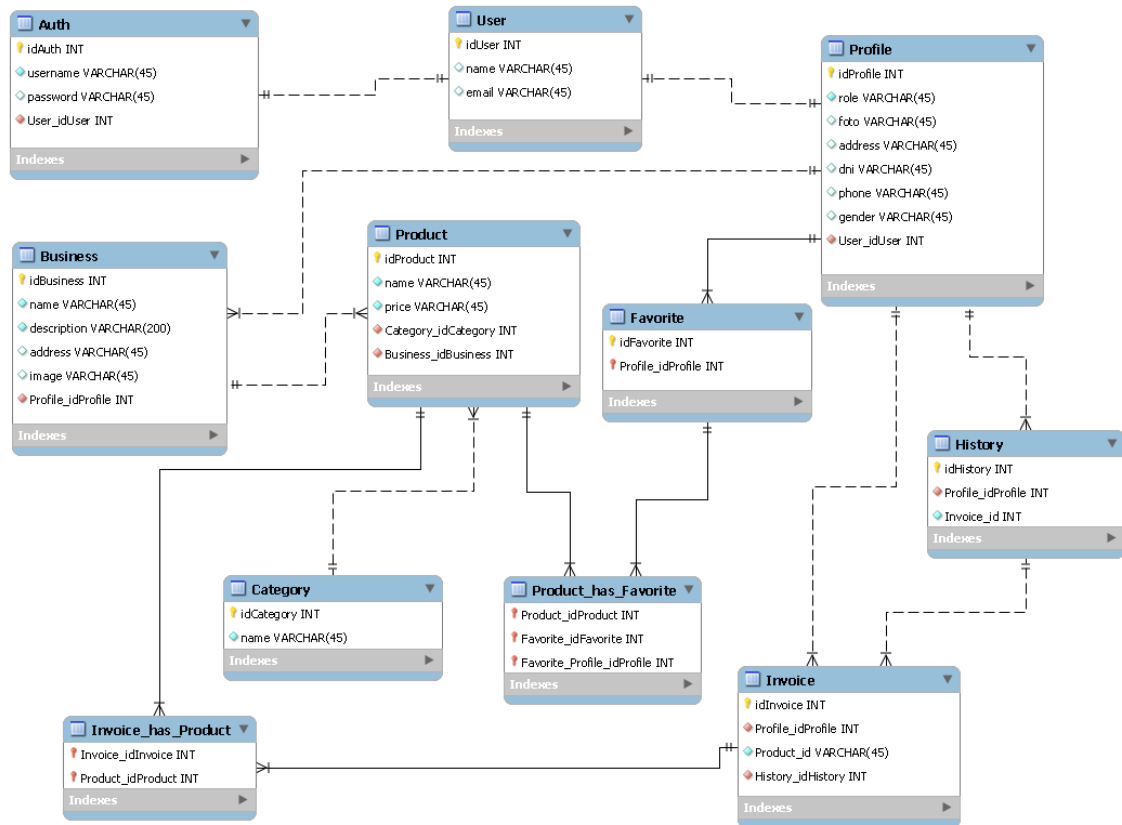


Ilustración 12: Diagrama de diseño de la base de datos

A continuación, indicamos información relevante a cerca de cada una de las tablas que componen la base de datos:

- **Auth:** Hace referencia a la tabla autenticación (Authentication), en ella se almacenarán únicamente los datos necesarios para hacer el *Loguin*, vale destacar que el token no se almacena en la base de datos, pues se genera un nuevo token cada vez que el usuario intenta acceder a la aplicación. La tabla *Auth*, tendrá una relación de uno a uno con la tabla *User* lo que significa que un usuario solo puede registrarse una vez, es decir, no se permite que dos o más usuarios tengan los mismos datos de acceso (usuario o email).
- **User:** Almacena los datos básicos del usuario que se obtienen cuando este se da de alta en la aplicación (nombre, nombre de usuario, email y contraseña). El nombre de usuario y la contraseña se utilizarán para crear las credenciales de acceso en la tabla *Auth*, ambos junto al email deben ser únicos.
- **Profile:** En esta tabla se almacenan entre los datos personales del usuario, especificando el rol para restringir el acceso del usuario a ciertas funcionalidades, por ejemplo: Un usuario que tenga el rol de comprador no se le permite crear un negocio. Una vez dado

de alta a un usuario, este podrá configurar su perfil cuando lo desea, cada usuario tendrá asociado un único perfil.

- **Business:** Almacena los datos del negocio asociado al perfil de un determinado usuario. Un negocio solo puede estar asociado al perfil de un único usuario, sin embargo, un perfil puede estar asociado a varios negocios.
- **Product:** Almacena los datos asociados a cada uno de los productos registrados en la aplicación. Establece relación de uno a muchos con la tabla *category* y una relación de uno a muchos con la tabla *business* ya que un mismo producto solo puede pertenecer a un único negocio.
- **Category:** Identifica las diferentes categorías en las que se pueden clasificar los productos.
- **Invoice:** Almacena las facturas de las compras que realizan los usuarios. Tiene una relación de muchos a muchos con la tabla *Product*.
- **Favorite:** Se almacenan los productos favoritos de los usuarios, un usuario puede tener una o varias listas de sus productos favoritos.
- **History:** Se almacenan historiales de compras llevados a cabo por cada uno de los usuarios.

4.3 Paso 3: Estructura del proyecto Backend

En este apartado se describirá de forma detallada la estructura del proyecto *Backend*, indicando cada uno de los directorios y los ficheros que componen el api:

- **app.js:** Es el fichero que se utiliza para arrancar la aplicación y donde se importan todas las configuraciones necesarias definidas en otros directorios de la aplicación. También se definen las rutas para el acceso para llevar a cabo las peticiones a cada una de las entidades mediante la instrucción *app.use()*, la cual recibe como parámetro la ruta donde se encuentra implementada las peticiones de la entidad a la que se quiere acceder.
- **src/:** Es el directorio principal donde realiza el desarrollo de la aplicación y el cual contiene todos los elementos que definen el funcionamiento de esta. Este contiene los siguientes directorios:
 - **apiServices/:** Contiene los subdirectorios que representan cada una de las entidades que estructuran los datos de la aplicación. Estos subdirectorios tienen el mismo nombre que la entidad que representa, de tal forma que para la implementación de cada entidad son necesarios los siguientes ficheros: *controller.js*, *dao.js*, *router.js* *model.js* y *dto.js*.
 - **bin/:** Implementa la configuración para la inicialización de la aplicación.
 - **config/:** Contiene el fichero *config.js* donde se encuentran definidas e inicializadas las variables con valores por defecto para el arranque de la aplicación, conexión a la base de datos y la palabra secreta para la generación del token. Los valores del *puerto*, *host* y *db_url* definidos en estos ficheros solo se utilizan si no se ha definido los en el fichero de configuración del entorno (los cuales tienen la extensión. *env*).
 - **constants/:** Contiene la definición de las diferentes constantes utilizadas en la aplicación (como parte de las buenas prácticas de la programación), de modo que cuando queremos mostrar un determinado mensaje, imprimimos la constante asociada al mismo. *Por ejemplo*, en el fichero *constants.js* se encuentra definida la constante *PRODUCT_CREATED= "The producto has been created successful"*. De esta forma, cuando se registre un nuevo producto de forma exitosa, se imprime el contenido de dicha variable
 - **middleware/:** Implementa las funcionalidades para gestionar el control de acceso de los usuarios a los distintos servicios que ofrece la *API*. Este control se llevará a cabo protegiendo el acceso a las diferentes rutas frente a las diferentes peticiones que puedan llevarse a cabo, esto mediante la verificación del rol que tiene el usuario cuando lleva a cabo dicho acceso.
 - **routes/:** Contiene la implementación de las rutas para gestionar las peticiones que se pueden realizar en la *API* y permite obtener información necesaria de las distintas entidades. Para cada entidad se ha creado un fichero correspondiente con el mismo nombre donde se implementan las rutas para gestionar las peticiones realizadas hacia la misma.
 - **services/:** Implementa los servicios necesarios para gestionar la conexión y/o comunicación entre el backend y otras aplicaciones externas, *por ejemplo*, la

conexión con la base de datos, las peticiones hacia las *APIs* externas empleadas o las diferentes comunicaciones con el servicio de email.

- **uploads/:** Almacena los contenidos estáticos que se van subiendo a la aplicación, tales como imágenes y/o documentos, cada uno de los cuales se almacenan en un subdirectorío con el mismo nombre de la entidad con la cual están relacionados. *Por ejemplo*, las imágenes de los productos se almacenan en el directorio *uploads/products*.
- **Dockerfile:** Configuración para generar la imagen de Docker.
- **Package.json:** Se crea por defecto al crear el proyecto. Contiene un listado con cada uno de los nombres de los módulos que se utilizarán durante el desarrollo de la aplicación. También se utiliza para definir algunos comandos importantes para la inicialización de la aplicación.
- **.env:** Contiene la definición e inicialización de las variables de entorno.
- **.gitignore:** Listado de aquellos ficheros y directorios que no serán subidos al repositorio remoto de *GitHub*.

4.4 Paso 4: Funcionamiento general del Backend

El desarrollo del backend se centra en la implementación de los elementos que conforman las diferentes entidades definidas en la aplicación. Cada entidad está compuesta por los siguientes elementos:

- **routes.js:** Implementan todos los métodos para atender las diferentes peticiones del usuario respecto a la entidad en cuestión. Estos métodos se clasifican según la función que desarrollan, algunos de los cuales son: *get*, *post*, *delete*, o *put*.
 - Cuando se realiza una petición, el componente *router* identifica de que tipo se trata y su existencia, posteriormente se comunica con el *controller* enviándole los parámetros recibidos en caso de ser necesario. Si la petición realizada por el usuario no se corresponde con ningún método implementado, la aplicación devolverá el error 404.
- **controller.js:** Responde a las peticiones del *routes.js* en base al método especificado. Su función principal es establecer la comunicación entre el *routes* y el *dao*.
- **dao.js (data access object):** Define el acceso a los datos correspondientes a la entidad se encarga de enviar las consultas a la base de datos y devolver al *controller* lo que haya obtenido de la consulta.
- **dto.js (data transfer object):** Representa el objeto de transferencia de los datos y define la información queremos devolver al usuario tras realizar una determinada petición.
 - En casos específicos, el *dao* devuelve un objeto con toda la información almacenada referente a la consulta y posteriormente el *dto* define aquellos datos que nos interesa mostrarle al usuario.
- **model.js:** Define la estructura de los datos asociados a la entidad.

Para un mejor entendimiento, analicemos el funcionamiento de las clases anteriores en base a al siguiente ejemplo: Imaginemos realizar una petición a través de la siguiente ruta: *http://localhost:3000/api/v1/products*, de la cual se espera obtener una lista con todos los productos almacenados en la base datos de la aplicación.

Las peticiones realizadas por los clientes son identificadas en base al tipo y a la dirección a la cual son enviadas. Dichas rutas son controladas en las diferentes clases routes.js, sin embargo, para poder llegar hasta ellas el fichero raíz del proyecto app.js se encarga de implementar las redirecciones hacia los diferentes ficheros routes.js en base a las entidades a las cuales afecta.

De este modo, la petición es recibida por el fichero app.js y una identifica se envía a través de la función app.use(...) a su correspondiente router. Así mismo, este identifica el tipo de petición y accede al método correspondiente que hace la solicitud al controller para a través del dao.js devolver la respuesta esperada.

La implementación para este ejemplo en concreto sería:

```

app.use('/api/v1', userRoutes);
app.use('/api/v1', categoryRoutes);
app.use('/api/v1', businessRoutes);
app.use('/api/v1', productRoutes);

```

Ilustración 13: clase app.js. fuente: elaboración propia

```

router.getAsync('/', (req, res)=> {
  controller.getProduct()
    .then(products =>{
      res.status(200).json(products);
    }).catch(err=>{
      res.status(500).send('Internal Error')
    })
});

```

Ilustración 14: Método get de la clase routes.js de la entidad products. Fuente: Elaboración propia

```

async getProduct(){
  const productDao = await dao.getProduct();
  const productDto = dto.multiple(productDao);

  return productDto;
},

```

Ilustración 15: Método getProduct de la clase controller.js de la entidad products. Fuente: Elaboración propia

```

async getProduct() {
  return new Promise((resolve, reject) => {
    Model.find({}).sort({date: 1})
      .populate('businessId')
      .populate('categoryId')
      .exec((error, product) => {
        if(error) {
          reject(error);
          return false;
        }
        resolve(product);
      });
  });
},

```

Ilustración 16: Método getProduct de la clase dao.js de la entidad products.js. Fuente: Elaboración propia

Una vez hemos obtenido todos los relacionados con la consulta, definiremos en el dto cuales son aquellos que queremos mostrar al usuario.

```

const single = (resource) => ({
  id: resource._id,
  Name: resource.name,
  Price: resource.price,
  Coin: resource.coin,
  Unit: resource.unit,
  CategoryId: resource.categoryId._id ,
  Category: resource.categoryId.name ,
  StoreId: resource.businessId._id,
  Store: resource.businessId.name
});

const multiple = (resources) => resources.map((resource) => single(resource));

module.exports = {
  single,
  multiple
}

```

Ilustración 17: clase dto.js de la entidad products. Fuente: Elaboración propia

4.5 Paso 5: Obtención de la información nutricional

Para proporcionar información nutricional de un producto, hemos utilizado el servicio *Nutrition API* de *API Ninjas*. Esta extrae información nutricional del texto (en nuestro caso, el nombre del producto) mediante procesamiento de lenguaje natural. Desde blogs de comida hasta menús y recetas, puede leer cualquier texto y calcular los datos nutricionales correspondientes.

Una característica inteligente de esta API es el cálculo personalizado de porciones: Si el texto especifica cantidades de alimentos o ingredientes individuales, el algoritmo escalará automáticamente los datos nutricionales en el resultado en consecuencia.

Para utilizar la API en referencia, es necesario la Api-Key de esta, la cual se obtiene al registrarse en la aplicación. En la siguiente imagen se puede ver la codificación para la conexión de nuestro servidor backend con la API:

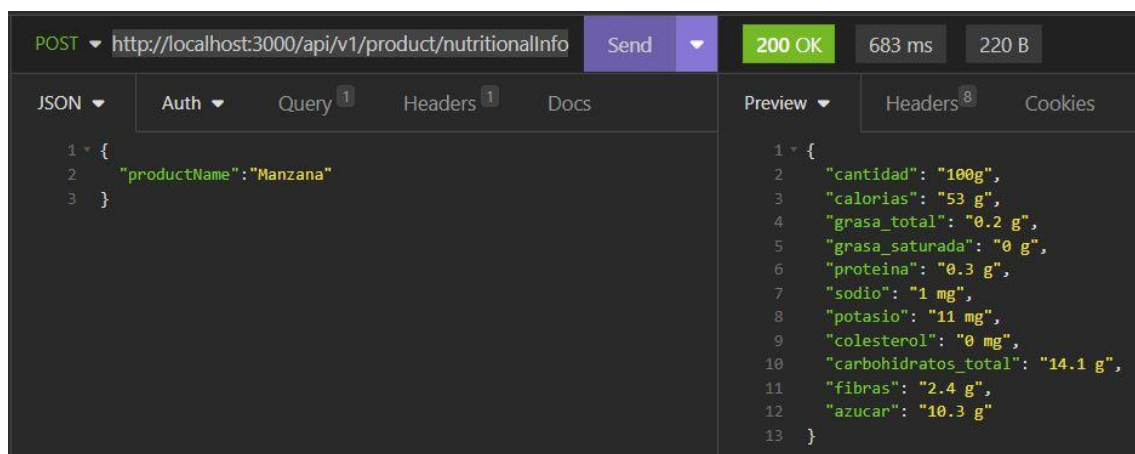
```
async getNutritionalInfoProduct(productName) {
  const query = translator[productName.toLowerCase()];
  return new Promise((resolve, reject) => {
    request.get({
      url: 'https://api.api-ninjas.com/v1/nutrition?query=' + query,
      headers: {
        'X-API-Key': 'fP0afWLmdTc9sR52l/Bucg==IdEvQUhE7zBJ9Zbc'
      },
    },
    function(error, response, body) {
      if(error) return console.error('Request failed:', error);
      else if(response.statusCode !== 200) return console.error('Error:', response.statusCode, body.toString('utf8'));
      resolve(JSON.parse(body));
    });
  });
}
```

Ilustración 18: Conexión con la API para obtener la información nutricional de un producto. Fuente: Elaboración propia

Dado que no todos los nombres de alimentos en español son reconocidos por la API, hemos implementado un método traductor, que dado el nombre de un producto en español lo traduce en inglés y, la llamada al método GET de la API se hace mediante la traducción en inglés.

El método GET devuelve información nutricional correspondiente a cada 100 gramos del producto, tales como la cantidad de calorías, grasas, colesterol, carbohidratos, proteínas, vitaminas y minerales en la porción indicada del alimento.

Para obtener la información nutricional de un determinado producto desde nuestro servidor, enviamos una petición GET a la ruta <http://localhost:3000/api/v1/product/nutritionalInfo> y en el cuerpo de la petición, colocamos el nombre del producto.



The screenshot shows a REST client interface with a POST method selected for the URL `http://localhost:3000/api/v1/product/nutritionalInfo`. The request body is a JSON object: `{ "productName": "Manzana" }`. The response status is `200 OK` with a response time of `683 ms` and a body size of `220 B`. The response body is a JSON object containing nutritional information for an apple (Manzana).

Field	Value
cantidad	"100g"
calorias	"53 g"
grasa_total	"0.2 g"
grasa_saturada	"0 g"
proteina	"0.3 g"
sodio	"1 mg"
potasio	"11 mg"
colesterol	"0 mg"
carbohidratos_total	"14.1 g"
fibras	"2.4 g"
azucar	"10.3 g"

Ilustración 19: Información nutricional del producto. Fuente: Elaboración propia.

4.6 Paso 6: Estructura del proyecto Frontend

En este apartado se describirá de forma detallada la estructura del proyecto *Frontend*, indicando cada uno de los directorios y los ficheros relevantes que componen la aplicación:

- **src/:** Es el directorio principal donde realiza el desarrollo de la aplicación y el cual contiene todos los elementos que definen el funcionamiento de esta. Este contiene los siguientes directorios:
 - **app/:** Contiene todos aquellos elementos que definen el funcionamiento de la aplicación, los cuales se encuentran organizados en subdirectorios dependiendo de la función que desarrollan dentro del proyecto. Los principales directorios y ficheros que contiene son:
 - **components/:** Contiene todos y cada uno de los componentes de carácter unitario desarrollados en la aplicación, los cuales serán utilizados a lo largo de todo el proyecto. Los componentes están compuestos por un total de cuatro archivos:
 - Un archivo de definición de elementos *.html*.
 - Un archivo de definición de estilos *.scss*.
 - Un archivo de programación web *.ts*.
 - Un archivo de pruebas unitarias *.spec.ts*.
 - **modules/:** Contiene cada uno de los módulos que componen la aplicación, los cuales son los encargados de establecer la navegación entre las diferentes secciones de la aplicación. Los módulos están compuestos por diferentes componentes, cada uno de los cuales representa una página diferente dentro de la sección que controla el módulo en cuestión.
 - **assets/:** Almacena los elementos multimedia estáticos utilizados para el desarrollo de la aplicación, tales como imágenes, vídeos, audios, etc.
 - **environments/:** Contiene las variables de entorno que definen el funcionamiento de la aplicación. Estas variables suelen tener diferentes valores dependiendo del contexto en el que se ejecute la aplicación, por lo que se definen tantos ficheros como cada contexto de ejecución queramos utilizar.
 - **styles/:** Contiene la definición de los estilos personalizados para los diferentes elementos de la interfaz empleados en el proyecto, tales como botones, textos, etc. Estos estilos se almacenan en diferentes ficheros dependiendo del elemento de la interfaz para el cual estén diseñados.
 - **theme/:** Contiene la definición de variables *css* utilizadas para aplicar los estilos estándar a la aplicación. Estas variables están implementadas inicialmente por *ionic*, pero pueden ser modificadas a lo largo del desarrollo del proyecto.
- **.gitignore:** Listado de aquellos ficheros y directorios que no serán subidos al repositorio remoto de *GitHub*.
- **ionic.config.json:** Se crea por defecto al generar el proyecto. Contiene la configuración estándar el proyecto de *ionic*.

- **package.json:** Se crea por defecto al generar el proyecto. Contiene un listado con cada uno de los nombres de los módulos que se utilizarán durante el desarrollo de la aplicación. También se utiliza para definir algunos comandos importantes para la inicialización de la aplicación.
- **tsconfig.json:** Se crea por defecto al generar el proyecto. Contiene los parámetros de configuración del lenguaje *typescript*, utilizado en los ficheros de programación web del proyecto.

4.7 Paso 7: Creación del Frontend y navegación entre pantallas

Una vez se ha elaborado la estructura del proyecto vista en el paso anterior, se procede a crear las diferentes pantallas a utilizar e implementar la navegación entre las mismas. Dentro de la navegación se diferenciarán dos secciones claramente contrapuestas:

- **Sección exterior:** Dentro de esta sección el usuario aún no se ha autenticado en la aplicación, por lo no podrá acceder visualizar o interaccionar con el contenido de esta. Esta sección está compuesta por las siguientes pantallas:
 - **Acceso:** Esta pantalla permite al usuario introducir sus credenciales, con las que podrá acceder al interior de la aplicación e interaccionar con la misma. Para hacer esto las credenciales deben haber sido registradas anteriormente en la aplicación.
 - Esta pantalla se corresponde al componente *login-page*, dentro del módulo *Access-section*.
 - **Creación de nueva cuenta:** Esta pantalla permite crear una nueva cuenta en el sistema en base a registrar unas nuevas credenciales de acceso.
 - Esta pantalla se corresponde al componente *create-account-page*, dentro del módulo *Access-section*.
 - **Nueva cuenta creada con éxito:** Indica al usuario que el proceso llevado a cabo para la creación de una nueva cuenta ha concluido con éxito.
 - Esta pantalla se corresponde al componente *create-account-success-page*, dentro del módulo *Access-section*.
- **Sección interior:** Dentro de esta sección el usuario puede visualizar e interaccionar con el contenido de la aplicación, realizando todas aquellas acciones que requieren de un registro previo para llevarse a cabo.
 - **catálogo:** Es la página principal de la aplicación y nos permite tanto visualizar los diferentes productos registrados que componen el catálogo como incluir los mismos en nuestra cesta. Esta página debe poder ser accesible desde cualquiera de las que componen la sección interior de la aplicación.
 - Esta pantalla se corresponde al componente *catalog-page*, dentro del módulo *catalog-section*.
 - **producto:** Contiene toda la información referente a un producto concreto registrado en la aplicación, incluyendo los valores nutricionales obtenidos a través de la *API FarSecret*. Esta ventana también permite al usuario incluir el producto en la cesta.
 - Esta pantalla se corresponde al componente *product-page*, dentro del módulo *catalog-section*.
 - **perfil:** Contiene toda la información referente a la cuenta que ha realizado a la autenticación, incluyendo elementos como su perfil, credenciales de acceso, negocios asociados a la cuenta, etc.
 - Esta pantalla se corresponde al componente *profile-page*, dentro del módulo *profile-section*.

- **negocio:** Contiene toda la información relacionada con el negocio en cuestión y con los productos registrados en el mismo.
 - Esta pantalla se corresponde al componente *business-page*, dentro del módulo *business-section*.
- **Cesta:** Muestra todos y cada uno de los productos seleccionados para la futura compra del cliente. Debe mostrar un resumen de los datos más importantes de la misma y dar la opción de modificar la cantidad de los productos seleccionados previamente.
 - Esta pantalla se corresponde al componente *shipping-cart-page*, dentro del módulo *shopping-cart-section*.

Una vez descritas cada una de las escenas que componen el proyecto, en la siguiente imagen podemos ver el diagrama de navegación que conecta todas y cada una de las mismas.

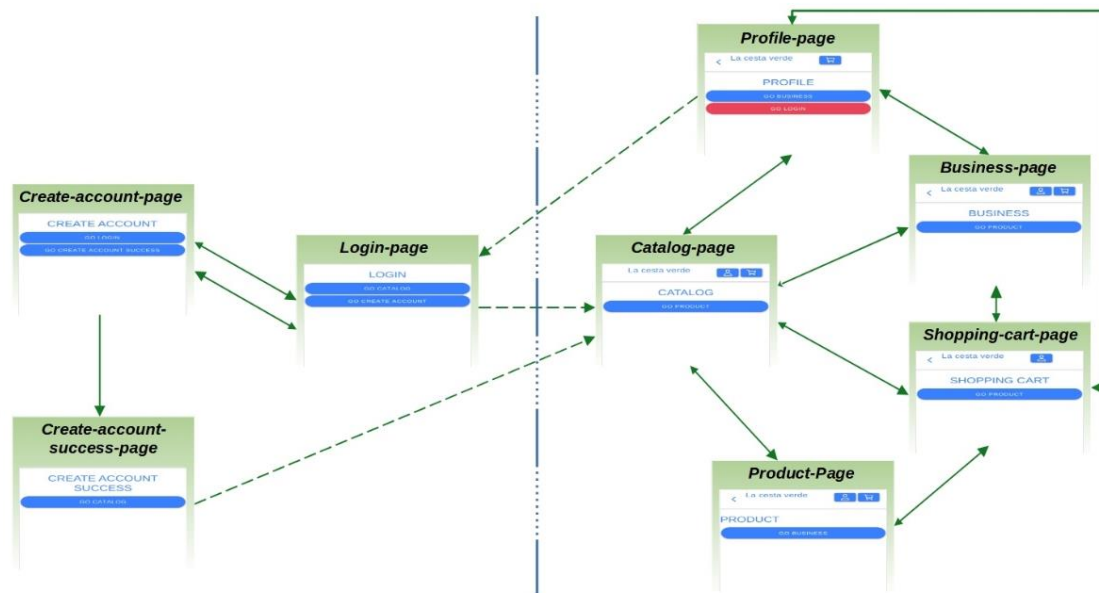


Ilustración 20: Esquema de navegación entre las pantallas de la aplicación

Para poder implementar correctamente la navegación básica entre las diferentes pantallas de la aplicación es necesario crear el primer componente de carácter unitario. Este tendrá por nombre *main-header* y será accesible en forma de cabecera dentro de todas las pantallas que conforman la sección interior de la aplicación.

Este componente participará de un mayor número de acciones que no están directamente relacionadas con la navegación de cara a un futuro, pero en la primera versión del mismo únicamente abarcará este aspecto.

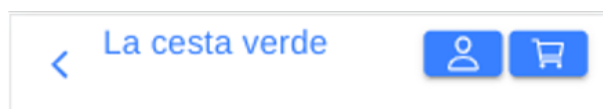


Ilustración 21: Apariencia del componente main-header

En la siguiente imagen podemos ver el fichero *html* que define todos los componentes que conforman el componente de cabecera:

```

1 <div class="divStyle_headFrame">
2   <ion-toolbar>
3     <ion-row>
4       <ion-col size="1">
5         <ion-back-button icon="chevron-back-outline" color="primary"></ion-back-button>
6       </ion-col>
7       <ion-col size="7">
8         <ion-title class="ion-text-left" color="primary" (click)="buttonClic_goCatalog()">La cesta verde</ion-title>
9       </ion-col>
10      <ion-col size="4">
11        <ion-button *ngIf="actualPage!='profile'" (click)="buttonClic_goProfile()" size="small">
12          <ion-icon slot="icon-only" name="person-outline"></ion-icon>
13        </ion-button>
14        <ion-button *ngIf="actualPage!='shoppingCart'" (click)="buttonClic_goShoppingCart()" size="small">
15          <ion-icon slot="icon-only" name="cart-outline"></ion-icon>
16        </ion-button>
17      </ion-col>
18    </ion-row>
19  </ion-toolbar>
20 </div>

```

Ilustración 22: Fichero HTML del componente main-header

En la siguiente imagen podemos ver un ejemplo de como se invoca el componente de cabecera dentro mediante el framework de *Angular*. Esto nos puede servir como ejemplo para entender como se lleva a cabo la invocación del resto de componentes que conforman la aplicación.

```

1 <ion-header>
2   <app-main-header actualPage="catalog"></app-main-header>
3 </ion-header>

```

Ilustración 23: Invocación del componentes main-header

4.8 Paso 8: Maquetación de escenas del Frontend para sistemas móviles

En las siguientes imágenes podemos ver la maquetación de cada una de las pantallas en base a los diversos componentes que nos proporciona *ionic* y su correspondencia con la primera versión implementada. Aquellos elementos que se encuentran dentro de un mallado gris, se corresponden con componentes encapsulados de desarrollo propio.

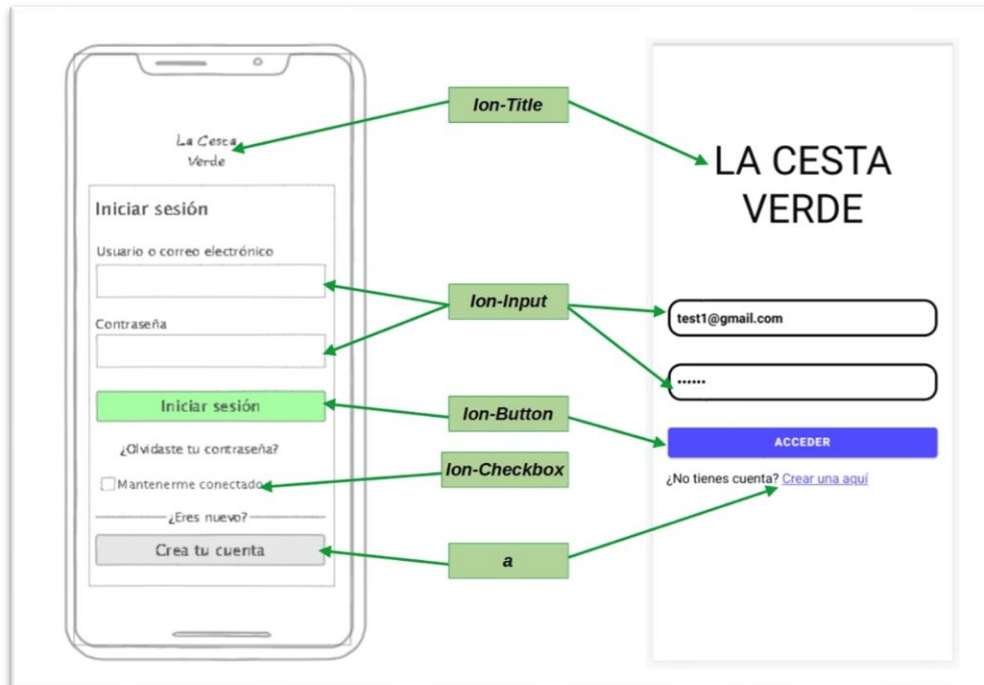


Ilustración 24: Correspondencia de elementos entre el mockUp y la aplicación móvil en el acceso

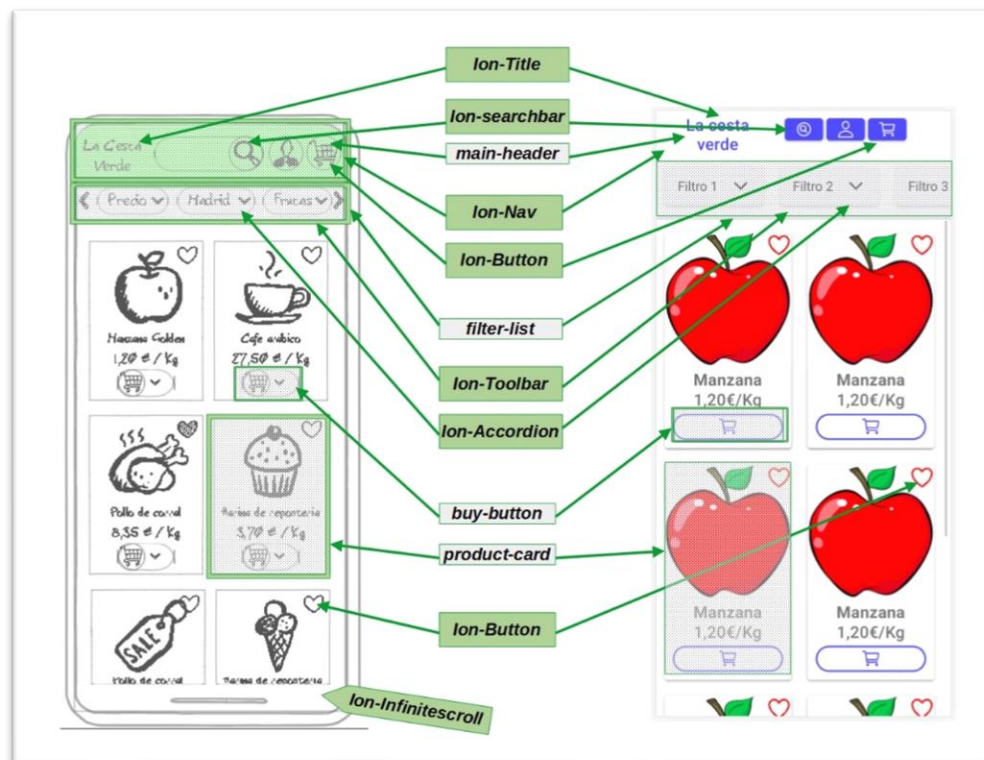


Ilustración 25: Correspondencia de elementos entre el mockUp y la aplicación móvil en el catálogo general

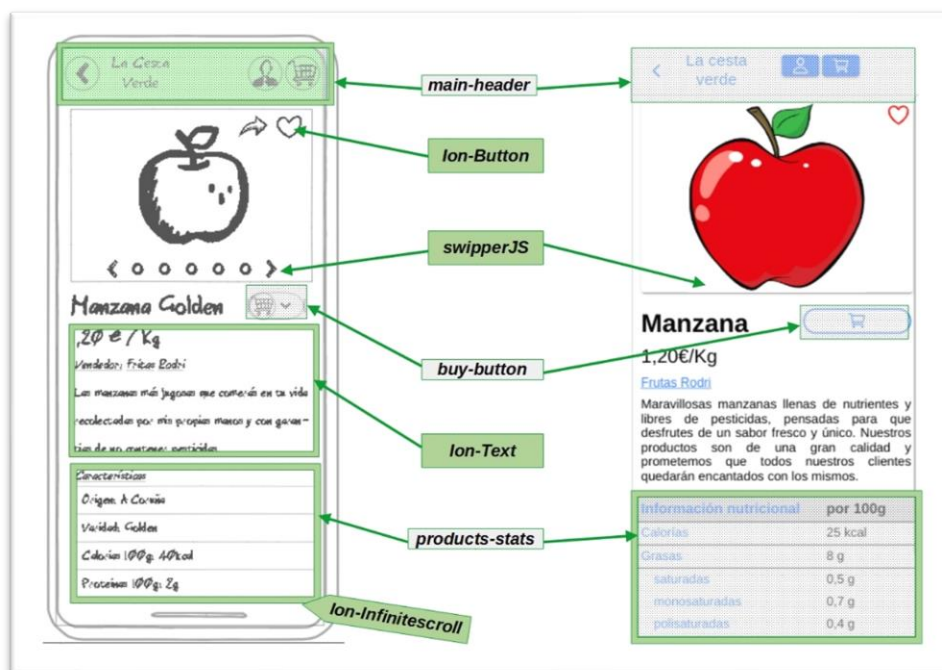


Ilustración 26: Correspondencia de elementos entre el mockUp y la aplicación móvil en el artículo



Ilustración 27: Correspondencia de elementos entre el mockUp y la aplicación móvil en la cesta

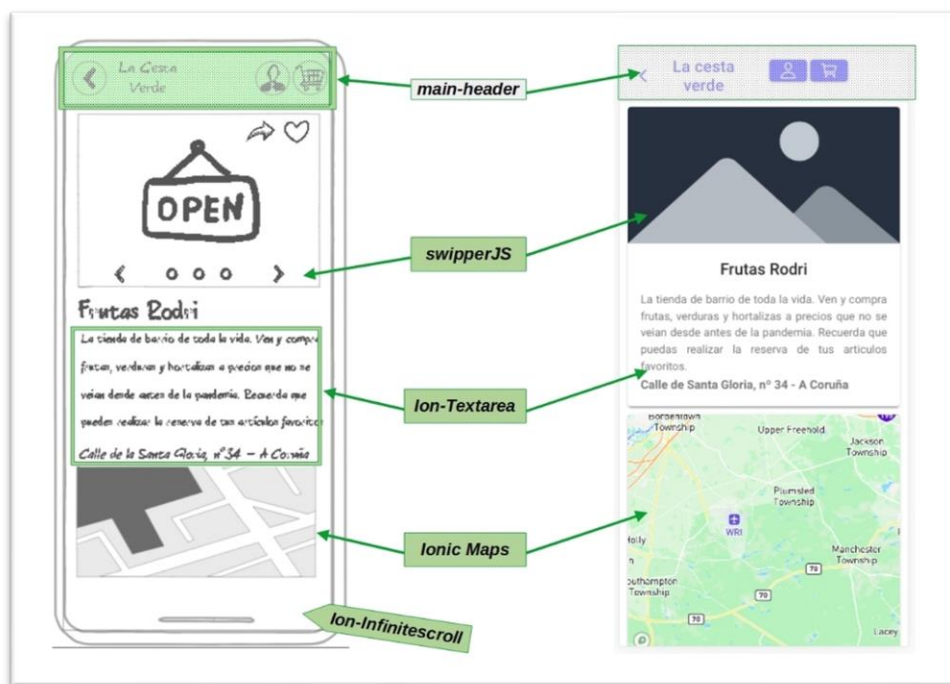


Ilustración 28: Correspondencia de elementos entre el mockUp y la aplicación móvil en el negocio



Ilustración 29: Correspondencia de elementos entre el mockUp y la aplicación móvil en el perfil

4.9 Paso 9: Desarrollo de componentes propios

Para la correcta implementación de la aplicación se han desarrollado un total de siete componentes propios, cuya implementación dentro de cada pantalla puede visualizarse en la sección de maquetación. A continuación, podemos ver tanto la apariencia como los elementos *HTML* que componen cada uno de dichos elementos:

El componente *Business-card* se utiliza para visualizar una pequeña tarjeta con información relevante sobre un negocio. Se emplea en la sección de perfil con los negocios vinculados al mismo.



Ilustración 30: Visualización y estructura del componente business-card



Ilustración 31: Visualización y estructura del componente buy-button



Ilustración 32: Visualización y estructura del componente filter-list

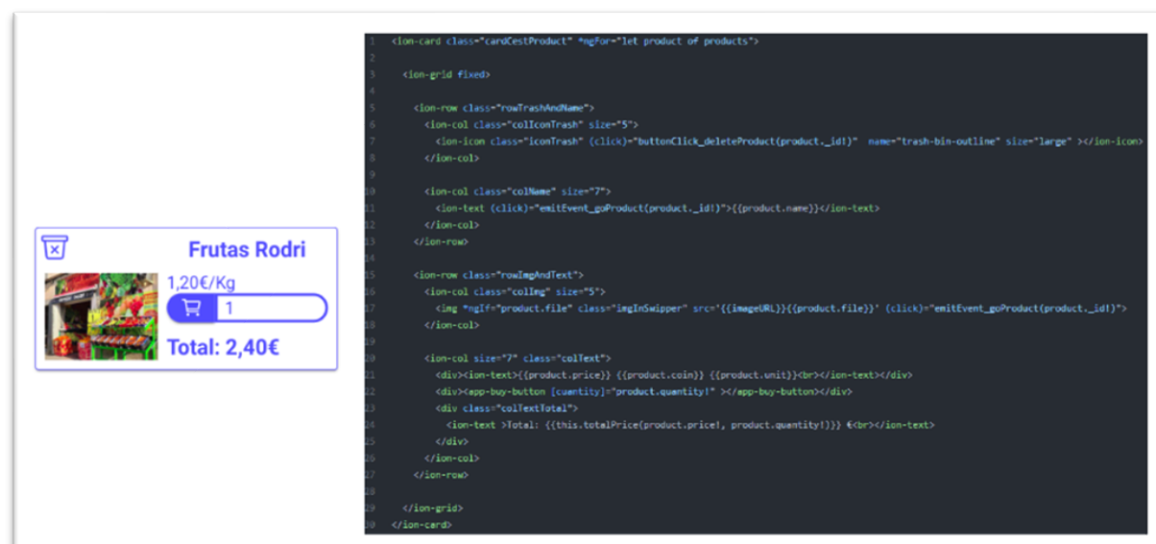


Ilustración 33: Visualización y estructura del componente cest-product-cart

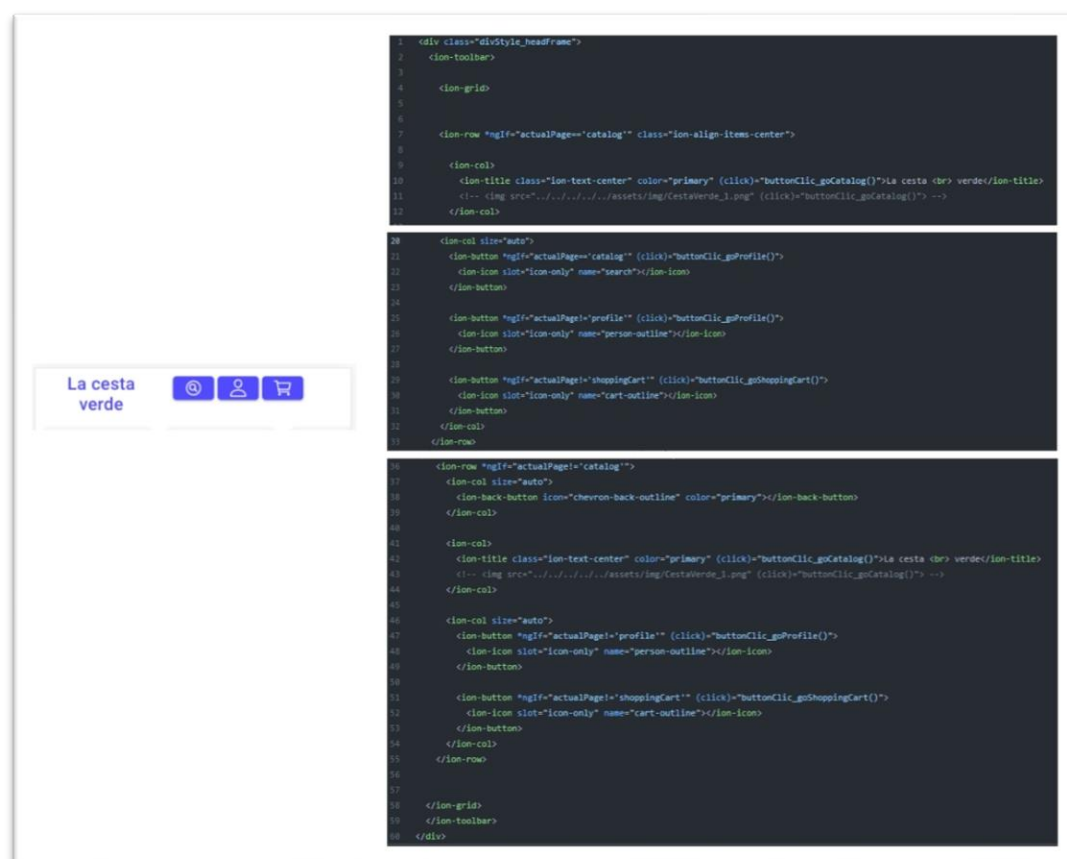


Ilustración 34: Visualización y estructura del componente main-header



```

1 <ion-card class="cardInCatalog">
2
3   <ion-img src="{{imageURL}}{{product.file}}" (click)="emitEvent_goProduct()"></ion-img>
4   <ion-icon *ngIf="this.favorite" class="buttonFavorite" (click)="buttonClick_favorite()" name="heart-sharp"></ion-icon>
5   <ion-icon *ngIf="!this.favorite" class="buttonFavorite" (click)="buttonClick_favorite()" name="heart-outline"></ion-icon>
6
7   <ion-grid fixed>
8     <ion-row>
9       <ion-title class="ion-text-center ion-text-blond" (click)="emitEvent_goProduct()"><b>{{product.name}}</b></ion-title>
10    </ion-row>
11
12    <ion-row>
13      <ion-title class="ion-text-center ion-text-blond">{{product.price}}{{product.coin}}{{product.unit}}</ion-title>
14    </ion-row>
15
16    <ion-row>
17      <ion-col size="12">
18        <app-buy-button [product] = "product"></app-buy-button>
19      </ion-col>
20    </ion-row>
21  </ion-grid>
22
23
24 </ion-card>

```

Ilustración 35: Visualización y estructura del componente product-cart

Información nutricional	por 100g
Calorías	25 kcal
Grasas	8 g
saturadas	0,5 g
monosaturadas	0,7 g
polisaturadas	0,4 g

```

1 <ion-grid fixed class="gridStyle">
2
3   <!-- ROW CALORIES -->
4   <ion-row class="rowHead">
5     <ion-col size="8"><ion-text color="primary">Información nutricional</ion-text></ion-col>
6     <ion-col size="4"><ion-text>por {{nutritionalInfo.cantidad}}</ion-text></ion-col>
7   </ion-row>
8
9   <!-- ROW CALORIES -->
10  <ion-row class="rowLv1">
11    <ion-col size="8"><ion-text color="primary">Calorías</ion-text></ion-col>
12    <ion-col size="4"><ion-text>{{nutritionalInfo.calorias}}</ion-text></ion-col>
13  </ion-row>
14
15  <!-- ROW FATS -->
16  <ion-row class="rowLv1">
17    <ion-col size="8"><ion-text color="primary">Grasas</ion-text></ion-col>
18    <ion-col size="4"><ion-text>{{nutritionalInfo.grasa_total}}</ion-text></ion-col>
19  </ion-row>
20
21  <!-- ROW SATURATE FATS -->
22  <ion-row class="rowLv2">
23    <ion-col size="8"><ion-text class="ion-margin-start" color="primary">saturadas</ion-text></ion-col>
24    <ion-col size="4"><ion-text>{{nutritionalInfo.grasa_saturada}}</ion-text></ion-col>
25  </ion-row>
26
27  <!-- ROW CARBOHYDRATES -->
28  <ion-row class="rowLv1">
29    <ion-col size="8"><ion-text color="primary">Hidratos de carbono</ion-text></ion-col>
30    <ion-col size="4"><ion-text>{{nutritionalInfo.carbohidratos_total}}</ion-text></ion-col>
31  </ion-row>
32
33  <!-- ROW POLISATURATE FATS -->
34  <ion-row class="rowLv2">
35    <ion-col size="8"><ion-text class="ion-margin-start" color="primary">azúcares</ion-text></ion-col>
36    <ion-col size="4"><ion-text>{{nutritionalInfo.azucar}}</ion-text></ion-col>
37  </ion-row>
38
39  <!-- ROW PROTEINES -->
40  <ion-row class="rowLv1">
41    <ion-col size="8"><ion-text color="primary">Proteínas</ion-text></ion-col>
42    <ion-col size="4"><ion-text>{{nutritionalInfo.proteina}}</ion-text></ion-col>
43  </ion-row>
44
45  <!-- ROW FIBRAS -->
46  <ion-row class="rowLv1">
47    <ion-col size="8"><ion-text color="primary">Fibras</ion-text></ion-col>
48    <ion-col size="4"><ion-text>{{nutritionalInfo.fibras}}</ion-text></ion-col>
49  </ion-row>

```

Ilustración 36: Visualización y estructura del componente product-stats

4.10 Paso 10: Maquetación de las escenas del Frontend para el modo Escritorio

Para la implementación de las escenas compatibles con el modo escritorio se ha tenido que identificar el tipo de dispositivo donde se está ejecutando la aplicación. Esto se ha llevado a cabo mediante la utilización del componente *Ionic-platform*, la cual nos devuelve una identificación de la plataforma, permitiéndonos desarrollar un código *HTML* diferente para ambos sistemas.

En las siguientes imágenes podemos ver unas correspondencias entre la versión de móvil y la versión de escritorio.



Ilustración 37: Correspondencia entre móvil y escritorio en el acceso

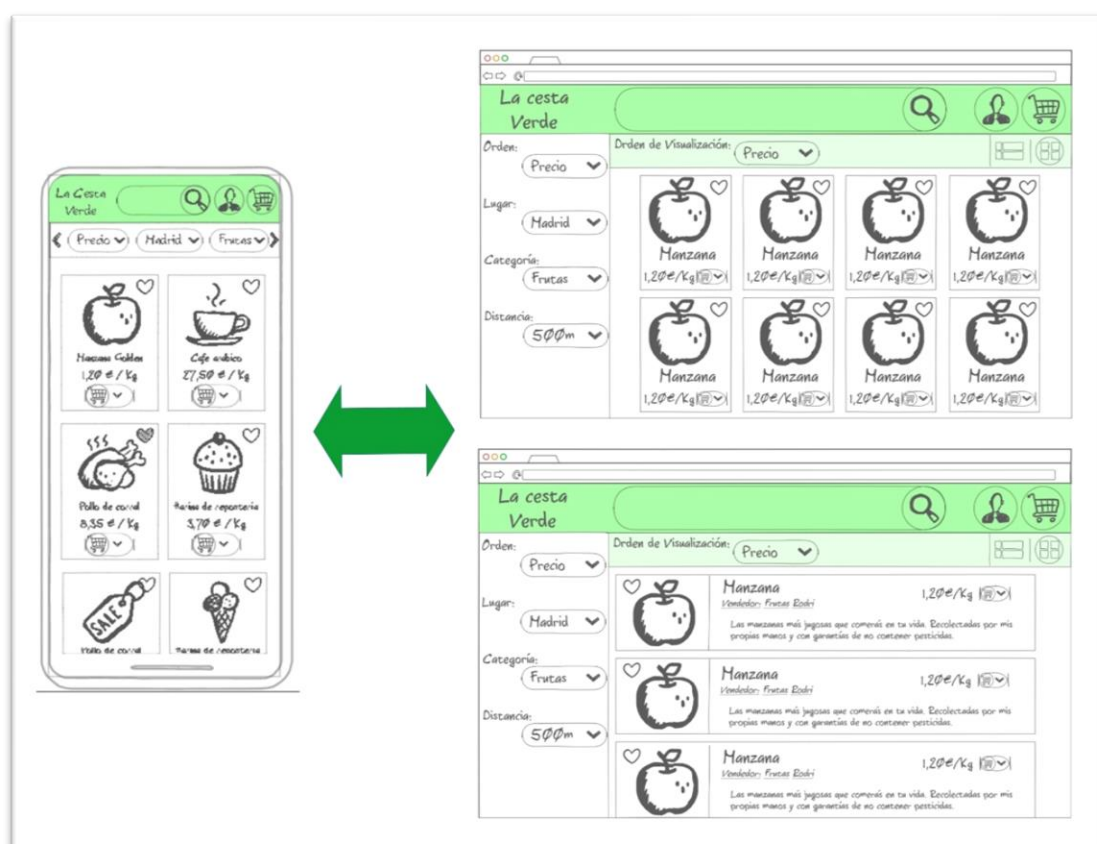


Ilustración 38: Correspondencia entre móvil y escritorio en el catálogo

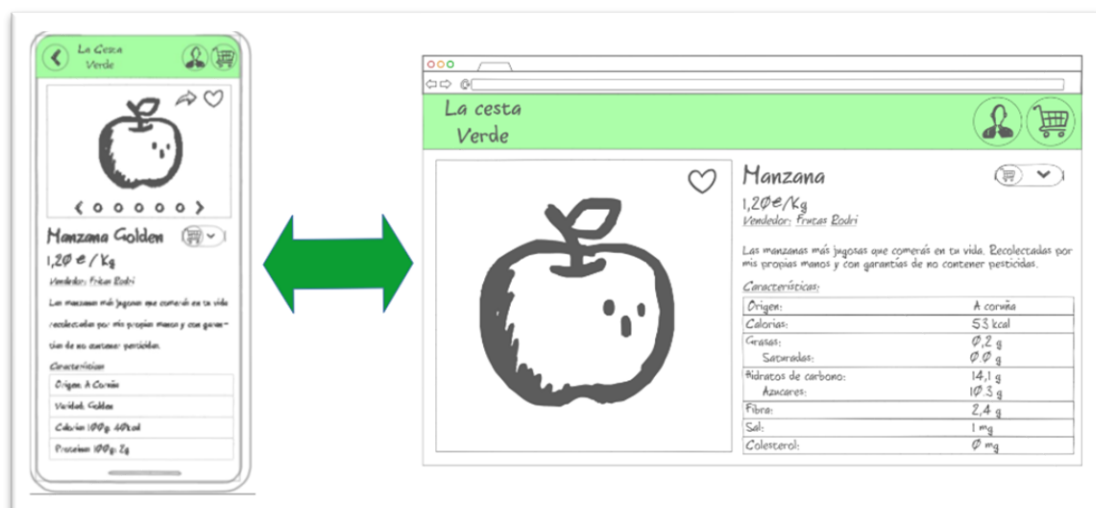


Ilustración 39: Correspondencia entre móvil y escritorio en el producto



Ilustración 40: Correspondencia entre móvil y escritorio en el perfil

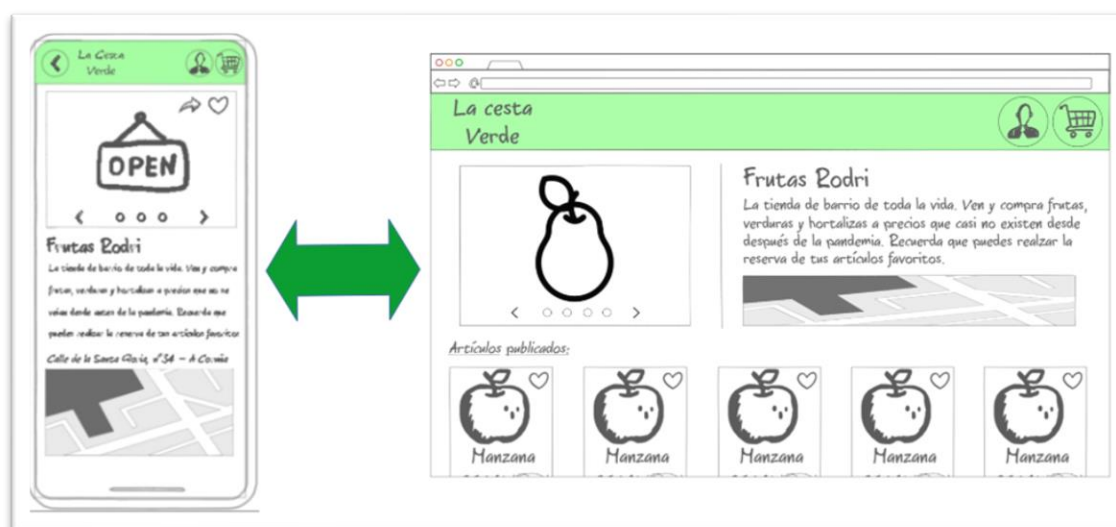


Ilustración 41: Correspondencia entre móvil y escritorio en el negocio

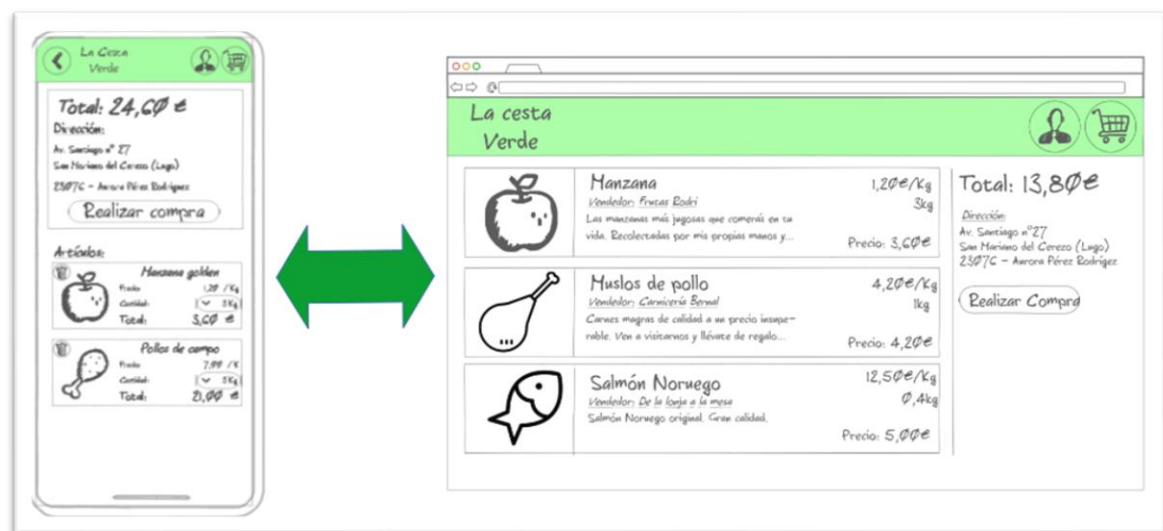


Ilustración 42: Correspondencia entre móvil y escritorio en la cesta

5. Manual de desarrollador

5.1 Peticiones del backend

Para poner en marcha el backend desde la terminal de comandos, accedemos a la raíz del proyecto y utilizamos el comando `npm start`.

Durante el desarrollo del backend hemos utilizado la plataforma Postman para construir y usar la API con la que se establecerán las comunicaciones hacia y desde el frontend. Postman incluye un conjunto integral de herramientas que ayudan a acelerar el ciclo de vida de la API, desde el diseño, pruebas, documentación y simulación hasta el uso compartido y la detección de sus API.

Durante el desarrollo del proyecto fue bastante útil para probar el correcto funcionamiento de los métodos implementados en nuestro backend. En la descripción de las peticiones se pueden ver imágenes de la aplicación Postman, con la que se pretende ilustrar el ensamblado de dichas peticiones. El backend atiende a las siguientes peticiones, las cuales describiremos con mayor detalle más adelante:

- Registro de usuario (<http://localhost:3000/api/v1/user>).
- Acceso (<http://localhost:3000/api/v1/login>).
- Registro de negocio (<http://localhost:3000/api/v1/business>).
- Obtener negocios registrados (<http://localhost:3000/api/v1/business>).
- Registro de categorías (<http://localhost:3000/api/v1/category>).
- Obtener categorías (<http://localhost:3000/api/v1/category>).
- Registro de producto (<http://localhost:3000/api/v1/product>).
- Obtener producto (<http://localhost:3000/api/v1/product>).
- Obtener datos nutricionales (<http://localhost:3000/api/v1/product/nutritionalInfo>).

5.1.1 POST: Registro de usuario

Para dar de alta un usuario en la plataforma, se necesitan los siguientes datos: Nombre, apellidos, email, nombre de usuario y la contraseña.

Desde la plataforma Postman, hacemos una petición de tipo POST al backend mediante la ruta <http://localhost:3000/api/v1/user>. El cuerpo de esta petición debe ser un documento json que contenga los datos del usuario mencionados anteriormente, como vemos en la imagen:

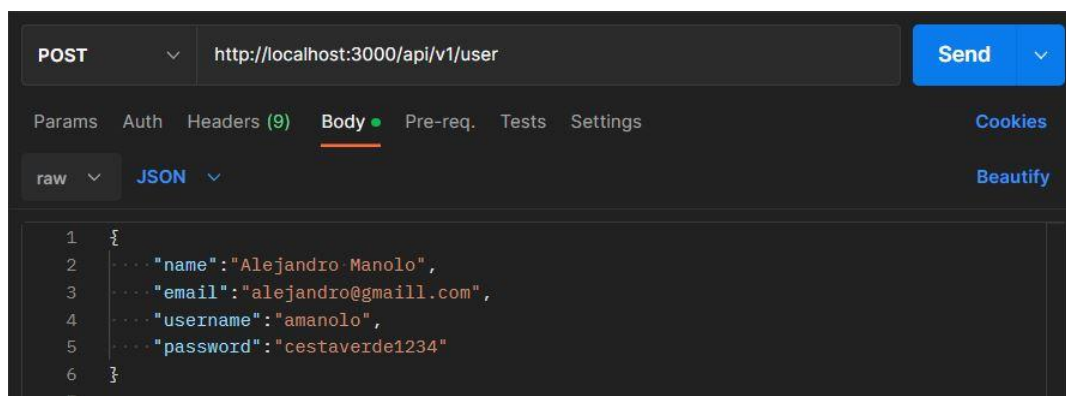
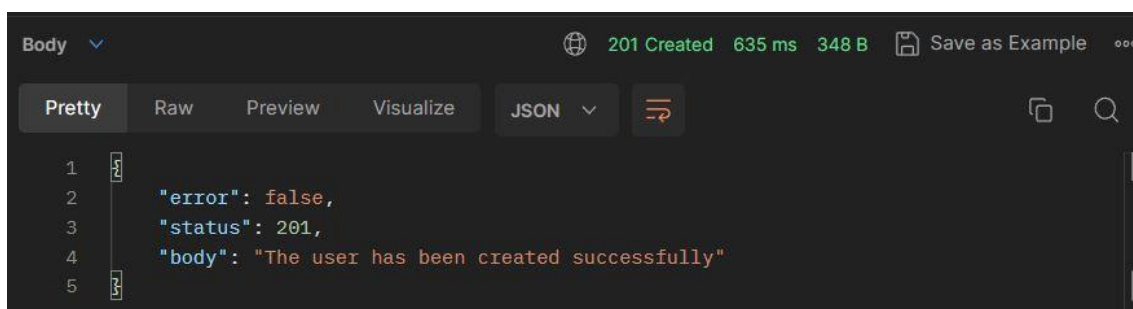


Ilustración 43: Registro del usuario utilizando el entorno de desarrollo Postman. Fuente: Elaboración propia

Si todos los datos han sido introducidos correctamente, tras haber enviado la petición al backend (haciendo clic en botón *Send*), se registra el usuario y la plataforma nos devuelve una respuesta de tipo json que contiene:

- **Status:** Código de identificación del tipo de respuesta obtenida.
- **Body:** Mensaje proporcionando información adicional sobre la respuesta obtenida.
- **Error:** Valor booleano que nos indica si se ha producido un error.

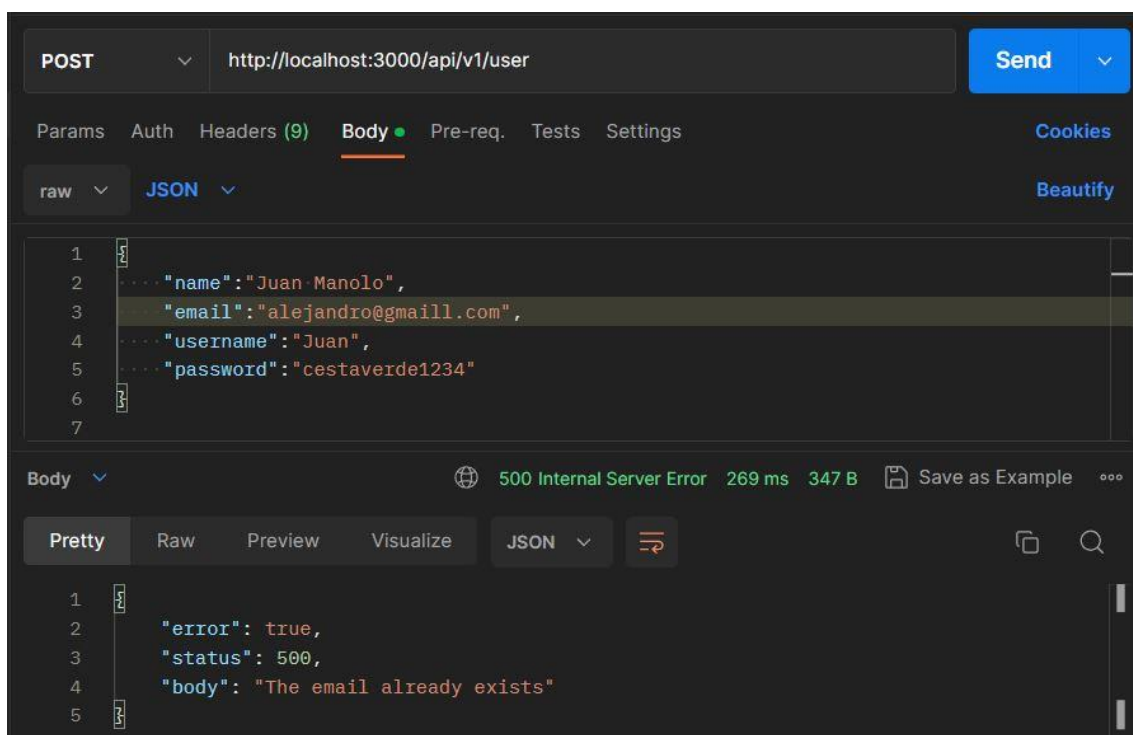
En la siguiente imagen podemos ver un ejemplo de respuesta donde se pueden ver los parámetros que la componen:



```
Body
201 Created 635 ms 348 B Save as Example
Pretty Raw Preview Visualize JSON
1
2   "error": false,
3   "status": 201,
4   "body": "The user has been created successfully"
5
```

Ilustración 44: Respuesta de la creación del usuario. Fuente: Elaboración propia

Por otra parte, tanto el email como el nombre del usuario proporcionados en la petición de registro deben ser únicos. En las siguientes imágenes podemos ver ejemplos donde intentamos realizar el registro de un usuario con el nombre de usuario o email ya existentes en la aplicación.



```
POST http://localhost:3000/api/v1/user Send
Params Auth Headers (9) Body Pre-req. Tests Settings Cookies Beautify
raw JSON
1
2   .... "name": "Juan-Manolo",
3   .... "email": "alejandro@gmail.com",
4   .... "username": "Juan",
5   .... "password": "cestaverde1234"
6
7
500 Internal Server Error 269 ms 347 B Save as Example
Pretty Raw Preview Visualize JSON
1
2   "error": true,
3   "status": 500,
4   "body": "The email already exists"
5
```

Ilustración 45: Creación del usuario con el email existente. Fuente: Elaboración propia

Como podemos ver, el backend nos ha devuelto una respuesta que contiene un error con código 500 y una descripción que nos informa que el email proporcionado ya se encuentra en uso.

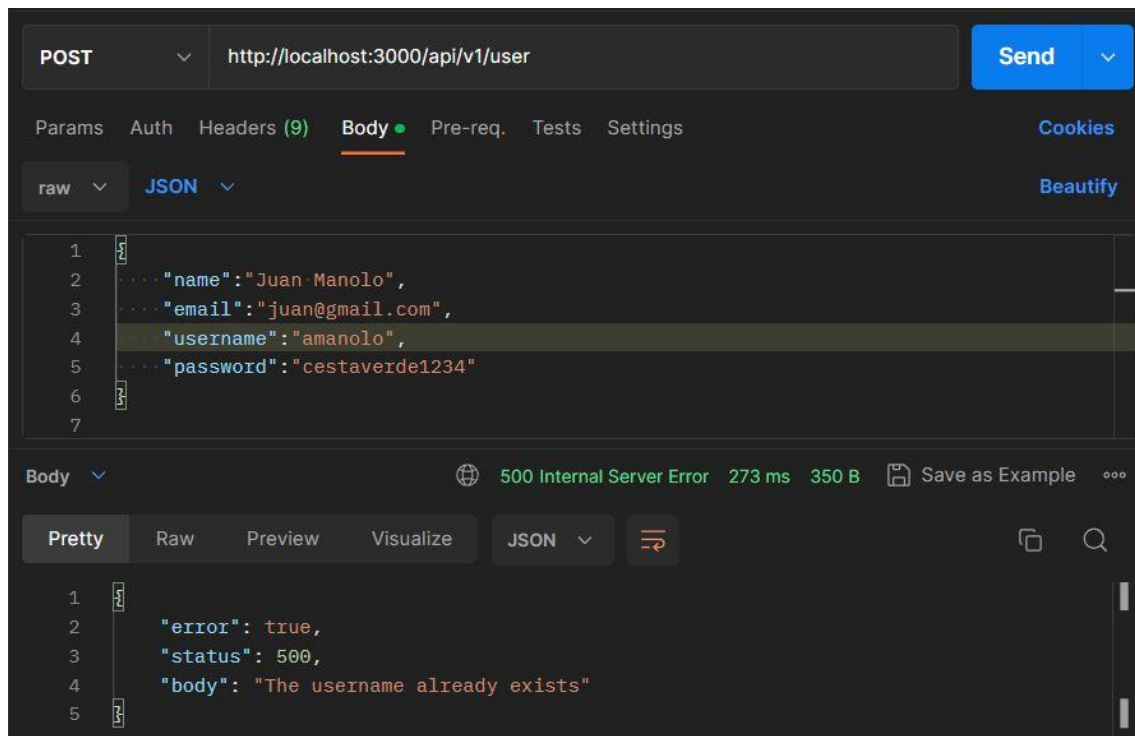


Ilustración 46: Creación del usuario con el nombre de usuario existente. Fuente: Elaboración propia

Al igual que en el caso anterior, podemos ver como el backend retorna tanto error, como el código y el mensaje de información.

5.1.2 POST: Acceso

Todos los usuarios registrados en la plataforma pueden realizar el proceso de acceso e identificación para realización de diversas actividades en función del rol que se les ha otorgado.

Para llevar a cabo el proceso de acceso, enviamos una petición de tipo POST al backend mediante la ruta <http://localhost:3000/api/v1/login> con los datos de autenticación del usuario (en este caso únicamente el usuario y la contraseña). Si el usuario existe en la base de datos de la plataforma devolverá los datos de este y el token de autenticación que se podrá usar desde el frontend para comprar la seguridad y el tiempo de duración que el usuario pasa autenticado.

En la siguiente imagen podemos ver la estructura de una petición de acceso y la respuesta dada por el servidor. Este responde con los siguientes parámetros:

- **Message:** Mensaje proporcionando información adicional sobre la respuesta obtenida.
- **User:** Información relativa al usuario que ha realizado el acceso. Esta contiene los siguientes elementos:
 - **Id:** Número de identificación del usuario dentro de la aplicación.
 - **Name:** Nombre personal del usuario dentro de la aplicación.
 - **Email:** Correo electrónico al que está asociada la cuenta.
 - **Username:** Nombre de identificación del usuario dentro de la cuenta.
 - **Fecha_de_alta:** Fecha en la que se produjo el registro de la cuenta.

- **Token:** JasonWebToken generado tras el acceso y usado para medidas de seguridad.

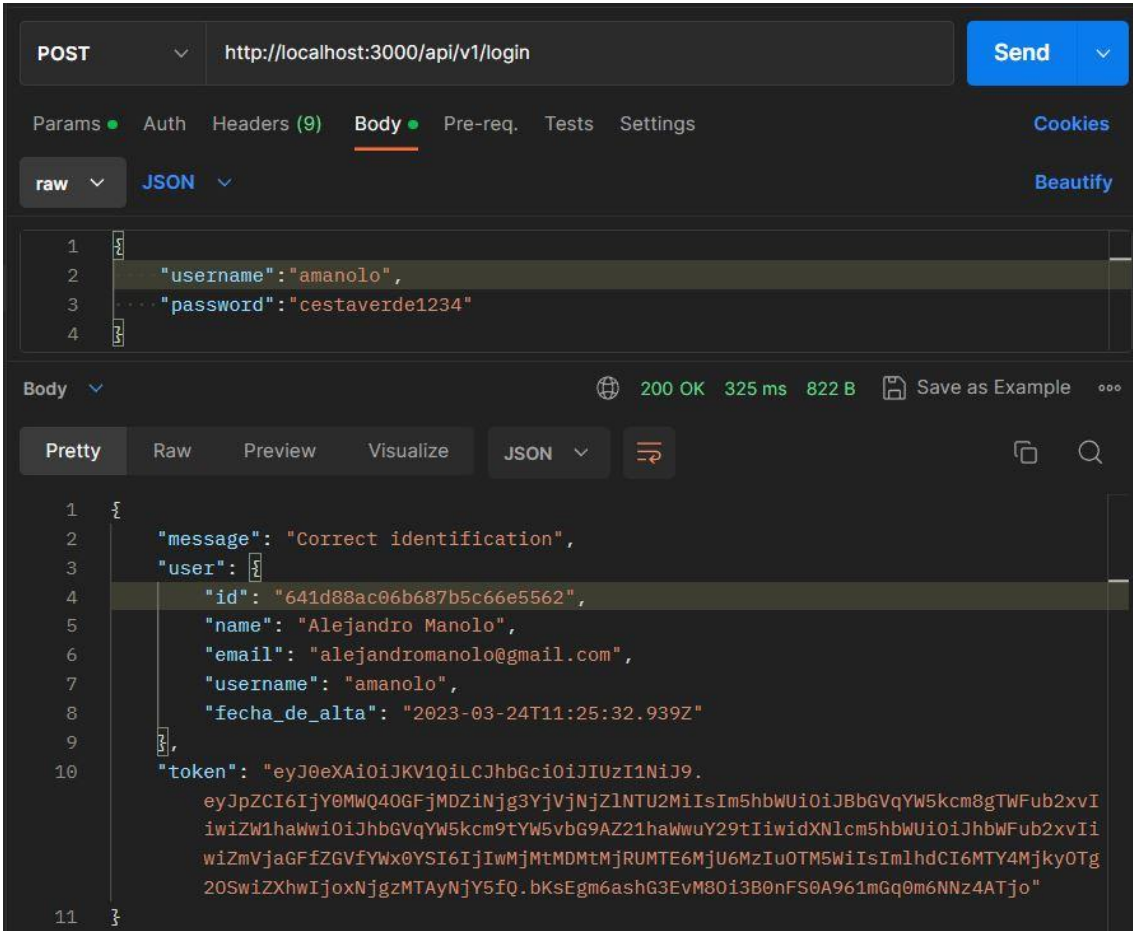


Ilustración 47: Autenticación con los datos existentes. Fuente: Elaboración propia

En el caso de proporcionar un usuario no existente, la plataforma mostrará un mensaje de error especificando que los datos son incorrectos. En la siguiente imagen podemos ver un ejemplo:

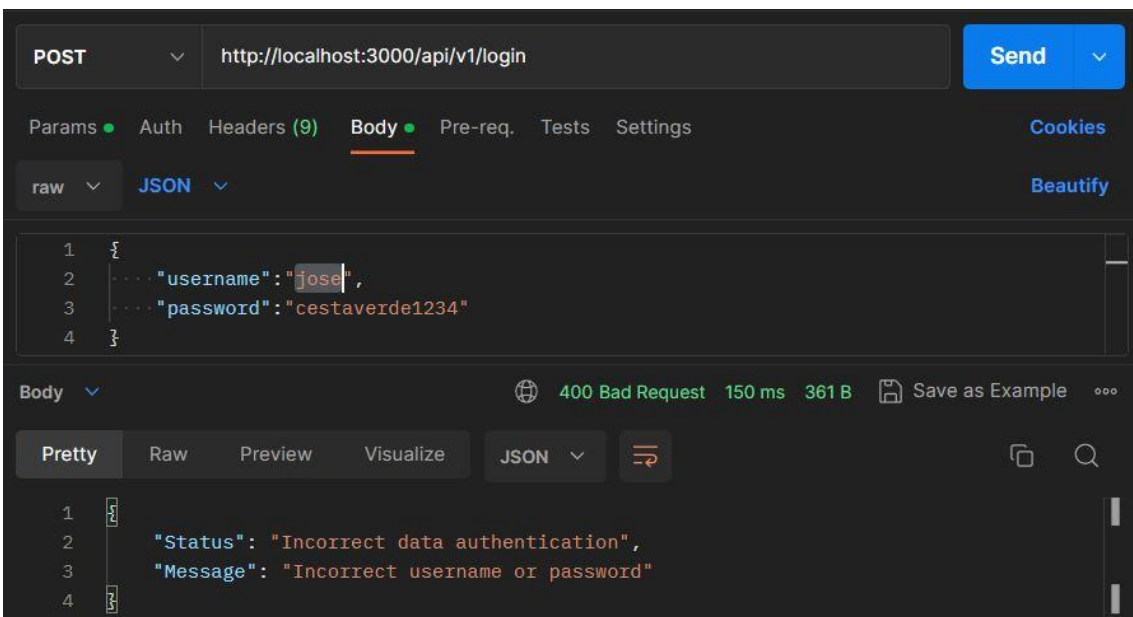


Ilustración 48: Autenticación con los datos de usuario incorrectos. Fuente: Elaboración propia

En el caso de no proporcionar los datos de usuario necesarios para llevar a cabo el acceso, el backend devuelve un error:

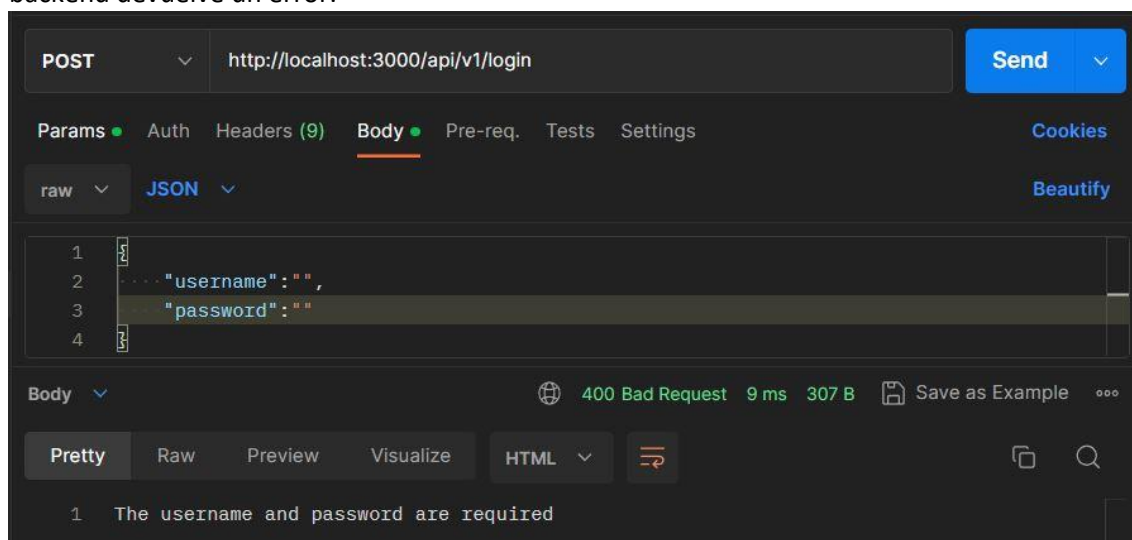


Ilustración 49: Autenticación sin datos del usuario. Fuente: Elaboración propia

5.1.3 POST: Registro de negocio

Tal como se ha mostrado anteriormente en el diseño de la base de datos, un usuario puede tener uno o muchos negocios. Para efectuar el registro de un negocio a la plataforma, enviamos una petición de tipo POST al backend a través de la ruta <http://localhost:3000/api/v1/business>.

El cuerpo de esta petición debe ser un json que contenga los datos del negocio (nombre del negocio, descripción, dirección, coordenadas para la localización en el mapa y el identificador del usuario). Si todos los campos han sido introducidos de forma correcta y se comprueba que existe el identificador del usuario proporcionado, se creará el negocio y el sistema devolverá tanto una respuesta con el estatus 201 como el mensaje de creación satisfactoria del negocio.

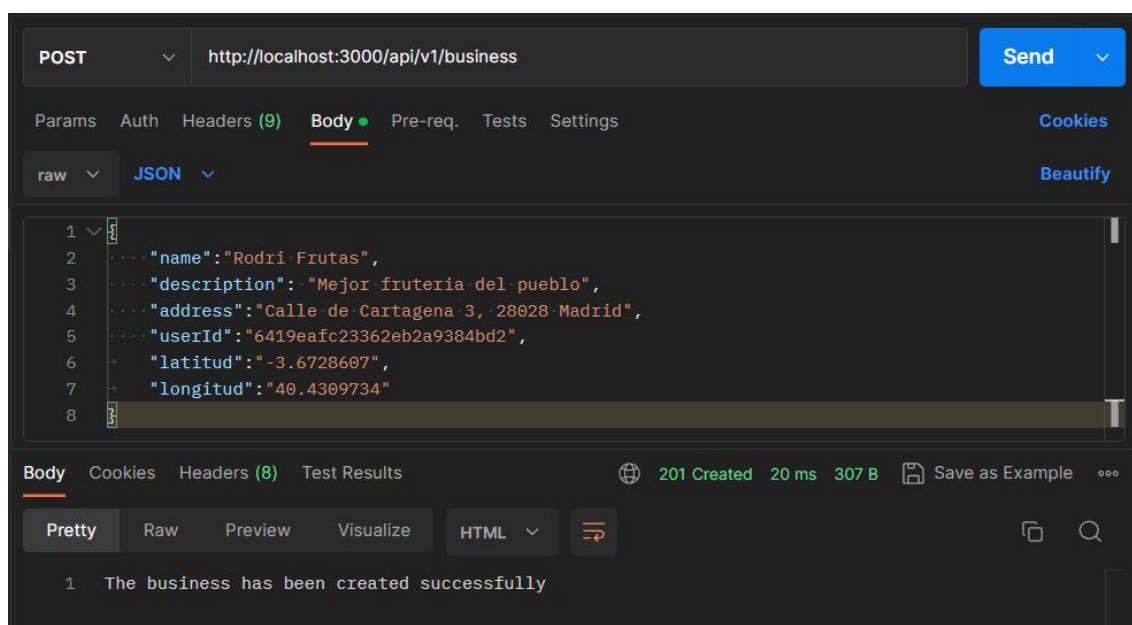
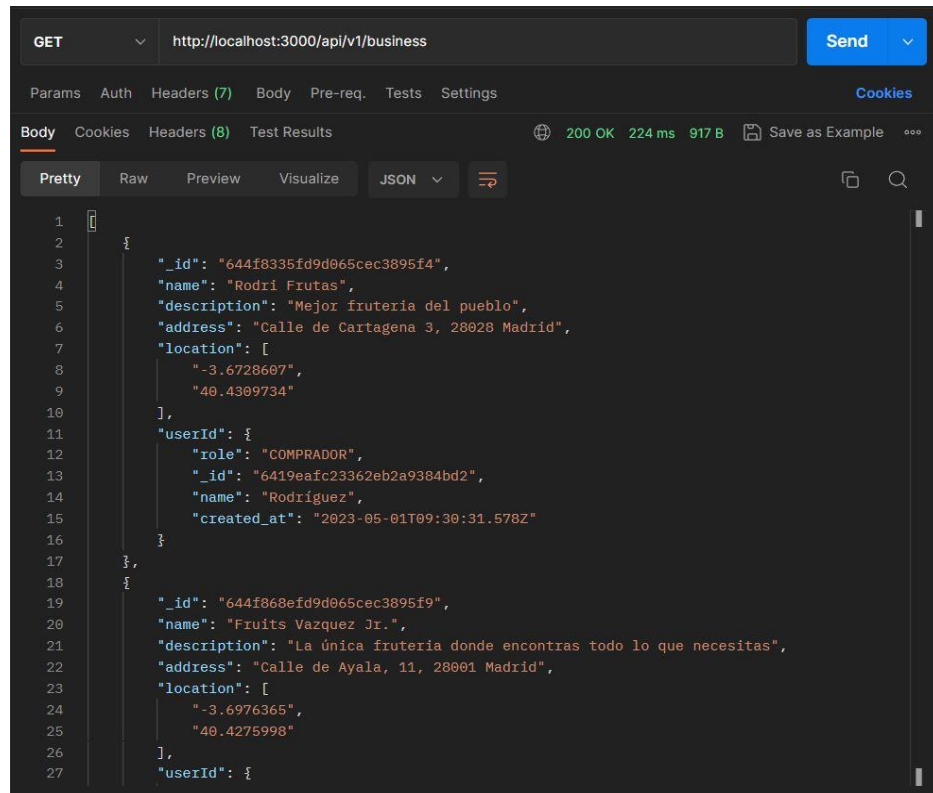


Ilustración 50: Creación del negocio. Fuente: Elaboración propia

5.1.4 GET: Obtener negocios registrados

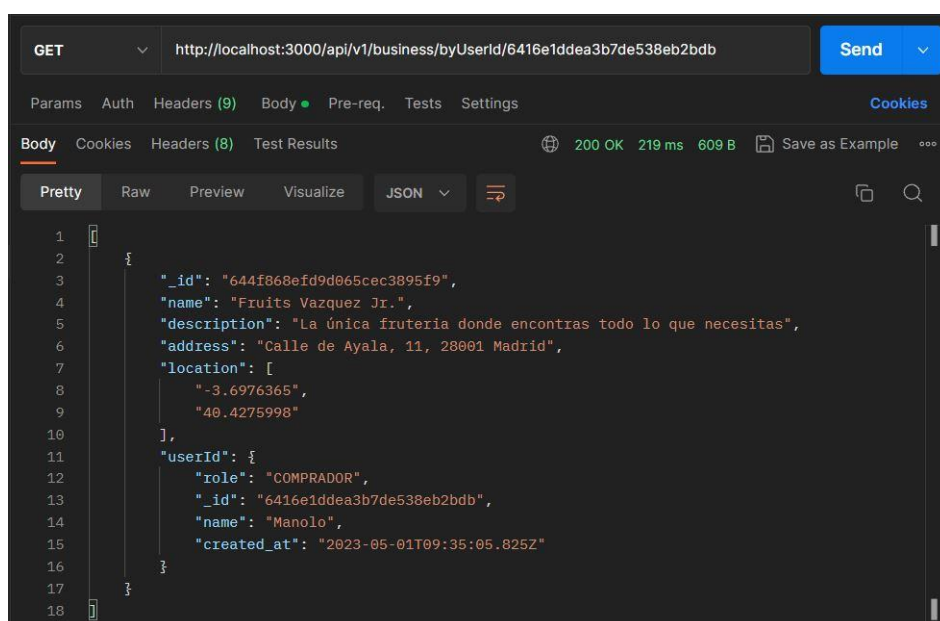
Por otra parte, podemos obtener un listado con los negocios registrados en la plataforma enviando una petición de tipo GET al backend a través de la ruta <http://localhost:3000/api/v1/business>. La respuesta incluye tanto los datos del negocio como los datos del usuario al cual se encuentra vinculado.



```
1  [
2    {
3      "_id": "644f8335fd9d065cec3895f4",
4      "name": "Rodr  Frutas",
5      "description": "Mejor fruter a del pueblo",
6      "address": "Calle de Cartagena 3, 28028 Madrid",
7      "location": [
8        "-3.6728607",
9        "40.4309734"
10     ],
11     "userId": {
12       "role": "COMPRADOR",
13       "_id": "6419eafc23362eb2a9384bd2",
14       "name": "Rod guez",
15       "created_at": "2023-05-01T09:30:31.578Z"
16     }
17   },
18   {
19     "_id": "644f868efd9d065cec3895f9",
20     "name": "Fruits Vazquez Jr.",
21     "description": "La  nica fruter a donde encuentras todo lo que necesitas",
22     "address": "Calle de Ayala, 11, 28001 Madrid",
23     "location": [
24       "-3.6976365",
25       "40.4275998"
26     ],
27     "userId": {
```

Ilustraci n 51: Listado de los negocios registrados en la plataforma. Fuente: Elaboraci n propia

Si hacemos la misma petici n al backend, pero a la ruta le a adimos el identificador del usuario, obtendr amos todos los negocios correspondientes al usuario con dicho identificador.



```
1  {
2    "_id": "644f868efd9d065cec3895f9",
3    "name": "Fruits Vazquez Jr.",
4    "description": "La  nica fruter a donde encuentras todo lo que necesitas",
5    "address": "Calle de Ayala, 11, 28001 Madrid",
6    "location": [
7      "-3.6976365",
8      "40.4275998"
9    ],
10   "userId": {
11     "role": "COMPRADOR",
12     "_id": "6416e1ddea3b7de538eb2bdb",
13     "name": "Manolo",
14     "created_at": "2023-05-01T09:35:05.825Z"
15   }
16 }
```

Ilustraci n 52: Negocios registrado a un usuario espec fico. Fuente: Elaboraci n propia

5.1.5 POST: Registro de categorías

Dentro de la plataforma, las categorías son importantes a la hora de registrar los productos, puesto que permiten a los negocios tener una agrupación más controlada de estos. Las categorías podrían ser: frutas, verduras, vegetales etc.

Para registrar una categoría enviaremos una petición de tipo POST al backend a través de la ruta <http://localhost:3000/api/v1/category>. En este caso se requiere menos información, dado que solo necesitamos conocer el nombre de la categoría.

El cuerpo del mensaje devuelto por el backend a dicha petición debe ser un json que contenga la siguiente información:

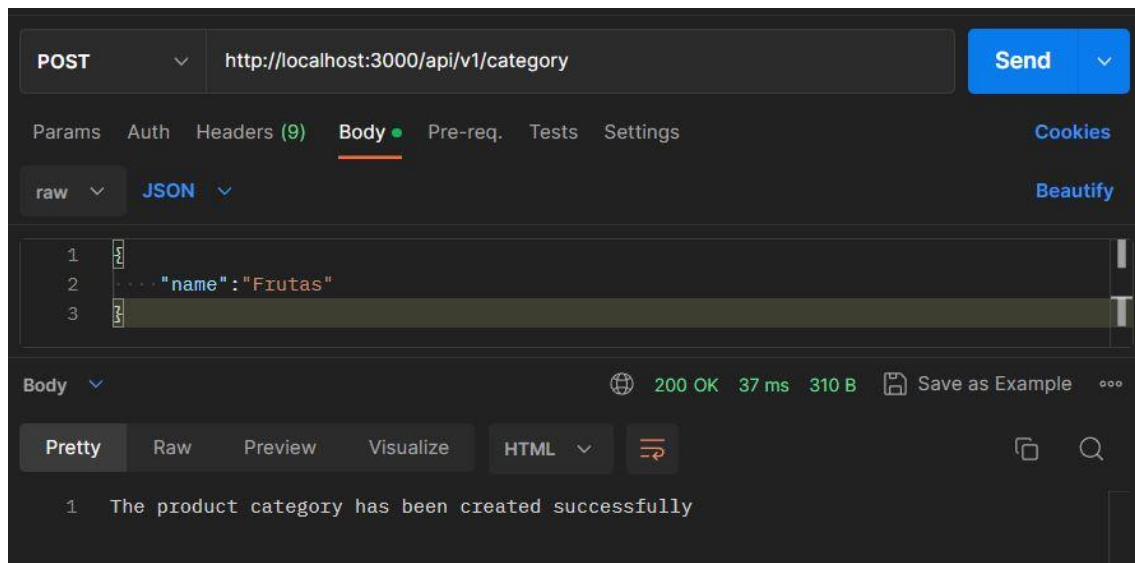


Ilustración 53: Registrar categoría. Fuente: Elaboración propia

5.1.5 GET: Obtener categorías

Podemos obtener el listado de todas las categorías registradas en la plataforma enviando una solicitud de tipo GET al servidor a través de la ruta <http://localhost:3000/api/v1/category>.

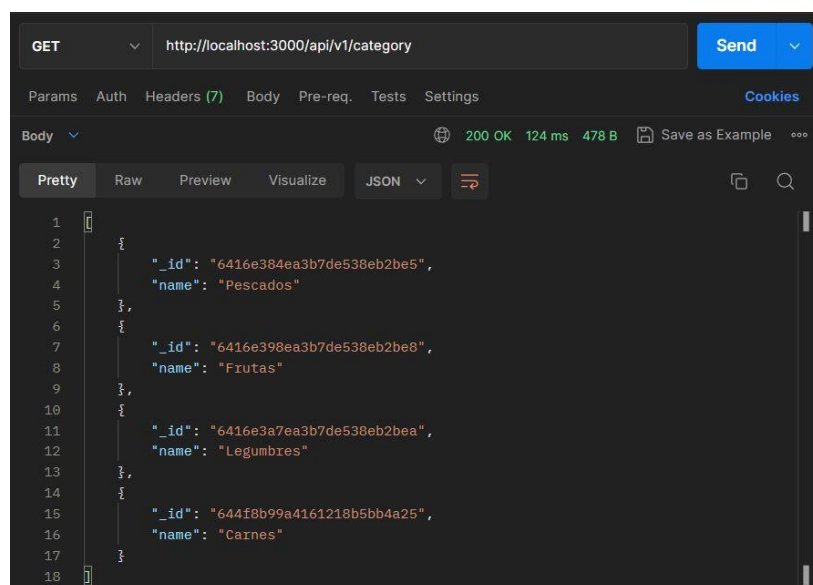


Ilustración 54: Listado de categorías. Fuente: Elaboración propia

Además, también podemos obtener una categoría específica haciendo una petición de tipo GET al servidor a través de la misma ruta, añadiendo a esta el identificador de la categoría que queremos obtener. En la siguiente imagen podemos ver un ejemplo:

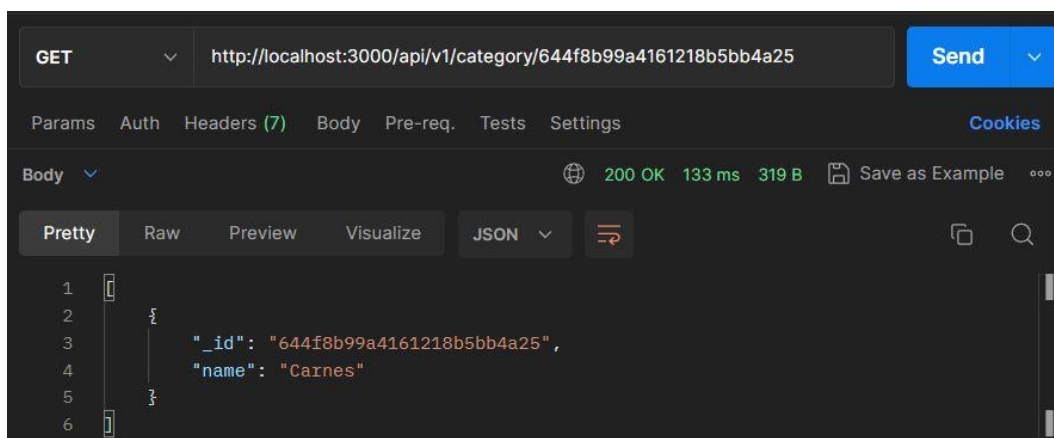


Ilustración 55: Categoría obtenida a través del id. Fuente: Elaboración propia

5.1.6: POST: Registro de producto

Una vez creada la categoría y el negocio, el usuario puede proceder con la creación del producto. Para ello hacemos una llamada de tipo POST a nuestro servidor a través de la ruta <http://localhost:3000/api/v1/product> y en el cuerpo de esta petición especificamos los siguientes atributos: nombre del producto, precio, moneda, unidad de venta, descripción del producto, categoría y el negocio.

Si los datos han sido introducidos correctamente y se valida la existencia de los que sean necesarios, el sistema devolverá la respuesta con el estatus 201 y el mensaje de creación del producto.

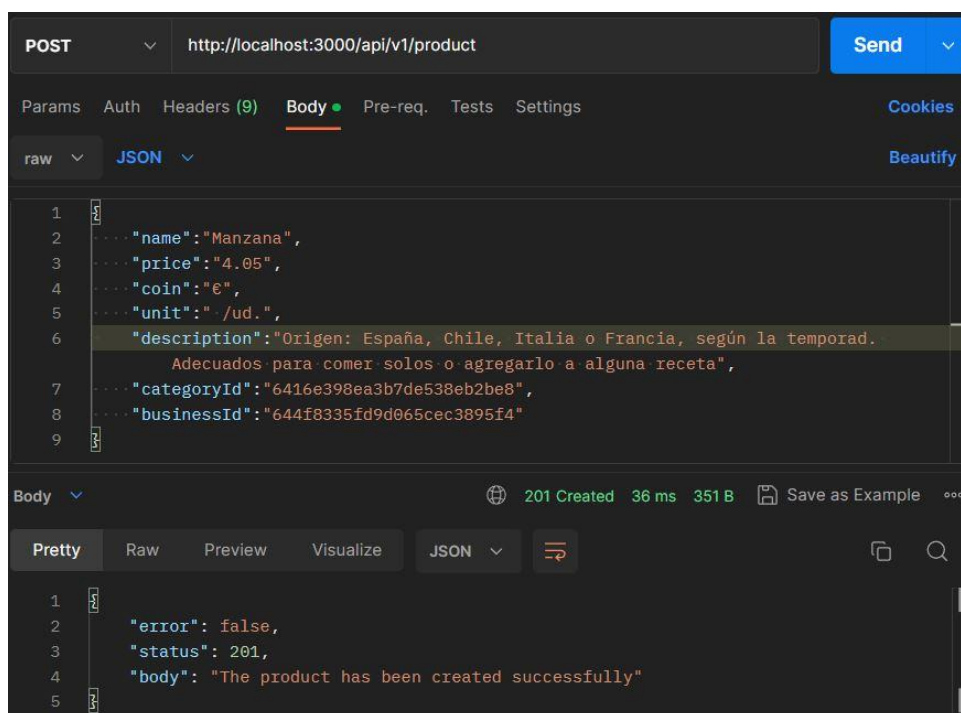


Ilustración 56: Creación del producto. Fuente: Elaboración propia

5.1.7: GET: Obtener productos

Para obtener el listado de todos los productos registrados en la plataforma, hacemos una petición de tipo GET a nuestro servidor a través de la ruta <http://localhost:3000/api/v1/product/>. En la respuesta se podrá observar los datos del producto y los datos del negocio en donde se registró cada producto.

En la siguiente imagen podemos ver la petición realizada para obtener la información de los productos registrados en la plataforma y la respuesta del servidor:

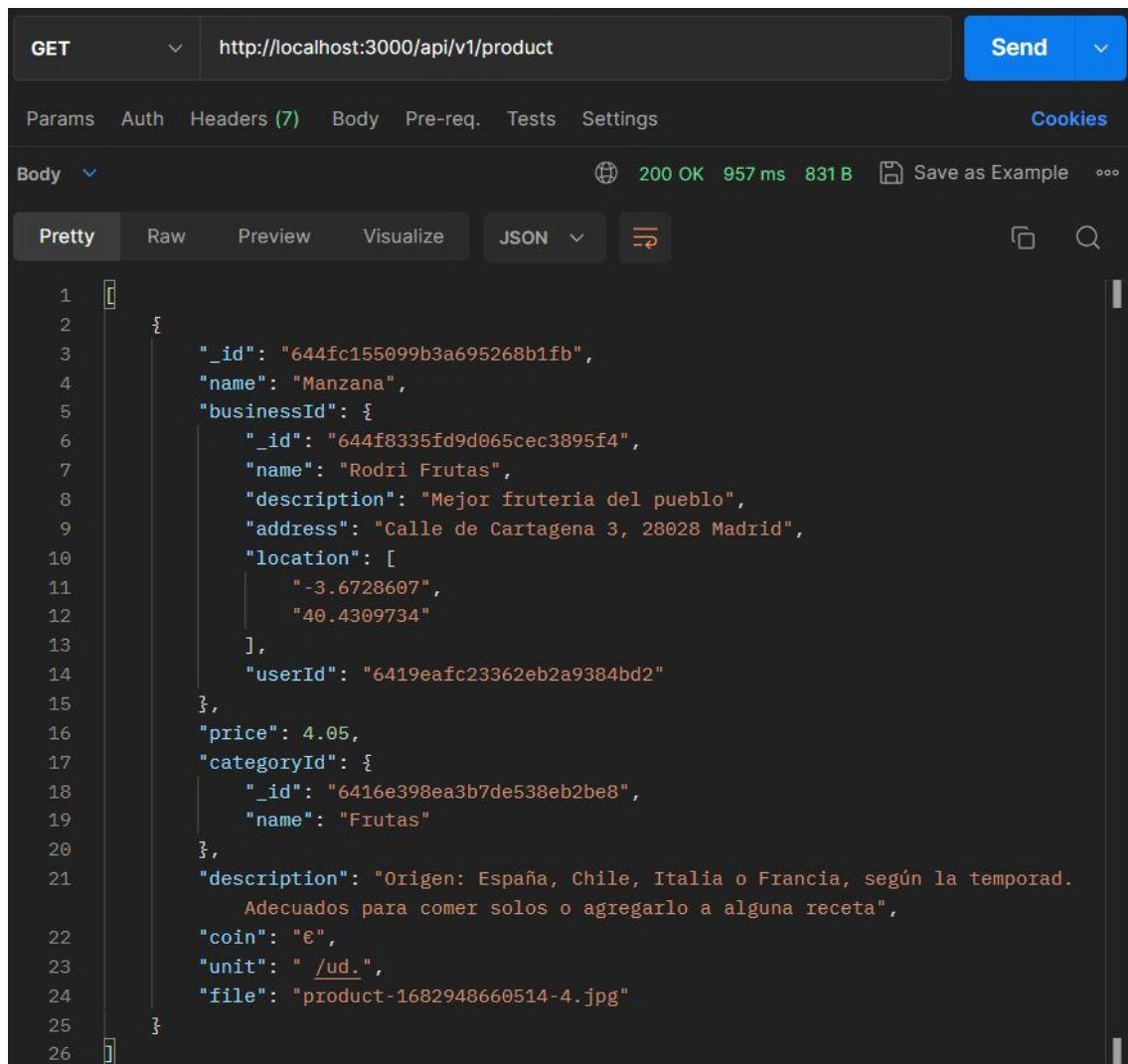


Ilustración 57: Obtener listado de los productos registrados en la plataforma. Fuente: Elaboración propia

De la misma forma, se puede obtener la información de un determinado producto enviando una petición de tipo GET a nuestro servidor a través de la ruta <http://localhost:3000/api/v1/product/product/'id'>, en este caso añadimos a la ruta el identificador del producto que queremos obtener su información, tal como se observa en la siguiente imagen.

En la siguiente imagen podemos ver la petición realizada para obtener la información de un producto concreto y la respuesta del servidor:

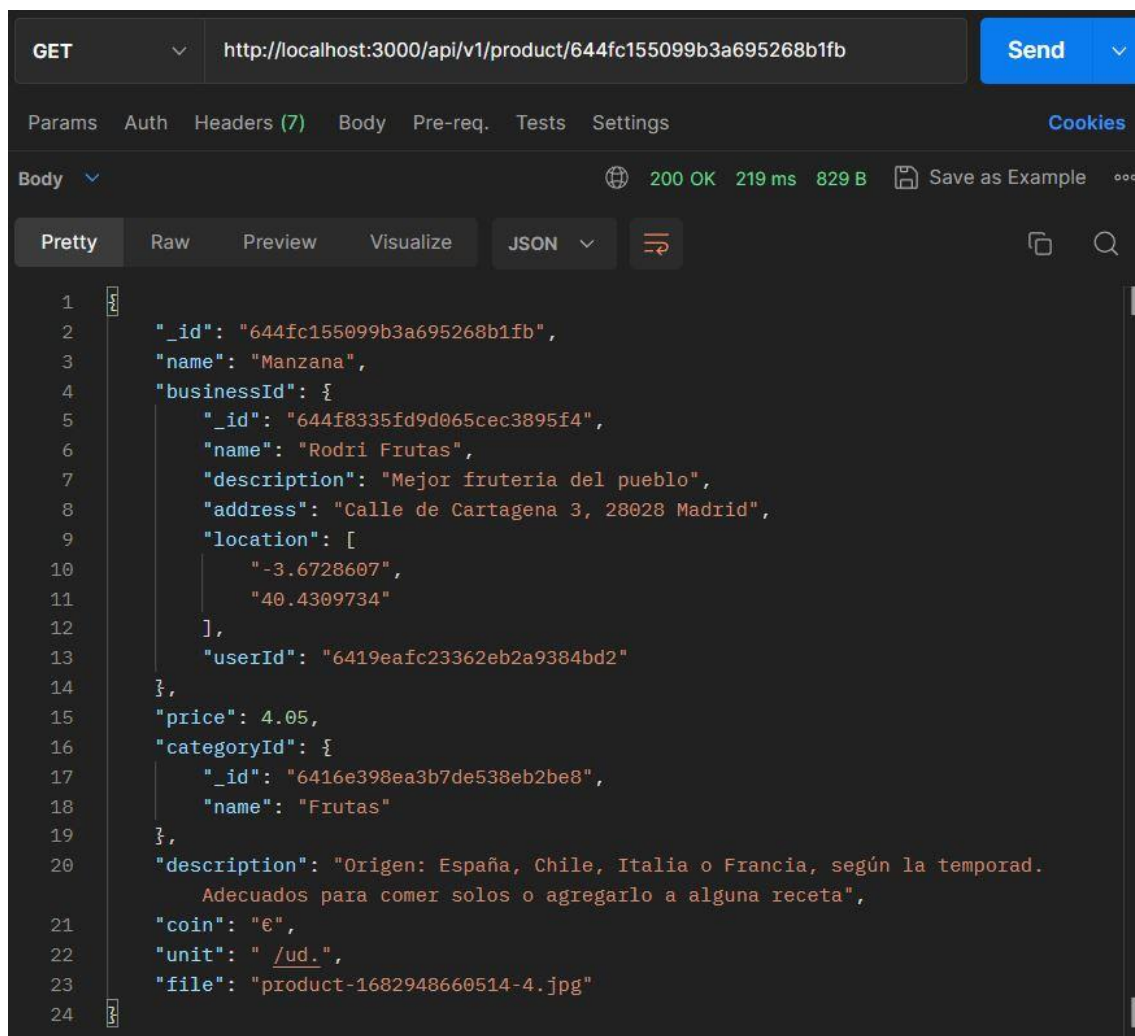
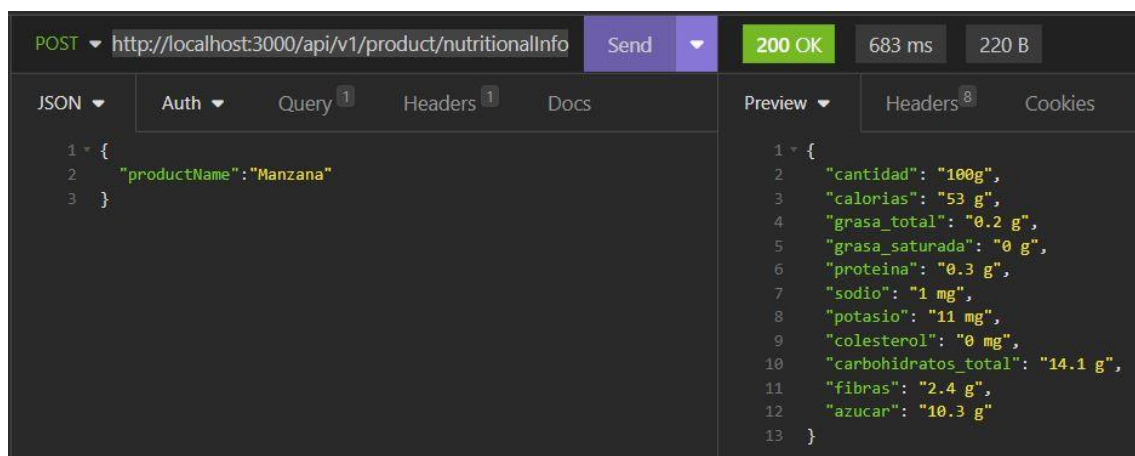


Ilustración 58: Producto obtenido a través de la id. Fuente: Elaboración propia.

5.1.8: GET: Obtener información nutricional

Para obtener la información nutricional de un determinado producto desde nuestro servidor, necesitamos enviar una petición de tipo GET a través de la ruta <http://localhost:3000/api/v1/product/nutritionalInfo>, y en el cuerpo de la petición, colocamos el nombre del producto.



6. Conclusiones

Tras la realización del presente proyecto se ha llegado a las siguientes conclusiones:

- Se ha logrado desarrollar la implementación de un Frontend mediante el uso de aplicaciones híbridas, llevando a cabo varias versiones de las mismas pantallas con el objetivo de que puedan ser utilizadas en diferentes dispositivos.
- Se ha diseñado la interacción del usuario para cada versión de la aplicación, primando la correcta representación de la información y facilitando su uso en base a la plataforma utilizada.
- Se ha llevado a cabo la compenetración del Frontend y al Backend para poder mostrar la información tramitada en el sistema correctamente. Teniendo en cuenta también el uso de bases de datos.
- Se ha logrado realizar llamadas a APIs de terceros para automatizar y facilitar el uso de contenido multimedia a través de la aplicación. Entre este contenido se encuentra:
 - La obtención de información nutricional relevante para los usuarios y su presentación en un formato de fácil entendimiento.
 - Generación de códigos QR para los productos subidos a la aplicación.
 - Generación de mapas para mostrar las localizaciones de los locales.
- Definición de estilos acordes a la temática de la aplicación, teniendo en cuenta colores y formas, incluyendo un logotipo personalizado. Se ha generado un Dossier con dichos elementos y una breve explicación.

7. Bibliografía

- **Proyecto Github:** <https://github.com/AvocadoTeamUCM/LaCestaVerde>
- **Página principal ninjaMock:** <https://ninjamock.com/>
- **Página principal de Ionic:** <https://ionicframework.com/>
 - **Ion-title:** <https://ionicframework.com/docs/api/title>
 - **Ion-input:** <https://ionicframework.com/docs/api/input>
 - **Ion-button:** <https://ionicframework.com/docs/api/button>
 - **Ion-checkbox:** <https://ionicframework.com/docs/api/checkbox>
 - **Ion-searchbar:** <https://ionicframework.com/docs/api/searchbar>
 - **Ion-nav:** <https://ionicframework.com/docs/api/nav>
 - **Ion-toolbar:** <https://ionicframework.com/docs/api/toolbar>
 - **Ion-infinite-scroll:** <https://ionicframework.com/docs/api/infinite-scroll>
 - **Ion-textarea:** <https://ionicframework.com/docs/api/textarea>
 - **Ion-card:** <https://ionicframework.com/docs/api/card>
 - **Ionic platform:** <https://ionicframework.com/docs/angular/platform>
- **Página principal de Angular:** <https://angular.io/>
- **Página principal de Node.js:** <https://nodejs.org/es>
- **Página principal de express.js:** <https://expressjs.com/>
- **Página principal de Docker:** <https://www.docker.com/>
- **Página principal de MongoDB:** <https://www.mongodb.com/>
 - **MongoDB Compass:** <https://www.mongodb.com/products/compass>
 - **MongoDB Atlas:** <https://www.mongodb.com/es/atlas/database>