WebApp: Alimentación Sostenible

Memoria de Proyecto Final de Grado/Máster

Máster universitario en Ingeniería Informática

El presente cuadro de texto tiene solamente fines informativos y no debe ser incluido en la memoria del alumno.

ACERCA DE LOS CONTENIDOS EN ESTE DOCUMENTO

Este documento incluye estilos predeterminados de texto, ejemplos de citas bibliográficas, notas a pie e inserción de figuras (imágenes y gráficos) y tablas, así como sección de bibliografía e índices automatizados listos para usar.

ACERCA DE LOS CAPÍTULOS DE ESTE DOCUMENTO

Aquellos capítulos con el título en color negro son obligatorios para todos los TF, mientras que aquellos en color gris son aquellos opcionales, susceptibles de ser incluidos en la memoria según el tipo de TF realizado. Es recomendable adaptar el orden de los capítulos a la naturaleza del TF a realizar, e incluso combinar dos o más capítulos en uno si se considera oportuno.

Autor: Marta Moral González

Consultor: Joan Giner Miguelez

Profesor: David García Solórzano

Fecha de entrega

03/03/2020

Créditos/Copyright

Abstract

El presente proyecto propone el desarrollo de una aplicación web con el objetivo de reducir el desaprovechamiento de alimentos que se está produciendo en la actualidad (aproximadamente un tercio de los alimentos destinados al consumo humano). La aplicación web de alimentación sostenible, permitirá que los comercios puedan publicar sus ofertas de productos a punto de caducar en la aplicación de forma que éstas lleguen a un mayor número de consumidores potencialmente interesados, y así mismo, permita a tales consumidores consultar qué productos a precio reducido están disponibles en los comercios dados de alta en la aplicación. De esta forma, se reduce el desaprovechamiento de los alimentos, a la vez que los consumidores ahorran y los comercios recuperan parte del dinero de la venta. De esta forma, todos ganamos.

La aplicación web de alimentación sostenible, objeto del presente proyecto, consta de una parte servidora desarrollada como un API REST implementado sobre Spring Boot, a través del cual se implementan las diferentes lógicas de negocio de las que consta la aplicación y una parte cliente, desarrollada con VueJS, HTML, Javascript y CSS, a través de la cual, tanto los usuarios consumidores como los comercios podrán interactuar con las diferentes funcionalidades ofrecidas por la aplicación.

Palabras clave: Alimentación sostenible, consumo responsable, aplicación web, Spring Boot, VueJS, API REST

Abstract (english version)

The present project proposes the development of a web application with the objective of reducing the waste of food that is currently taking place (approximately a third of the food destined for human consumption). The sustainable food web application will allow merchants to publish their offers of products that are about to expire in the application so that they reach a greater number of potentially interested consumers, and likewise, allow such consumers to consult which products with reduced prices are available in the stores registered in the application. In this way, food waste is reduced, while consumers save money and businesses recover part of the money from the sale. In this way, we all win.

The sustainable feeding web application, object of this project, consists of a server part developed as a REST API implemented on Spring Boot, through which the different business logics of which the application consists and a client part, developed with VueJS, HTML, Javascript and CSS, through which both consumer users and stores can interact with the different functionalities offered by the application.

Palabras clave: sustainable food, responsible consumption, web application, Spring Boot, VueJS, REST API

Notaciones y Convenciones

El formato del texto para los párrafos documentativos del presente documento utiliza la fuente Arial con tamaño 10, con justificación de texto e interlineado de 1,5, por ejemplo este propio párrafo.

El formato de texto para los fragmentos de código fuente utiliza la fuente Courier New en tamaño 10, sin justificación de texto e interlineado de 1,5, ejemplo:

```
@Configuration
@EnableWebSecurity
```

Índice

Una o varias páginas con el índice de la memoria, que debe incluir los títulos de los capítulos (estilo de texto Heading 1) así como sus secciones de primer nivel (estilo de texto Heading 2), sin profundizar más en la estructura.

Figuras y tablas

Lista de imágenes, tablas, gráficos, diagramas, etc., numeradas, con títulos y las páginas en las que aparecen.

Índice de figuras

Índice de tablas

1. Introducción/Prefacio

Teniendo en cuenta que, en nuestros días, aproximadamente una tercera parte de los alimentos destinados para el consumo humano se desperdicia en el mundo, o lo que es equivalente, 1300 millones de toneladas de alimentos se desperdician al año [1].

Además, también se desperdician los recursos necesarios para la producción de estos alimentos, y la contaminación que estos procesos de producción generan y que son perjudiciales para la salud del planeta no es justificable ya que el producto generado por los mismos no se aprovecha.

El desaprovechamiento de los alimentos es uno de los problemas del denominado “primer mundo”, ya que se produce más en las regiones con más renta per cápita que en los países en desarrollo. De hecho, FAO (2012) calcula: “ que el desperdicio per cápita de alimentos por consumidor en Europa y América del Norte es de 95 a 115 kg/año, mientras que en el África subsahariana y en Asia meridional y sudoriental esta cifra representa solo de 6 a 11 kg/año”(p.5).

Algunos comercios, de forma localizada en cada tienda, suelen ofrecer algunos productos a punto de caducar a precio reducido cada día para reducir el desaprovechamiento de los alimentos por su parte.

Siguiendo este tipo de iniciativas realizadas por algunos comercios para reducir el problema, se pretende realizar como objeto del presente TFM, el planteamiento, análisis, diseño e implementación de una aplicación web que tenga como fin permitir que los comercios puedan publicar sus ofertas de productos a punto de caducar en la aplicación de forma que éstas lleguen a un mayor número de consumidores potencialmente interesados, y así mismo, permita a tales consumidores consultar qué productos a precio reducido están disponibles en los comercios en base a diferentes filtros.

Así, los comercios podrán hacer llegar las ofertas de los productos a punto de caducar a un mayor número de consumidores, lo cual permitirá el mejor aprovechamiento de tales alimentos que en otro caso se desperdiciarían y que el comercio pueda aún sacar cierto beneficio por la venta de estos productos, que en otro caso no sacaría.

También los consumidores se verán beneficiados del uso de la aplicación, pues colaborarán con un modelo de alimentación más sostenible, reduciendo el desperdicio de alimentos, lo cual además beneficiará su economía familiar pues estos productos estarán en oferta a un precio reducido.

2. Descripción/Definición/Hipótesis

Teniendo en cuenta las cifras de comida desperdiciada mundialmente, y en especial en el primer mundo, así como las acuciantes noticias acerca de la contaminación que amenaza nuestro planeta, se ha decidido tratar aportar un granito de arena en la labor de la alimentación sostenible, mediante el desarrollo de una aplicación web que pueda facilitar el aprovechamiento de alimentos a punto de caducar, que en la mayoría de las ocasiones son desperdiciados. Existen otras aplicaciones con el mismo objetivo de reducir el desaprovechamiento de alimentos, estas pueden consultarse en en el apartado: *Análisis de Mercado* del presente documento.

Por tanto, como se verá en apartados posteriores del presente documento, el objetivo de este trabajo fin de máster es el planteamiento, diseño e implementación de una aplicación web que permita a los comercios publicar ofertas de productos cuya fecha de caducidad está próxima de forma que puedan llegar fácilmente a consumidores potenciales de los mismos, de forma que el comercio pueda obtener parte del beneficio en la venta de este producto rebajado (que en otros casos no tendría) y los usuarios puedan obtener productos para su consumo diario más económicos y además colaboren con el aprovechamiento de la alimentación.

Los comercios dados de alta en la aplicación podrán ver, buscar y dar de alta productos en la aplicación, así como añadir, editar y eliminar ofertas sobre los mismos que se encuentren disponibles en su tienda.

Los usuarios dados de alta en la aplicación, podrán ver y buscar productos en la aplicación, así como ver y buscar las ofertas publicadas por los diferentes comercios. Además, los usuarios podrán guardar productos como favoritos, de forma que recibirán una notificación en la aplicación cuando algún comercio publique alguna oferta sobre ellos, y guardar ofertas en su lista de favoritos para poder consultarlas cuando deseen.

Además, se proporcionan herramientas para la gestión de los datos del perfil tanto para las cuentas que sean de usuario consumidor como las cuentas que sean de comercio.

A continuación se muestra el diagrama de casos de uso con las funcionalidades a alto nivel proporcionadas por la aplicación de alimentación sostenible:

Figura 1: Diagrama de casos de uso general

Para más información pueden consultarse los apartados del presente documento: *Funcionalidades*, *Arquitectura de la aplicación, Diagramas UML y Usabilidad/UX*.

3. Objetivos

El objetivo de la aplicación web objeto de este TFM, es el facilitar el aprovechamiento de los alimentos que estén a punto de caducar de forma que se promueva un modelo de alimentación más sostenible y permita beneficiarse tanto a los comercios como a los consumidores de colaborar en tal iniciativa.

Así, los comercios que deseen hacer llegar sus ofertas (normalmente locales a la propia tienda) a un mayor número de consumidores y obtener aún beneficios por productos que de otro modo se desperdiciarían, podrán registrarse en la aplicación web como comercio y hacer llegar sus ofertas por productos a punto de caducar a un mayor número de usuarios.

Los consumidores, por su parte, también podrán unirse al movimiento de la alimentación sostenible registrándose en la aplicación y pudiendo ver todas las ofertas que los comercios están dado de alta acerca de productos que están a punto de caducar y pudiendo ahorrar así un dinero en la compra de sus productos de uso diario.

4. Análisis de Mercado

4.1 Love Food Hate Waste

Es una aplicación web en la que se busca concienciar a los usuarios acerca del desperdicio de alimentos y del impacto ambiental que esto conlleva, y propone que no se compre más de lo estrictamente necesario y que se aproveche la totalidad de los productos comprados antes de hacer una nueva compra. Para ello, ofrece un recetario de platos que pueden cocinarse a partir de los alimentos de los que se dispone, así como un planificador para comprar, almacenar y consumir los alimentos.

Es posible encontrar más información acerca de esta aplicación en su sitio web oficial: https://www.lovefoodhatewaste.com/

4.2 Yo no desperdicio

Dispone de página web y aplicación móvil disponible para IOS y Android, desarrollada por la ONG Prosalus, y permite compartir los alimentos excedentes que no van a ser consumidos por aquellos usuarios que deseen compartir sus alimentos, bien con otros usuarios o con diferentes organizaciones . Además, también proporciona un listado de ofertas, e ideas como recetas de aprovechamiento y claves para la compra responsable.

Es posible encontrar más información acerca de esta aplicación en su sitio web oficial: https://yonodesperdicio.org

4.3 Expire

Esta aplicación móvil disponible para IOS nos facilita el control a la hora de evitar que se nos caduquen los productos que hayamos comprado y que tengamos en casa, pues nos permite establecer la fecha de caducidad o consumo preferente de cada uno de los mismos y configurar alertas para que tengamos tiempo de utilizarlos antes de que se estropeen.

Es posible descargarla a través del App Store: https://apps.apple.com/es/app/expire/id570190807

4.4 Too Good To Go

Esta aplicación móvil disponible para IOS y Android se propone la lucha contra el desperdicio de alimentos conectando a potenciales consumidores con restaurantes y establecimientos hosteleros que tengan excedente de productos y que los ofrecen a precios reducidos.Esta aplicación ofrece a los usuarios "packs sorpresa", cuyo contenido no se conocerá hasta que el usuario recoja el pedido, esto es así, porque a priori no es posible conocer la cantidad y tipo de excedente del que se dispondrá en cada momento, lo cual, proporciona un factor de diversión y sorpresa para los usuarios.

Por tanto, esta aplicación proporciona una opción responsable e inteligente para comer bien, de forma más económica, a la vez que se reduce el desperdicio de alimentos y por tanto, se cuida además el medio ambiente, reduciendo las emisiones de CO2.

Es posible encontrar más información acerca de esta aplicación en su sitio web oficial: https://toogoodtogo.es

5. Marco teórico

A continuación se presentan las tecnologías utilizadas para la implementación de la WebApp objeto del presente TFM.

5.1 Servicios y APIs REST

Un servicio REST puede entenderse como cualquier interfaz para el intercambio de datos utilizando el protocolo de nivel de aplicación HTTP (o HTTPS si son peticiones seguras) y que permita tanto la obtención como la creación/modificación y/o borrado de contenidos, normalmente, utilizando formato JSON o XML.

Las características básicas de los servicios web son las siguientes:

- No se mantiene el estado: No es requerido que ni cliente ni servidor tengan que mantener ningún tipo de información previa a la petición en curso para poder satisfacerla. Es decir, entre peticiones de una misma navegación de un usuario no se mantiene ninguna información.

- Cacheables: Se debe de poder admitir un sistema de caché que permita aligerar el tráfico de red ante peticiones que recuperen el mismo recurso.

- Se permiten las operaciones basadas en los métodos del protocolo HTTP, de los cuales, aquellos de uso más frecuentes son:

    - GET (obtención de datos)
    - POST (creación de datos)
    - PUT (edición de datos)
    - DELETE (eliminación de datos)

- Los recursos y operaciones ofrecidos por un API REST serán únicamente accesibles a través de una URI y de ninguna otra forma, lo cual, junto con la limitación de las operaciones a las definidas por el protocolo HTTP, hace que las APIs REST tengan una interfaz uniforme.

- Uso de hipermedios: Especifica la capacidad de las APIs REST para que parte de la información devuelta por los mismos sean hipervínculos a otros recursos asociados.

La utilización de servicios REST en las arquitecturas de las aplicaciones web supone grandes ventajas, pues al separarse totalmente la lógica del servicio de la visualización de los datos devueltos por el mismo hay una independencia completa entre la parte cliente y la parte servidora de la aplicación web.

Así, los productos implementados de esta forma serán mucho más fáciles de evolucionar y mantener, así como de migrar y escalar cada una de las partes (como los frameworks utilizados en la parte cliente, los lenguajes utilizados en la parte servidora, los sistemas de bases de datos utilizados, etc), sin influir al resto de los componentes desarrollados, siempre y cuando se mantengan intactas las interfaces de los servicios y el formato de las peticiones y respuestas de los mismos.

5.2 Sistemas de autenticación

La autenticación forma parte de una de los tres pasos fundamentales que garantizan la seguridad de un sistema:

- Autenticación: Es el proceso por el cual se asegura de manera segura que un usuario que realiza una petición es quien dice ser.
- Autorización: Es el proceso por el cual se garantiza que los usuarios puedan acceder a los recursos que tienen permitidos y sólo a esos recursos.
- Auditoría: Es el proceso por el cual se registran todos los accesos a los recursos por parte de los usuarios, tengan estos permisos o no.

La autenticación, es el primer paso para asegurar el acceso seguro de los usuarios a cualquier sistema, y, en la actualidad, existen diversas maneras de implementarlo:

- Basándose en algo conocido (por ejemplo, una contraseña)
- Basándose en algo que se posee (por ejemplo, una tarjeta de coordenadas)
- Basándose en alguna característica física (por ejemplo, una huella dactilar).

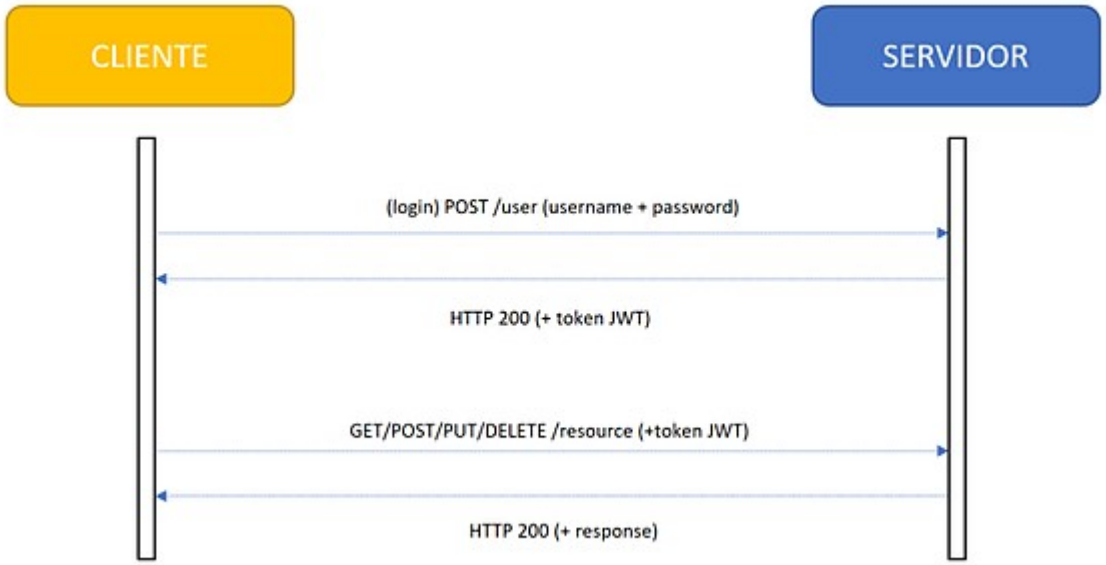
En este caso, se va a utilizar la autenticación basada en algo conocido, pues se utilizará primero una autenticación mediante usuario y password, y esta información permitirá la generación de un token JWT que será devuelto al usuario. Para peticiones subsiguientes a esta, el usuario únicamente tendrá que informar este token para autenticarse hasta que se venza el periodo de caducidad establecido por configuración para los tokens.

5.2.1 JWT

JWT es un estándar de código abierto que utiliza el lenguaje JSON para la generación de tokens de acceso que permiten otorgar seguridad las comunicaciones realizadas entre el cliente y el servidor.

Así, cuando un usuario quiere autenticarse contra un sistema, manda sus datos de login (usualmente nombre de usuario y contraseña) al servidor. El servidor por su parte, comprueba la veracidad de tales datos y, si son correctos, genera un token JWT que envía a la aplicación cliente. La aplicación cliente enviará este token en todas sus siguientes peticiones y el servidor verifica ese token para asegurar que el usuario se encuentra autenticado en el sistema.

De forma gráfica puede representarse este proceso de la siguiente forma:



[2] Figura 2: Proceso de autenticación mediante JWT.

Los JWT están formados por:

- Header: Que normalmente contiene el tipo de algoritmo utilizado para la generación de la firma y el tipo de token que va a generarse, que en este caso es JWT.
- Payload: Contiene una serie de atributos en forma de clave valor que van encriptadas en el token.
- Signature: Firma de JWT generada con el algoritmo de encriptación indicado en la header, a partir del header + el payload + clave secreta conocida únicamente por el servidor que creó el JWT.

5.3 Spring Boot

Spring Boot nace por la necesidad de simplificar la forma de crear aplicaciones con Spring Core, pues este último requiere gran dedicación aplicada a la su compleja configuración y al despliegue de la aplicación, haciendo que los desarrolladores tengan que dedicar un esfuerzo grande en estas tareas en lugar de permitirles centrarse en mayor medida en el desarrollo de la aplicación en sí mismo.

Las características principales de Spring Boot que hace que sea una de las mejores opciones para su uso con servicios REST son las siguientes:

- Configuración simplificada y automática: Spring Boot cuenta con un módulo interno que es capaz de autoconfigurar todos los aspectos de la aplicación de forma automática.
- Resolución de dependencias: El desarrollador únicamente tiene que indicar el tipo de proyecto que está realizando y Spring Boot se encargará de gestionar todas las dependencias requeridas para su correcto funcionamiento.
- Despliegue: Spring Boot permite ejecutar aplicaciones en modo Standalone, pero también permite la ejecución de aplicaciones web sin que sea necesario contar con un servidor de aplicaciones externo, ya que el propio Spring Boot cuenta con servidores de aplicaciones integrados (Tomcat, Jetty o Undertow).
- HealthCare: Spring Boot cuenta con aplicaciones que permiten conocer el estado de salud de la aplicación, si está online u offline, la memoria que está en uso actualmente y la memoria que aún queda disponible, etc.

Por todas estas características se ha decidido implementar el backend de la aplicación web objeto de este TFM con Spring Boot.

5.4 VueJS

Para la parte cliente de la aplicación web, se ha decidido utilizar VueJS, un framework open source progresivo para la construcción de interfaces de usuario, del cual una de sus principales características y ventajas es que puede ser adoptado de forma incremental.

VueJs es muy sencillo de utilizar y de integrar con otros proyectos y librerías existentes, de forma que es capaz de incrementar sus capacidades de forma incremental y simple a la vez que potente, por lo que esta ha sido la característica principal que ha hecho que sea elegido como framework con el que implementar la parte cliente de la aplicación web.

Además, VueJS cuenta con las siguientes características esenciales que lo convierten en una excelente opción para el proyecto objeto de este TFM:

- Desacoplamiento de las librerías del framework, lo que permite la inclusión únicamente de las librerías deseadas sin que sea necesario incluir partes no necesarias por ser dependencia de las mismas, y de forma tan sencilla como importar el script en la página web.
- El core principal de VueJS únicamente se encarga de renderizar los componentes visuales de la aplicación, pero es muy sencilla la inclusión de librerías tanto oficiales como desarrolladas por la comunidad que ampliarán rápidamente las capacidades disponibles.
- El funcionamiento de VueJS es tan intuitivo, en comparación con sus análogos ReactJS y Angular, que su uso cada vez está más extendido, con grandes exponentes, como Xiaomi, Alibaba o Gitlab.
- Cuenta con herramientas muy útiles que ayudan en la labor del desarrollador, como la CLI que permite la creación de un proyecto en base a una plantilla base configurada a nuestro gusto, numerosas extensiones para los navegadores Firefox y Chrome que ayudan en la depuración de las aplicaciones, además de plugins para los IDEs más utilizados.
- La comunidad de VueJS es muy activa y actualmente existe un repositorio oficial de VueJS en el que se encuentran todos los recursos relevantes como librerías, documentación, libros, etc.
- Los componentes desarrollados con VueJS almacenan todas sus partes (HTML, CSS y Javascript) en un único fichero con extensión .vue pero manteniendo separados y ordenados los diferentes conceptos, lo que facilita la comprensión y legibilidad de los diferentes componentes.

6. Funcionalidades

Las funcionalidades que tendrá la aplicación web de alimentación sostenible son las siguientes, especificadas en función de los diferentes perfiles de los que dispondrá la aplicación:

Funcionalidad	Cliente	Tienda	Administrador
Registrar nuevo usuario	✓	✗	✗
Obtener la información personal de una cuenta de usuario	✓	✗	✗
Obtener usuario	✓	✗	✓
Editar información de usuario	✓	✗	✗
Registrar nueva tienda	✗	✓	✗
Obtener información de tienda	✓	✓(1)	✓
Obtener lista de tiendas	✓	✓(1)	✓
Buscar tienda	✓	✓(1)	✓
Editar información de tienda	✗	✓	✗
Eliminar tienda	✗	✗	✓
Añadir un producto	✗	✓	✓
Obtener la lista de todos los productos	✓	✓	✓
Obtener producto	✓	✓	✓
Buscar producto	✓	✓	✓
Editar producto	✗	✗	✓
Eliminar producto	✗	✗	✓
Añadir una oferta	✗	✓	✗
Añadir una lista de ofertas	✗	✓	✗

Obtener la lista de todas las ofertas	✓	✗	✓
Obtener las ofertas de una tienda	✓	✓(2)	✓
Obtener una oferta	✓	✓(2)	✓
Buscar oferta	✓	✓(2)	✓
Editar una oferta	✗	✓(2)	✗
Eliminar una oferta	✗	✓	✓
Añadir una oferta a favoritos	✓	✗	✗
Eliminar una oferta de favoritos	✓	✗	✗
Obtener la lista de ofertas favoritas	✓	✗	✗
Añadir una cadena de tiendas	✗	✗	✓
Obtener una cadena de tiendas	✓	✓(3)	✓
Obtener la lista de todas las cadenas de tiendas	✓	✓(3)	✓
Obtener las tiendas pertenecientes a una cadena de tiendas	✓	✓(3)	✓
Editar cadena de tiendas	✗	✗	✓
Eliminar cadena de tiendas	✗	✗	✓
Añadir notificación de un producto	✓	✗	✗
Obtener lista de notificaciones	✓	✗	✗
Eliminar notificación	✓	✗	✗

Tabla 1: Funcionalidades ofrecidas por la aplicación en función del perfil

(1) Para tiendas pertenecientes a la misma cadena de tiendas
(2) Sólo para las ofertas que hayan sido dadas de alta por esa tienda.
(3) Para la cadena de tiendas a la que pertenece el comercio.

7. Contenidos

A continuación se presenta el modelo de datos utilizado en la aplicación web Alimentación sostenible objeto de este proyecto.

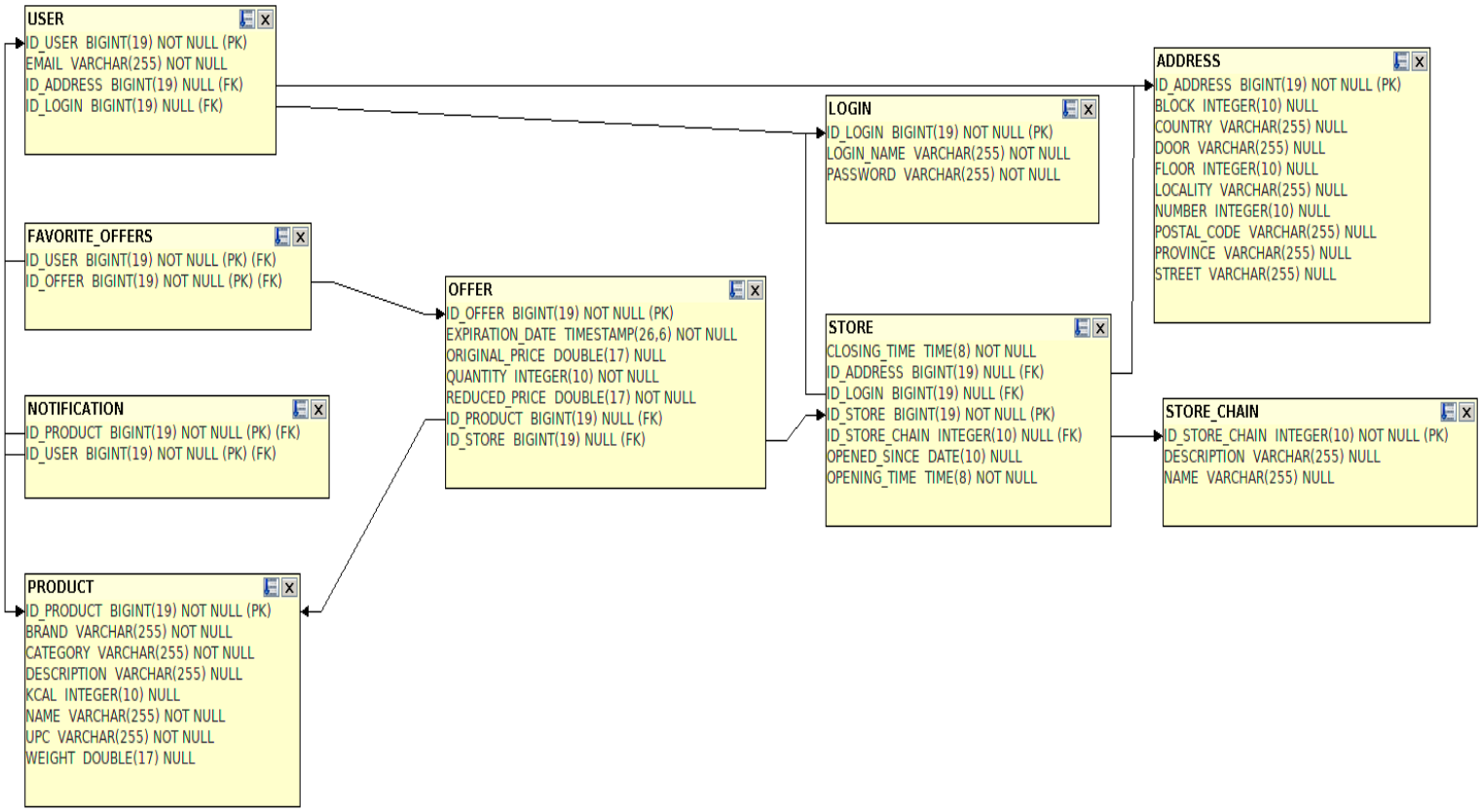


Figura 3: Modelo de datos de la aplicación

En la figura anterior pueden observarse las entidades que forman parte de la aplicación objeto de este proyecto, cada una de las cuales se detalla a continuación para una mayor claridad:

7.1 User

La entidad User representa a los usuarios consumidores que se darán de alta en la aplicación. Un usuario consumidor, podrá gestionar su información de perfil, podrá ver y buscar productos en la aplicación, así como ver y buscar las ofertas publicadas por los diferentes comercios. Además, los usuarios podrán guardar productos como favoritos, de forma que recibirán una notificación en la aplicación cuando algún comercio publique alguna oferta sobre ellos, y guardar ofertas en su lista de favoritos para poder consultarlas cuando deseen.

7.2 Store

La entidad Store representa a los comercios que se darán de alta en la aplicación. Un comercio podrá gestionar su información de perfil, así como ver, buscar y dar de alta productos en la aplicación, además de añadir, editar y eliminar ofertas sobre los mismos que se encuentren disponibles en su tienda.

7.3 Address

La entidad Address representa las direcciones tanto de los perfiles de las cuentas de usuario consumidor como de las cuentas de comercio.

7.4 Store Chain

La entidad Store Chain representa la cadena de tiendas a la que pertenece un comercio en caso de que pertenezca a una. Por ejemplo, Mercadona, Carrefour, Alcampo, etc.

7.5 Login

La entidad Login representa los datos de acceso (usuario y password) y el rol (User, Store, Admin) de la cuenta de acceso a la aplicación.

7.6 Product

La entidad Product representa un producto dado de alta en la aplicación.
Un producto entendido dentro del contexto de la aplicación de alimentación sostenible es un artículo comestible existente en el mercado, con independencia a los establecimientos donde puede ser adquirido. Está identificado por un código único dentro de la aplicación, además de un código upc que lo identifica de forma única en el mercado. Además cuenta con información adicional como: su nombre, su descripción, su marca, su categoría, las kilocalorías por 100 gramos y su peso. Ejemplos de producto serían: Caldo de pollo Gallina Blanca, helado almendrado Hacendado tarrina de 1L, etc

7.7 Offer

La entidad Offer representa una oferta sobre un producto que da de alta un comercio en la aplicación.
Una oferta entendida dentro del contexto de la aplicación de alimentación sostenible es una rebaja que un establecimiento da de alta sobre un producto que tenga en stock y que se encuentre a punto de caducar. La información que define una oferta es el producto sobre el cual se realiza, el comercio que pone el artículo en oferta, la fecha de caducidad, el precio original, el precio rebajado y la cantidad de productos ofertados por el comercio.

7.8 Favorite Offer

La entidad Favorite Offer representa una oferta marcada como favorita por un usuario. Cada usuario puede tener tantas ofertas favoritas como desee.

7.9 Notification

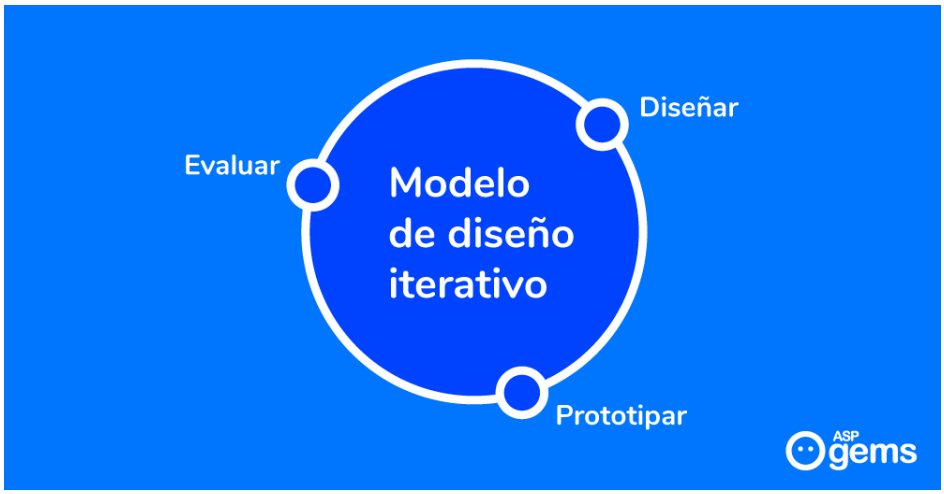
La entidad Notification representa un producto de que un usuario desea recibir notificaciones en el caso de que se añadan ofertas sobre el mismo.

8. Metodología

Para el desarrollo del presente TFM, y teniendo en cuenta que el equipo de trabajo ha estado formado únicamente por una persona, la autora del mismo, no se han utilizado metodologías de desarrollo basadas en la optimización del trabajo en equipo (como por ejemplo, SCRUM).

Por ello, se ha realizado el desarrollo del proyecto utilizando la metodología de desarrollo del software iterativa, precursora de las nuevas metodologías ágiles, pues se ha visto más conveniente para el trabajo unipersonal.

La metodología de desarrollo del software iterativa, consiste en identificar un conjunto de tareas que realizar en cada iteración de forma que en su conjunto aporten funcionalidad al sistema, y en consecuencia valor al cliente, y para llevarlas a cabo, se realicen las fases de análisis, diseño, implementación, prototipado y pruebas. Y repetir estas iteraciones hasta que el producto se dé por finalizado.



[3] Figura 4: Metodología de desarrollo del software iterativa.

Las ventajas de esta metodología es que permite proporcionar versiones funcionales de la aplicación de forma muy temprana, lo cual es recomendable para que el cliente o los stakeholders puedan ir viendo desde las primeras fases de desarrollo el progreso de la aplicación, y poder ir adaptándolo a sus necesidades desde las primeras versiones, evitando tener que hacer modificaciones, más costosas y con más riesgos, sobre las versiones finales (como ocurre en otras metodologías de diseño del software, por ejemplo el desarrollo en Cascada o Waterfall). Además, como en cada iteración se acometen únicamente un subconjunto de las tareas, se reduce la complejidad y se facilita el desarrollo por el paradigma de “Divide y vencerás”, reduciendo además el riesgo y el impacto que un error en una iteración tiene sobre el producto final.

9. Arquitectura de la aplicación/sistema/servicio

Diagrama general de la Arquitectura de la aplicación

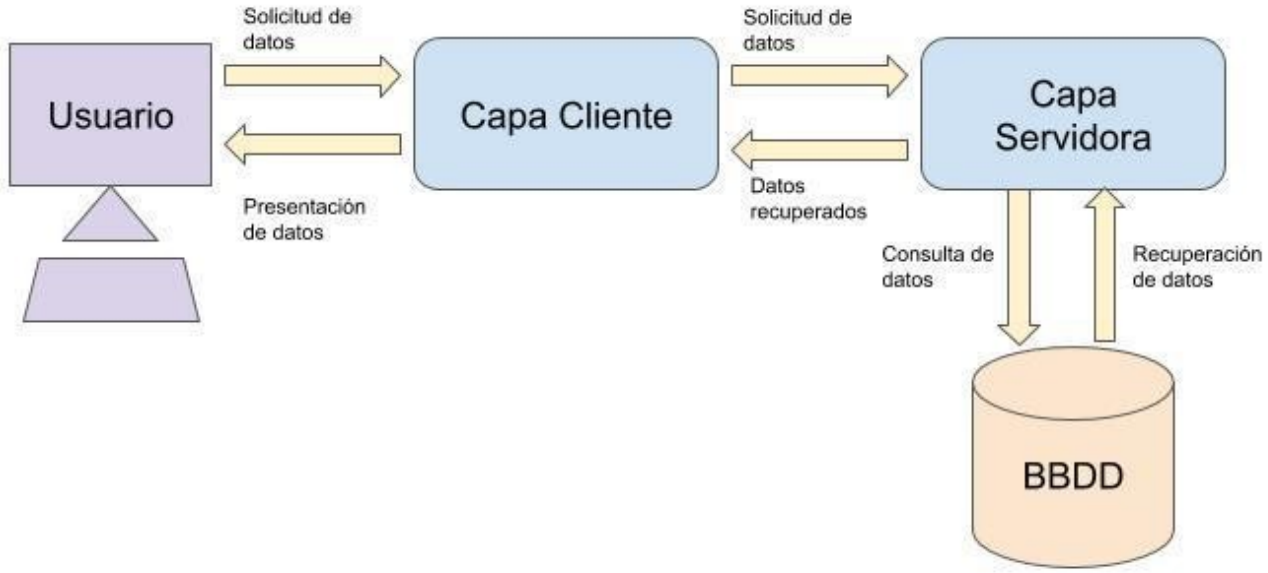


Figura 5: Diagrama general de la Arquitectura de la aplicación.

La arquitectura general de la aplicación se basa en un modelo cliente-servidor, en el cual, se independizan la capa de visualización y presentación de la información al usuario (parte cliente), de la capa de negocio (capa servidora) encargada de obtener y calcular la información en base a las peticiones recibidas desde la capa cliente, y una base de datos que contendrá el modelo de datos y los datos en sí mismos de los que va a hacer uso la aplicación.

Diagrama de la capa servidora

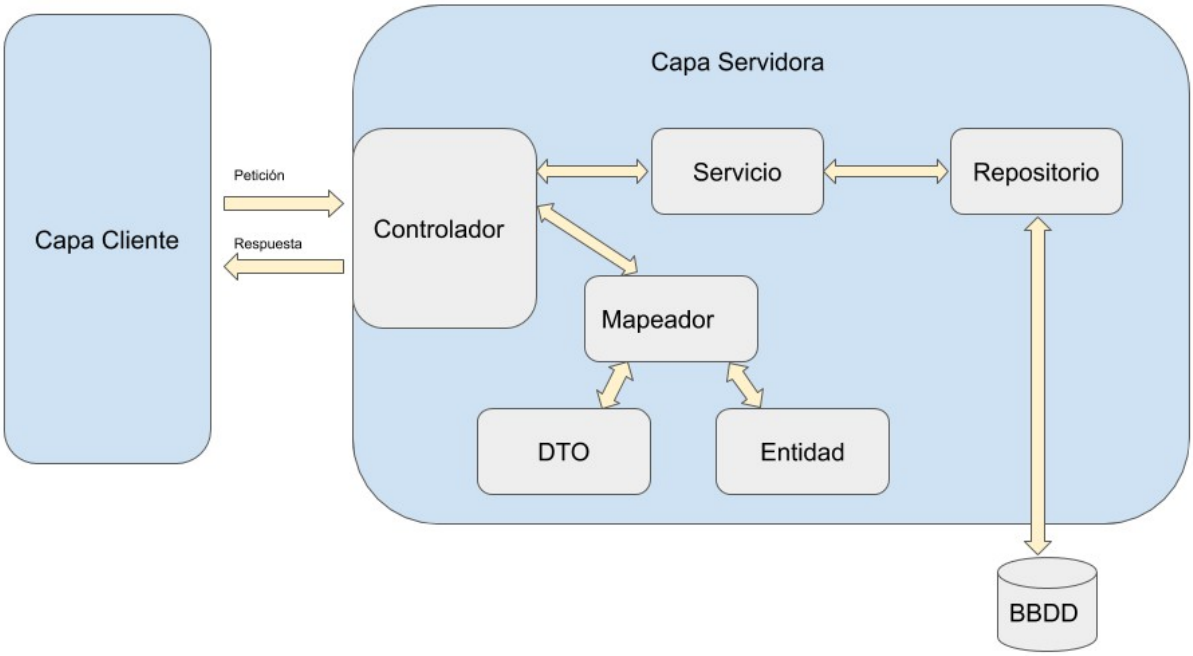


Figura 6: Diagrama de la capa servidora de la aplicación

La arquitectura de la capa servidora de la aplicación se basa en el uso de clases controlador (*Controller*) que serán las que atiendan las peticiones que el BackEnd reciba desde la capa cliente. El controlador, utilizará las clases mapeadoras (*Mapper)* para convertir los objetos DTO con información, recibidos desde el FrontEnd a objetos Entidad, manejados por los servicios y los repositorios JPA; y los servicios (*Service*), a través de los cuales se ejecutarán las diferentes lógicas de negocio necesarias para dotar de funcionalidad a la aplicación, utilizando las clases repositorio (*Repository*) cuando necesiten acceder a la base de datos.

Diagrama de la capa cliente

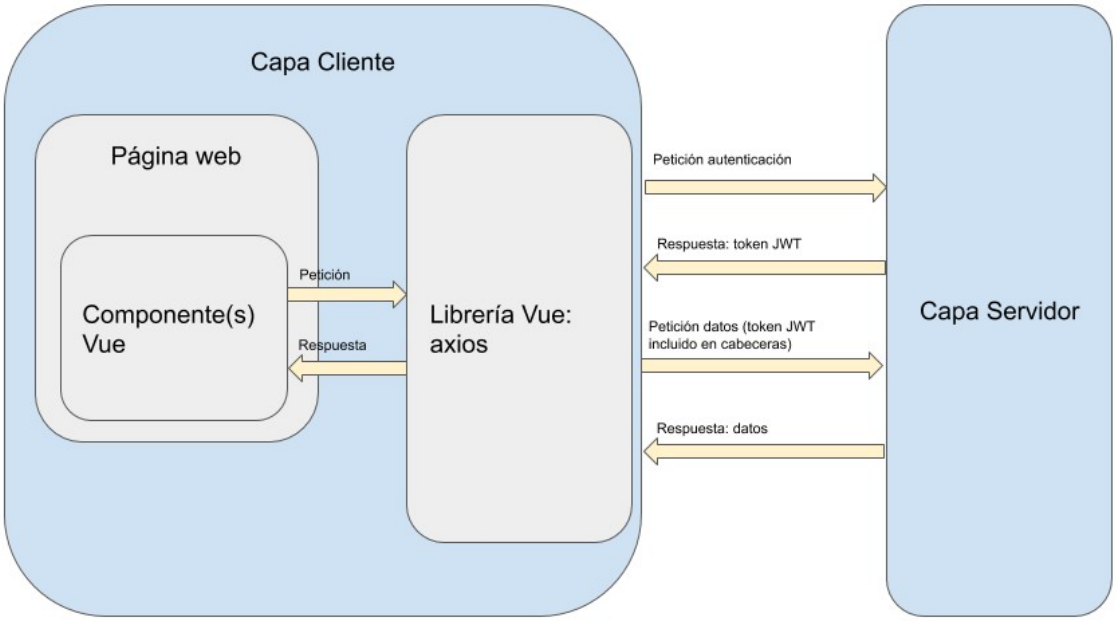


Figura 7: Diagrama de la capa cliente de la aplicación

La arquitectura de la parte cliente de la aplicación, se basa en el uso de componentes VueJS incluidos en las páginas HTML de las que consta la parte visual de la aplicación. A través de tales componentes de VueJS y apoyándose en la librería de VueJS axios, se realizan las diferentes peticiones al API REST, expuesto por la capa servidora de la aplicación, con el fin, tanto de obtener la autenticación y la autorización para la ejecución de las diferentes funcionalidades, como para la obtención de datos que mostrar al cliente.
Los datos obtenidos tras la invocación a las operaciones expuestas por el API REST, son renderizados mediante las funcionalidades proporcionadas por VueJS, como por ejemplo las directivas, el renderizado condicional, el renderizado de listas, etc.

10. Plataforma de desarrollo

Para el desarrollo del presente TFM, se han utilizado los siguientes elementos:

Software	IDE	IntelliJ IDEA Community Edition
	Sistemas Operativos (Dual Boot)	Linux (Ubuntu) 64bits y Windows 10 64bits
	Java	Java 8
	Gestor de proyectos y dependencias	Maven
	Sistema de control de versiones	Git
	Sistema de repositorios	Github
	Visor de Base de Datos	SQuirrel SQL
	Editor de diagramas	Umbrello
	Software de grabación de video	Camtasia
Hardware	Ordenador portátil: Memoria	16GiB
	Ordenador portátil: Procesador	Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
	Ordenador portátil: Gráficos	Intel® UHD Graphics 620 (Kabylake GT2)
	Ordenador portátil: Disco	249,9 GB

Tabla 2: Plataforma de desarrollo

11. Planificación

Para la planificación del proyecto a llevar a cabo como objeto de este TFM, se plantean los siguientes hitos y fechas clave mostrados a través del diagrama de Gantt que se muestra a continuación:

TAREAS	INICIO	FINALIZACIÓN	DÍAS
PEC 1	20-Feb	3-Mar	13
Introducción y planteamiento del proyecto	20-Feb	22-Feb	3
Marco teórico y justificación de decisiones de diseño	22-Feb	26-Feb	5
Objetivos del proyecto	27-Feb	29-Feb	3
Planificación	29-Feb	1-Mar	2
Preparación de entorno de desarrollo	1-Mar	3-Mar	3
PEC 2	4-Mar	1-Apr	29
Revisión PEC 1	4-Mar	9-Mar	6
Implementación de la base de datos	9-Mar	12-Mar	4
Descripción de la Arquitectura	12-Mar	15-Mar	4
Diagramas UML (casos de uso y clases)	15-Mar	19-Mar	5
Usabilidad y experiencia de usuario UX	19-Mar	25-Mar	7
Desarrollo de servicios REST	12-Mar	29-Mar	18
Documentación de API con Swagger	29-Mar	1-Apr	4
PEC 3	2-Apr	10-May	39
Previsión PEC 2	2-Apr	6-Apr	5
Desarrollo Front-End	6-Apr	22-Apr	17
Implementación de seguridad	22-Apr	27-Apr	6
Pruebas	27-Apr	2-May	6
Documentación y memoria	2-May	10-May	9
PEC 4 - Entrega final	10-May	8-Jun	30
Finalización del proyecto	10-May	18-May	9
Finalización de la memoria	18-May	26-May	9
Presentación del proyecto	26-May	1-Jun	7
Presentación en video	1-Jun	8-Jun	8
Defensa TFM	15-Jun	19-Jun	5

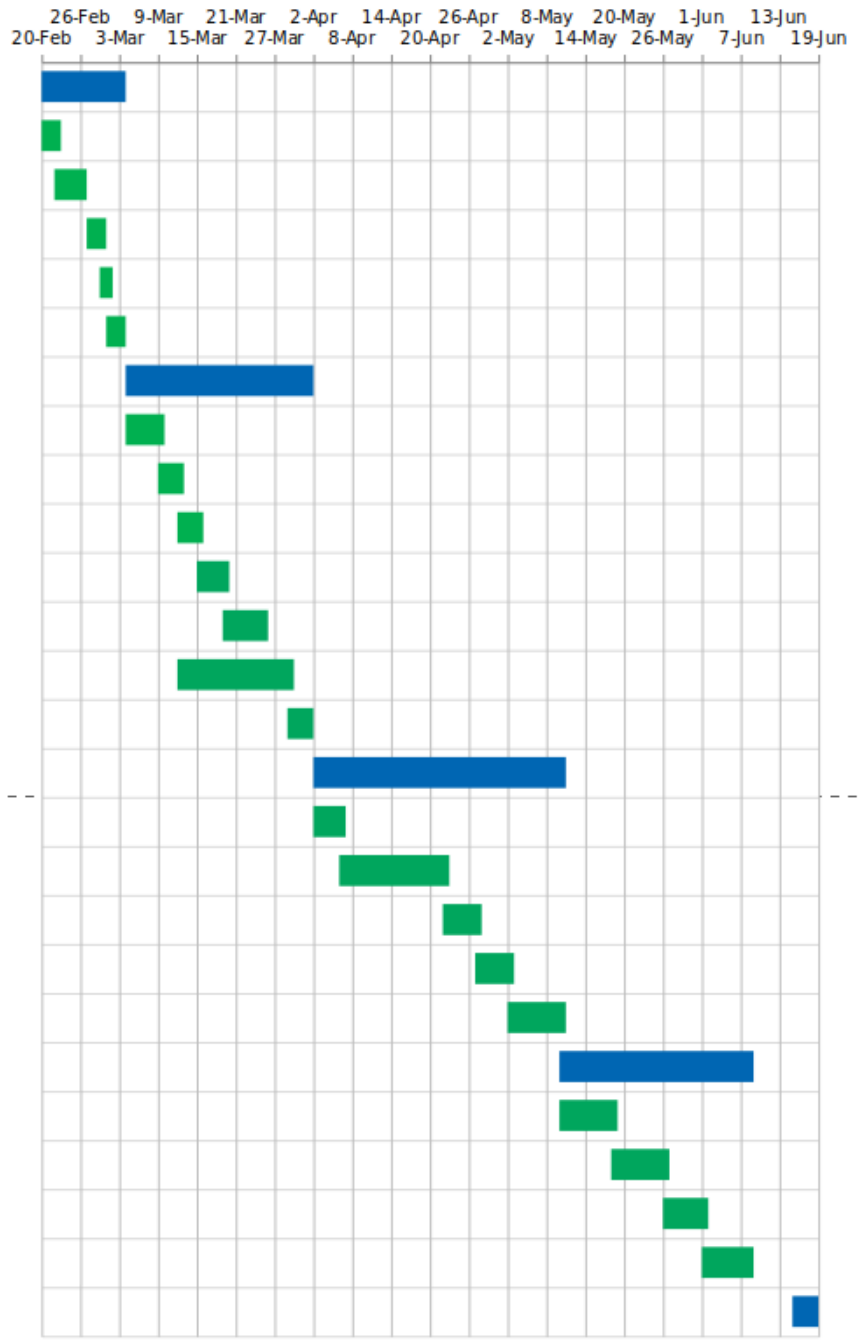


Figura 8: Diagrama de Gantt: planificación del proyecto

12. Proceso de trabajo/desarrollo

Para la realización del presente TFM, se ha dividido el proceso de desarrollo y documentación en fases, de forma que en cada fase se ha ido avanzando iterativamente hacia la finalización del proyecto, ajustándose además el avance de cada fase a cada una de las entregas planificadas y los requerimientos de las mismas.

Así, en la fase 1, el objetivo principal ha sido realizar el planteamiento del proyecto, consensuar los diferentes aspectos funcionales y tecnológicos con el tutor, así como su introducción, de forma que se tuviera claro el objetivo a perseguir con la realización del mismo. Además, para organizar el trabajo, se ha planteado una planificación del proyecto con las diferentes fases en las que se ha dividido el desarrollo y se ha preparado el entorno de desarrollo para poder comenzar con la implementación del producto en fases posteriores.

En la fase 2, se ha comenzado con la realización de cambios y mejoras sobre los aspectos ya desarrollados en la fase anterior. Además, se ha abordado el aterrizaje, el análisis, y el diseño de la propuesta, planteando el modelo de datos, la arquitectura de la aplicación web, los diagramas de casos de uso y los diagramas de clases, así como planteando en forma de prototipo de baja fidelidad la navegación entre las páginas web que conforman el sitio web. Por último, se ha desarrollado la capa servidora, en este caso, en forma de API REST asi como la base de datos en la que se sustentan los datos de la aplicación.

En la fase 3, se ha comenzado con la realización de cambios y mejoras sobre los aspectos ya desarrollados en la fase anterior, se ha implementado la parte cliente de la aplicación (Front End) y los mecanismos de seguridad de los que se ha dotado a la aplicación de alimentación sostenible. Además, se han realizado pruebas de validación de las diferentes funcionalidades.

Por último, en la fase 4, se han finalizado los apartados restantes de la memoria del presente TFM, se ha realizado una presentación y un video con los que realizar la defensa del mismo.

13. APIs utilizadas

Para la realización del presente TFM no se ha utilizado ninguna API de terceros.

Las tecnologías utilizadas son las que se han detallado en el apartado: *5.- Marco Teórico.*

14. Diagramas UML

● 14.1 Diagramas de casos de uso

14.1.1 Casos de uso: Usuario

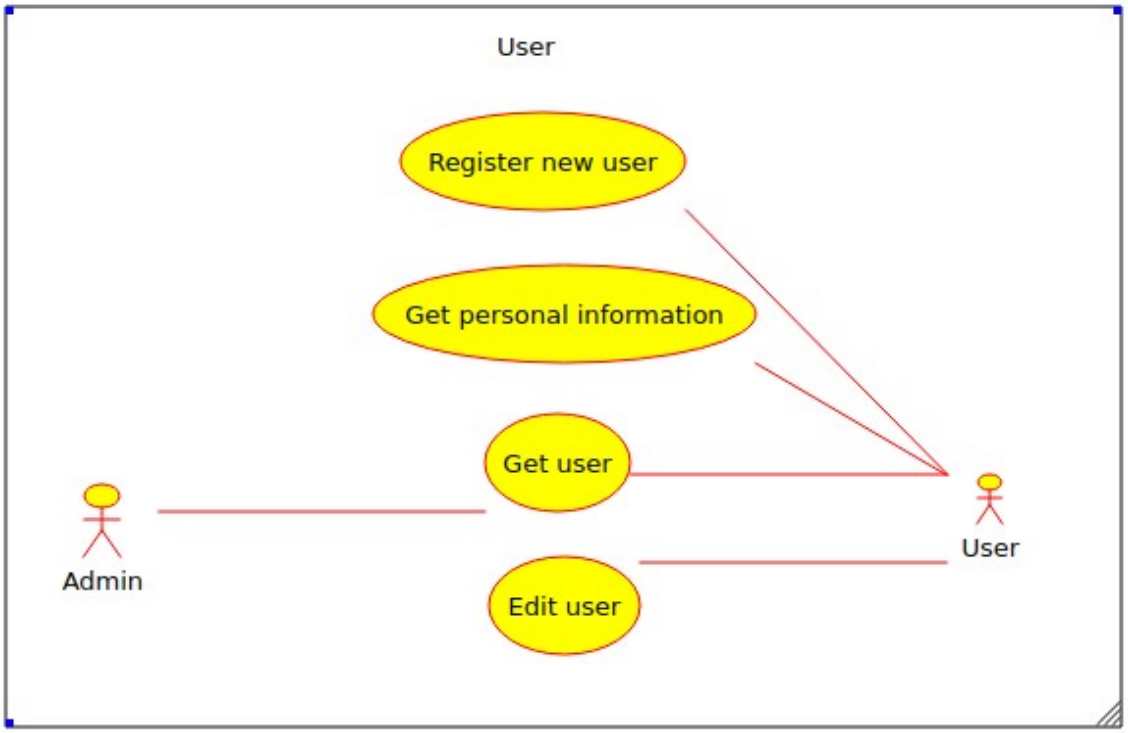


Figura 9: Diagrama de casos de uso: Usuario

14.1.2 Casos de uso: Tienda

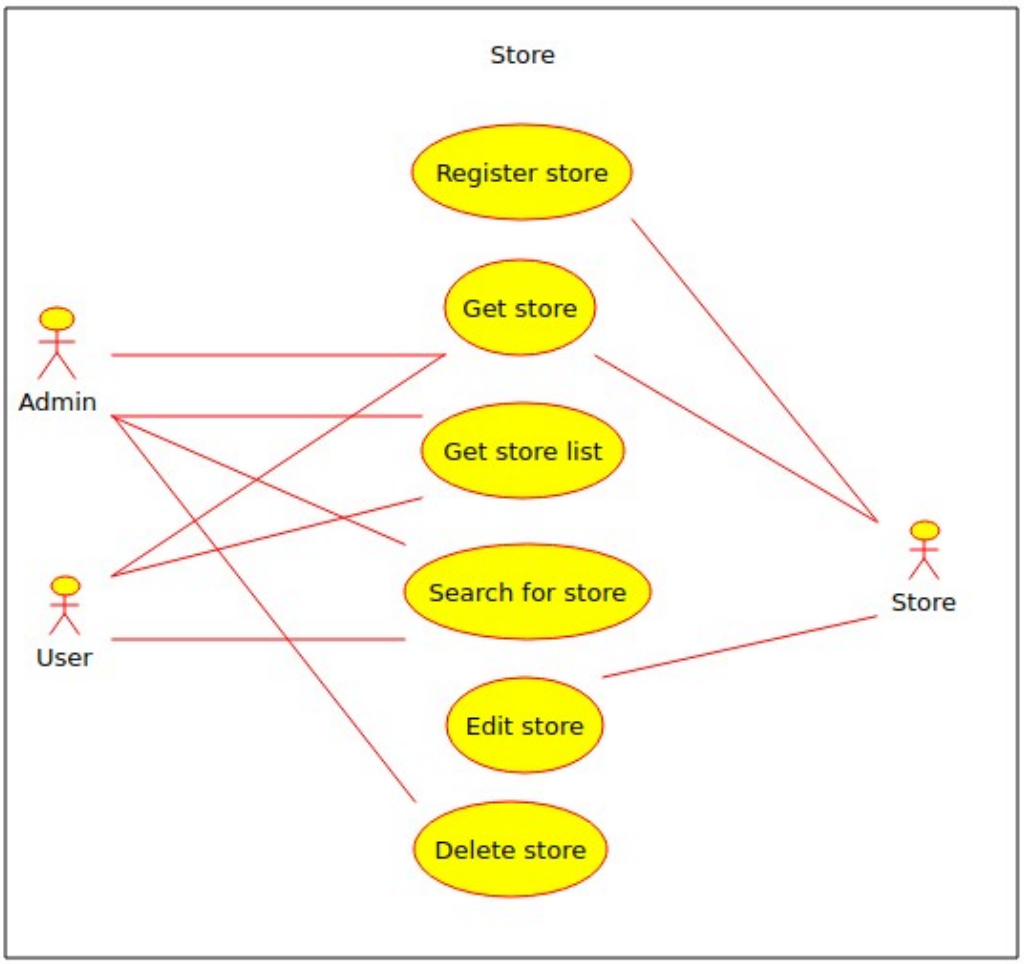


Figura 10: Diagrama de casos de uso: Tienda

14.1.3 Casos de uso: Producto

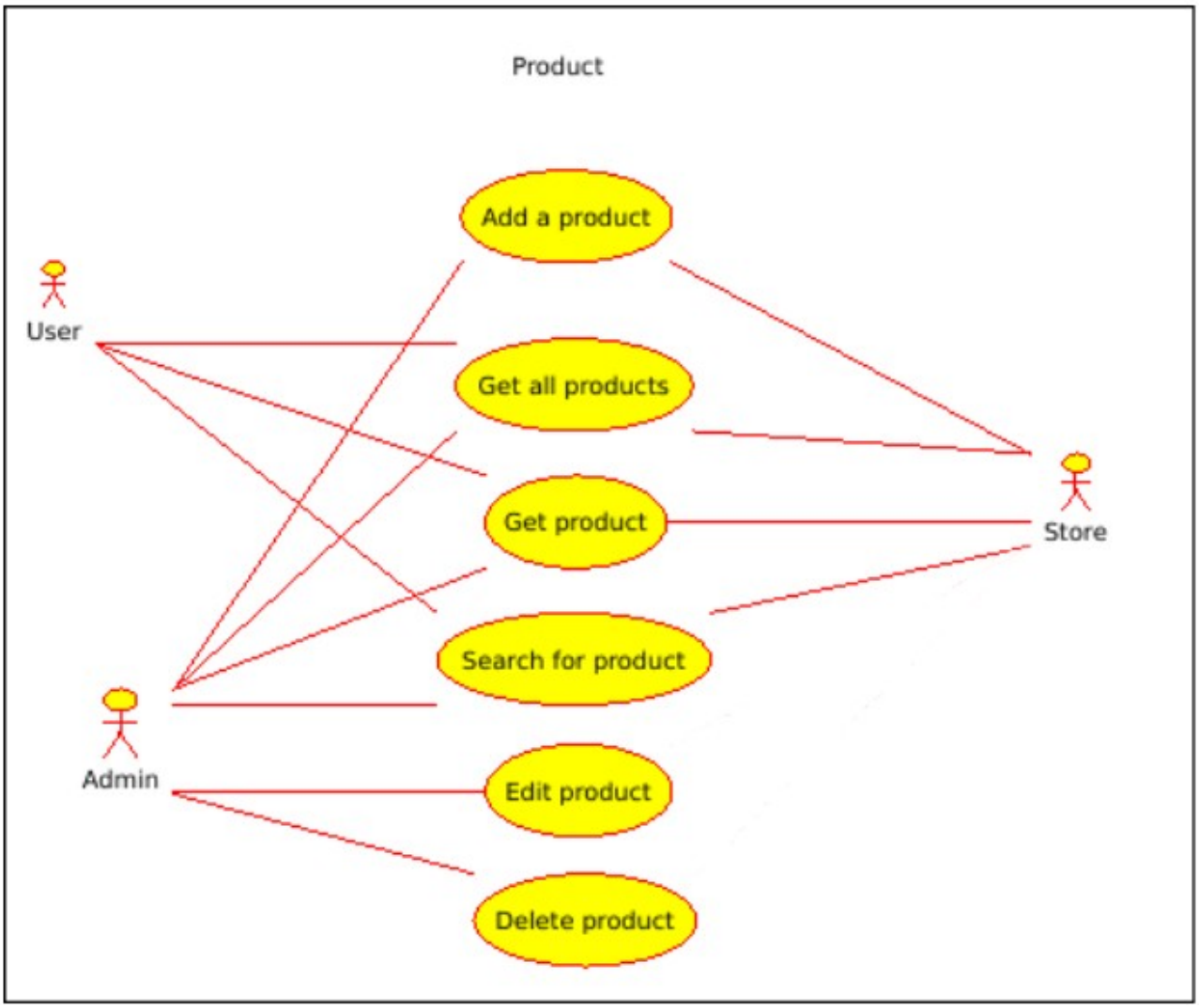


Figura 11: Diagrama de casos de uso: Producto

14.1.4 Casos de uso: Oferta

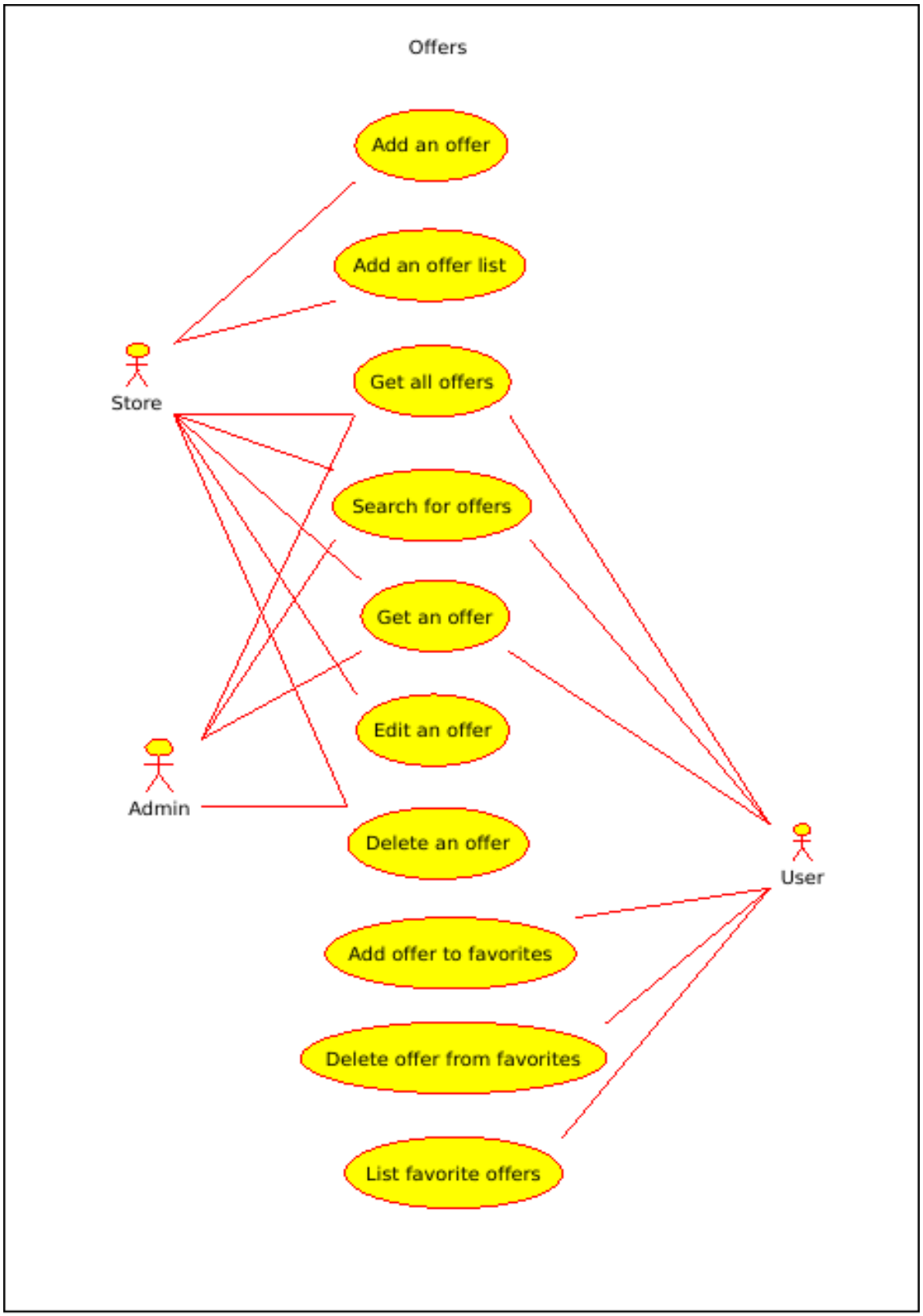


Figura 12: Diagrama de casos de uso: Oferta

14.1.5 Casos de uso: Cadena de tiendas

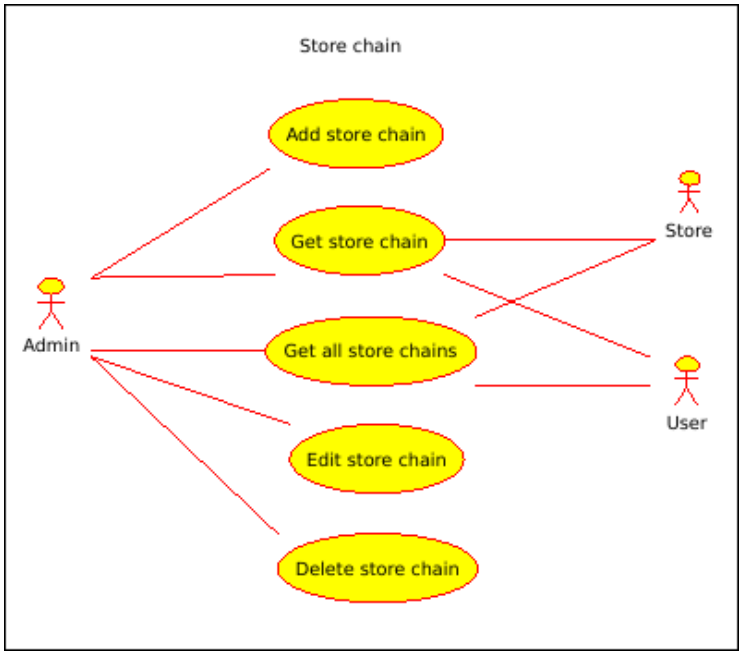


Figura 13: Diagrama de casos de uso: Cadena de tiendas

14.1.6 Casos de uso: Notificaciones

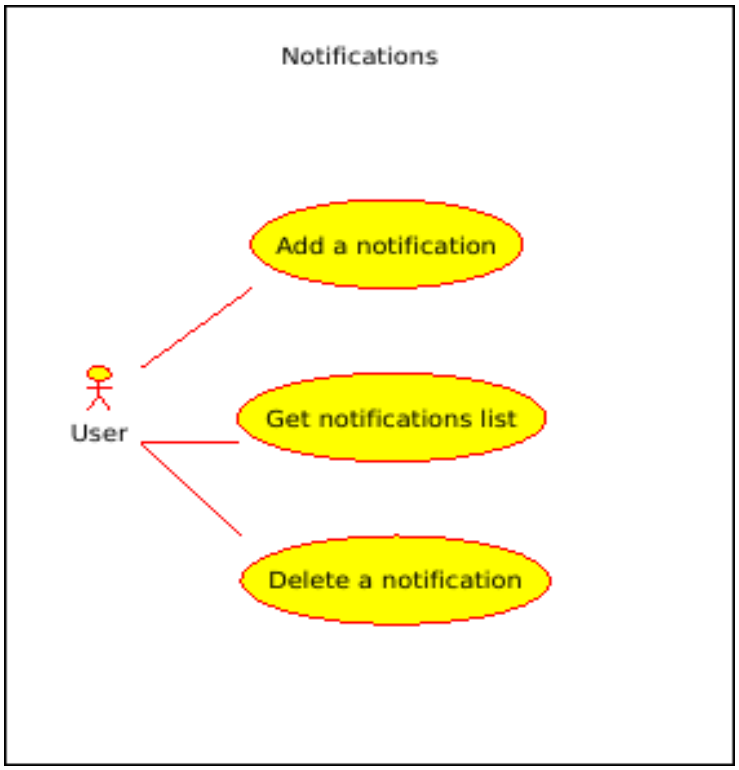


Figura 14: Diagrama de casos de uso: Notificaciones

14.2 Diagramas de clases

14.2.1 Usuario

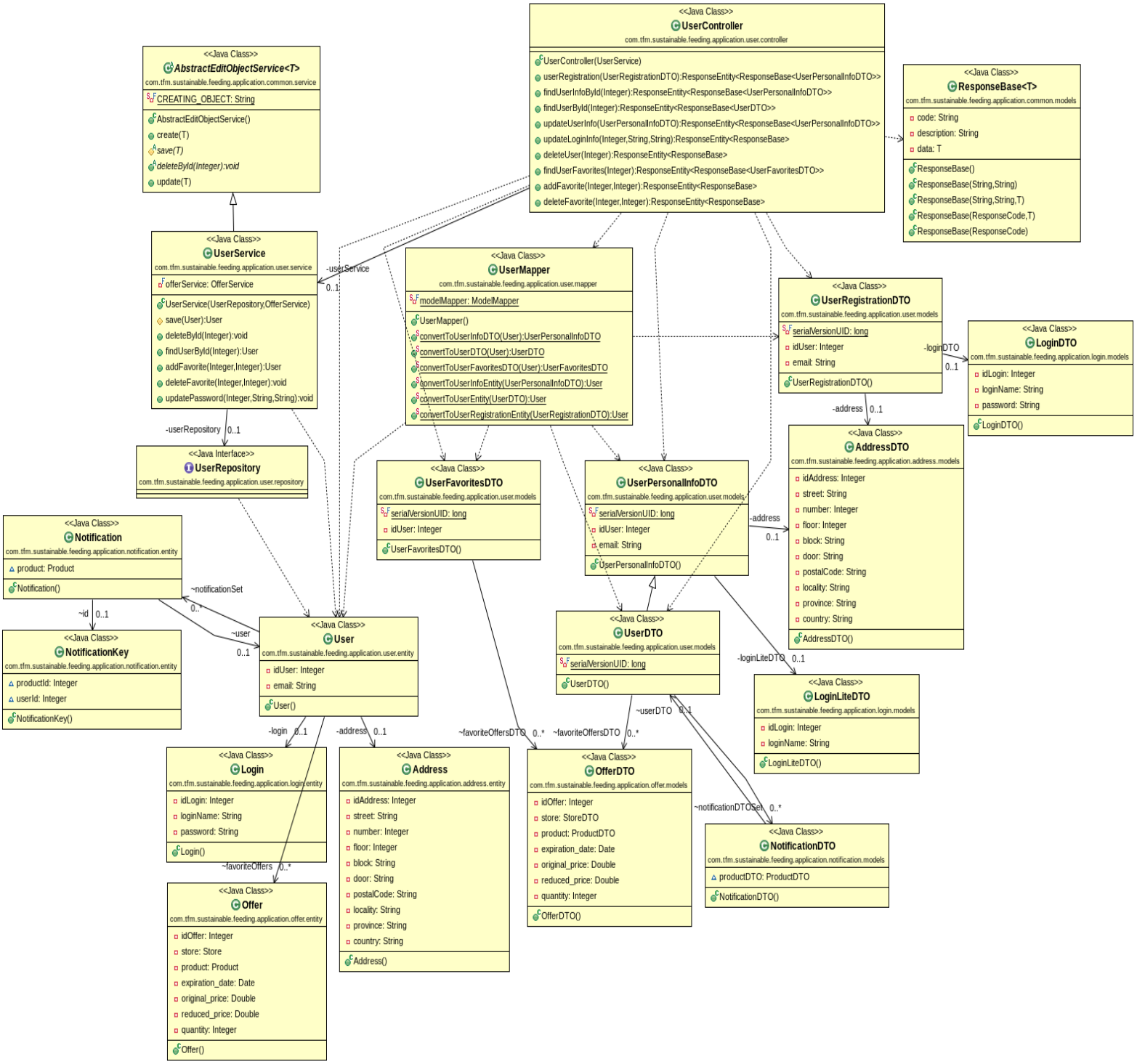


Figura 15: Diagrama de clases: Usuario

14.2.2 Tienda

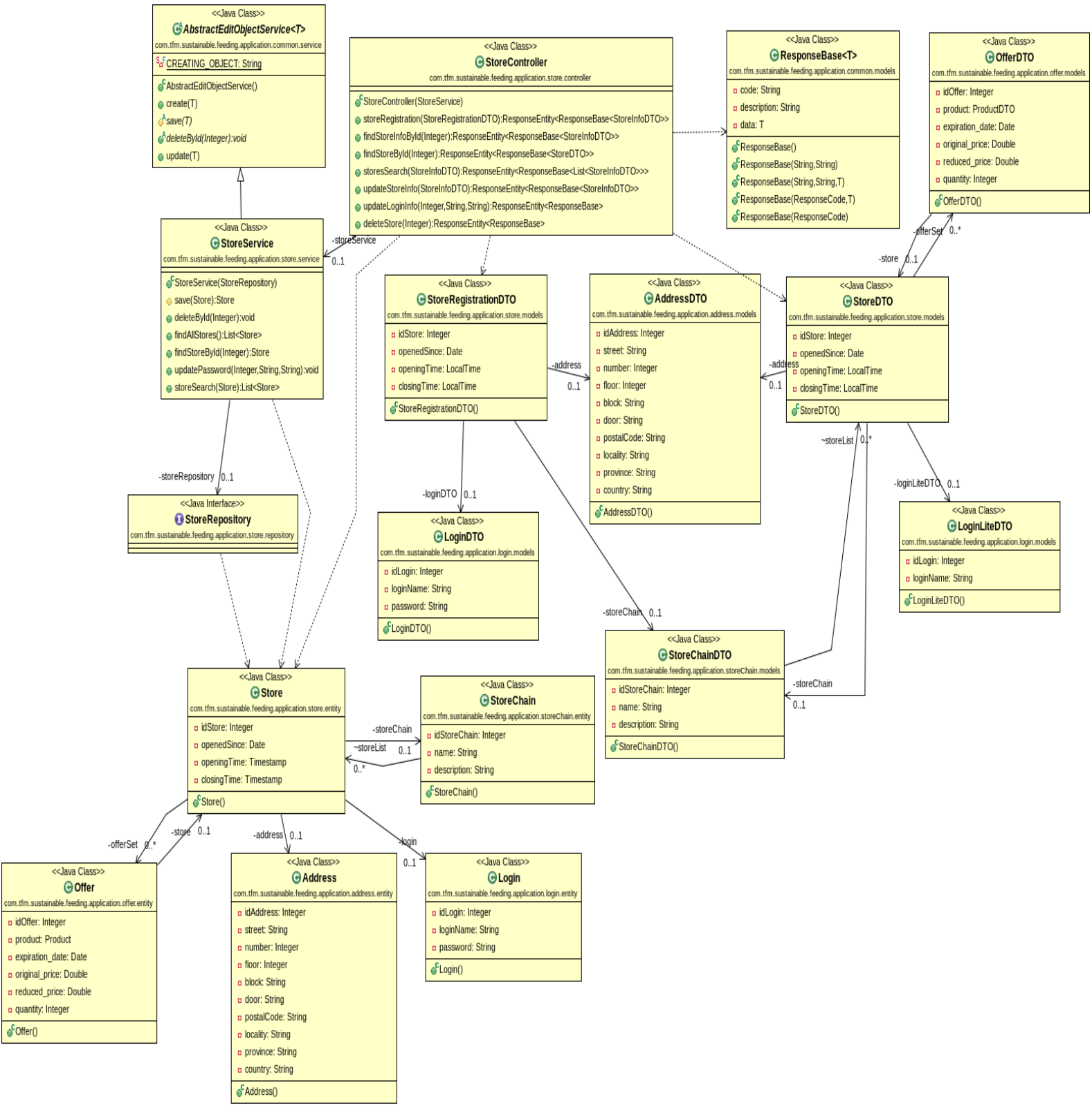


Figura 14: Diagrama de clases: Tienda

14.2.3 Dirección

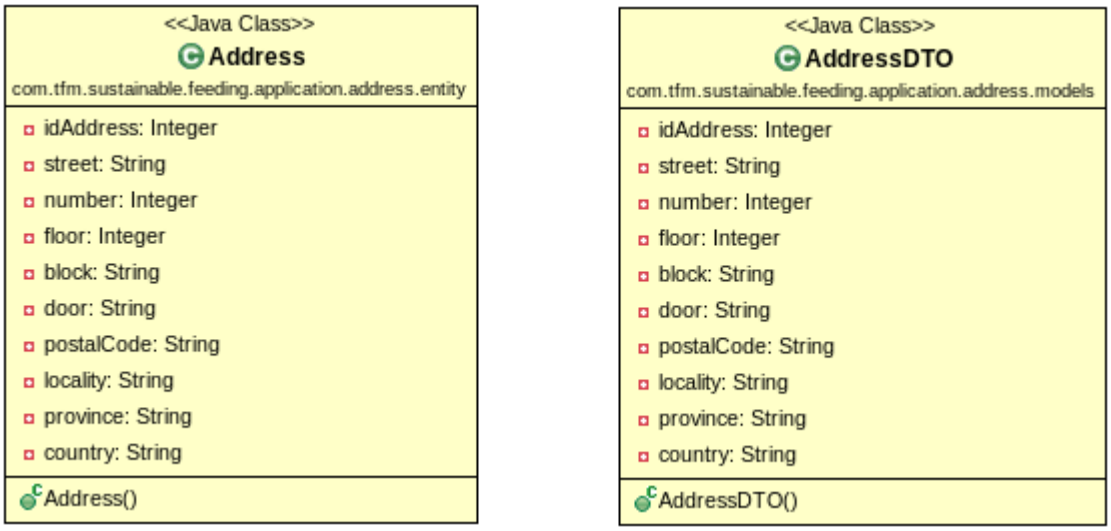


Figura 17: Diagrama de clases: Dirección

14.2.4 Login

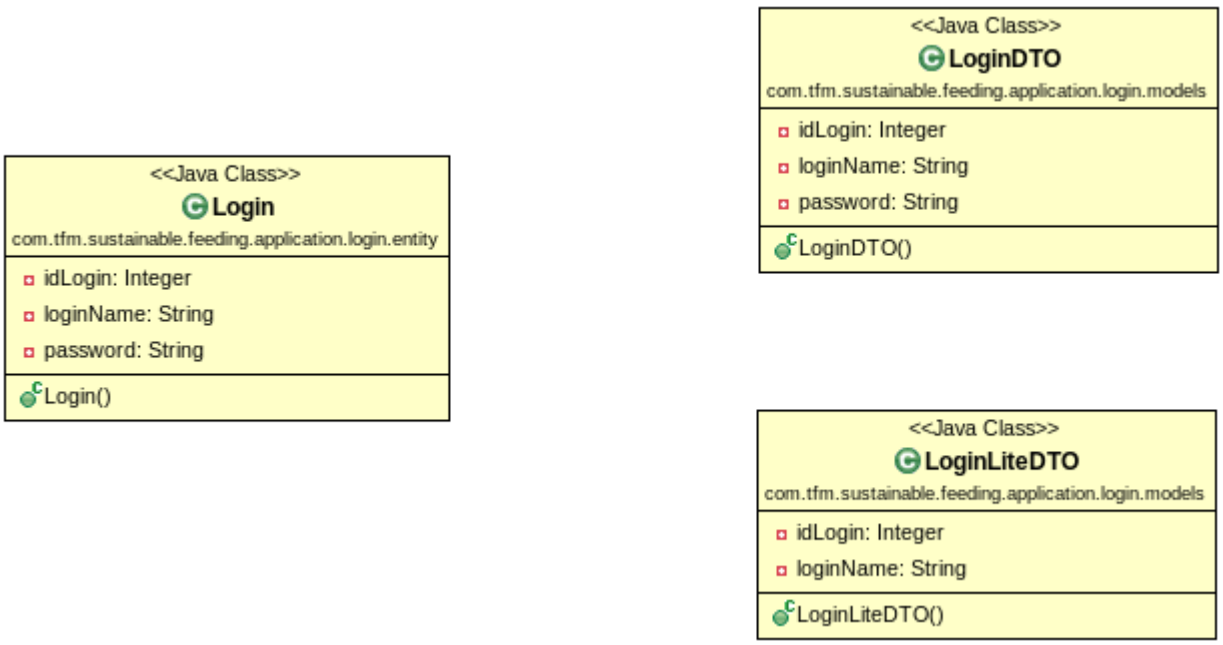


Figura 18: Diagrama de clases: Login

14.2.5 Cadena de tiendas

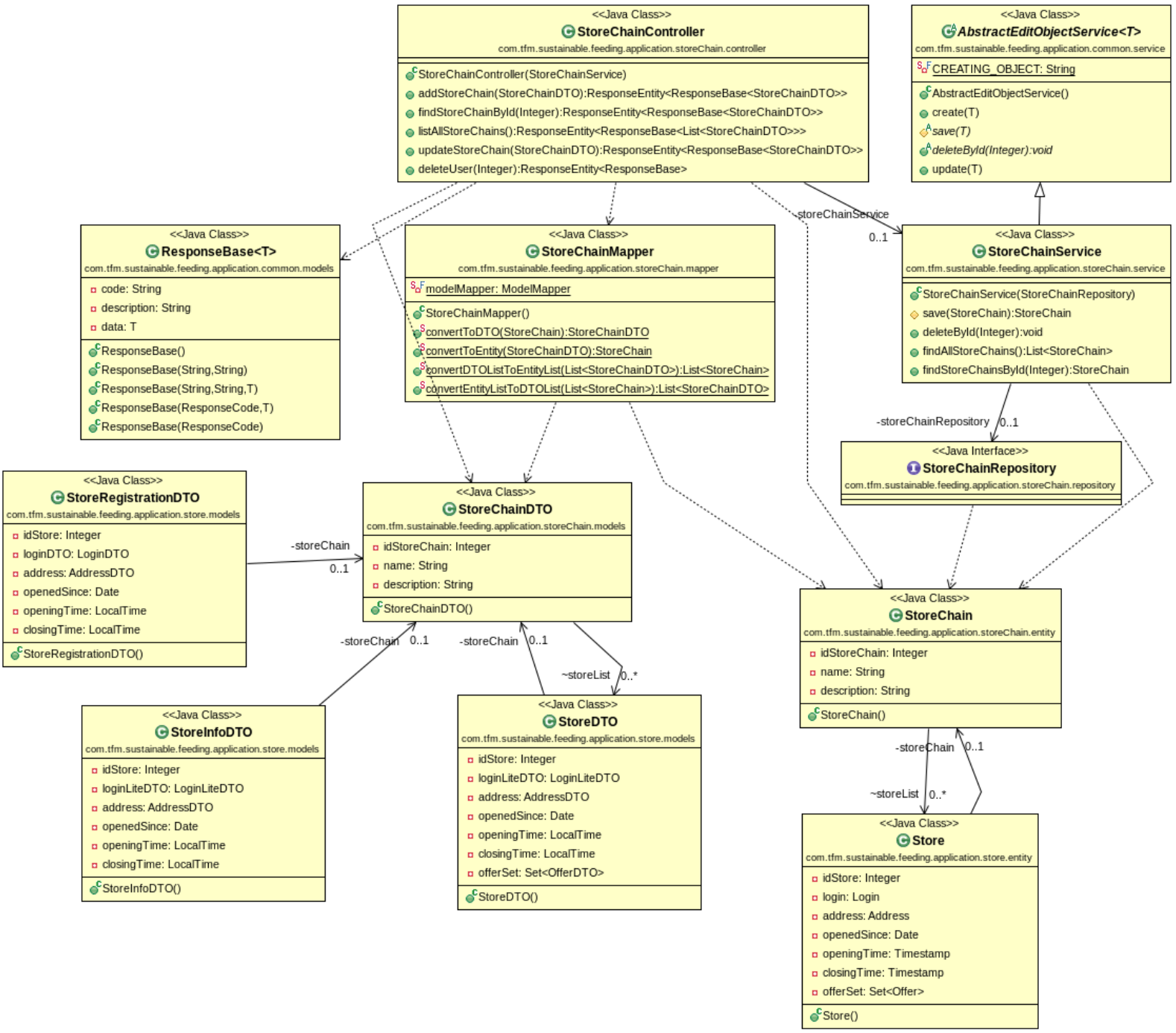


Figura 19: Diagrama de clases: Cadena de tiendas

14.2.6 Producto

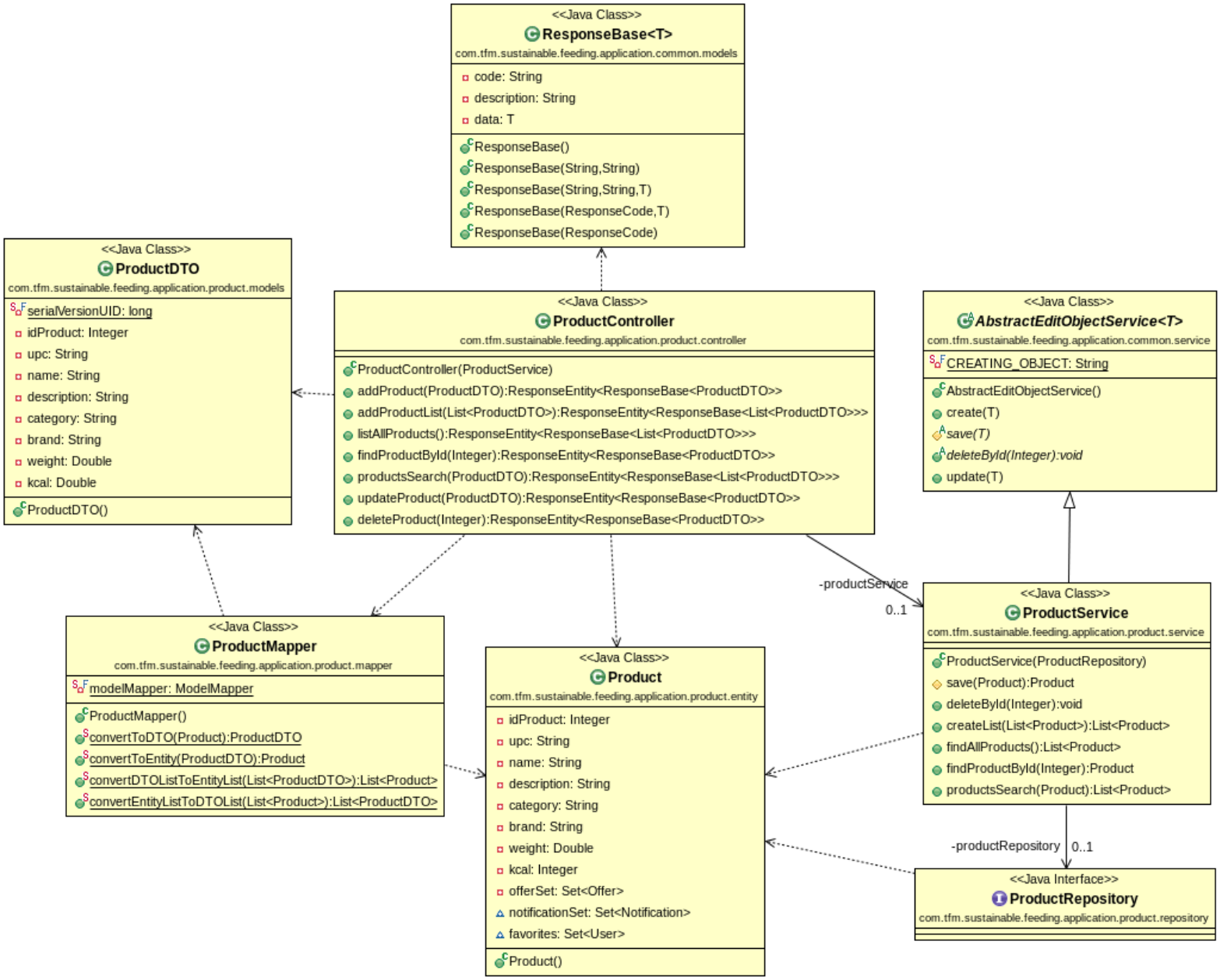


Figura 20: Diagrama de clases: Producto

14.2.7 Oferta

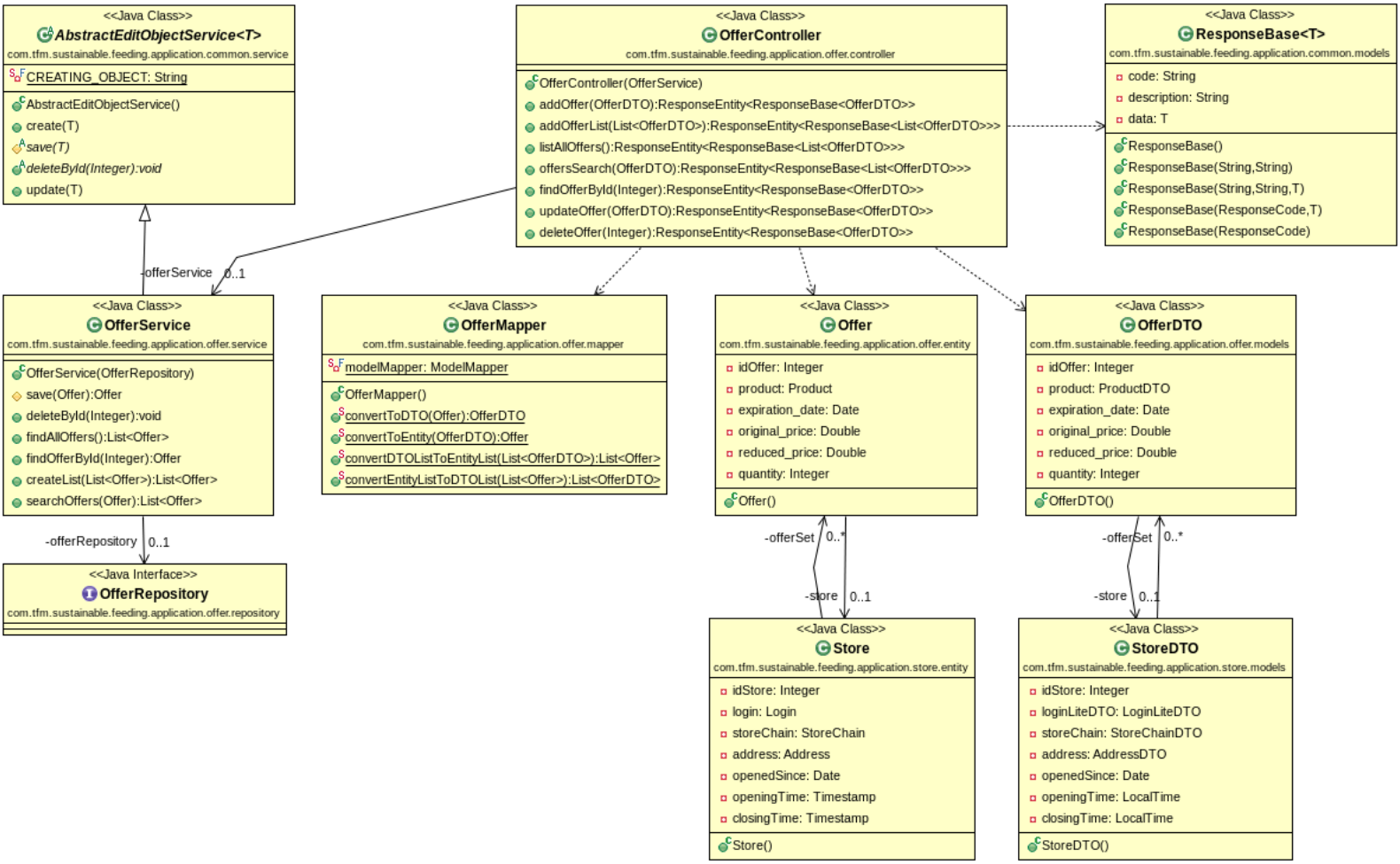


Figura 21: Diagrama de clases: Oferta

14.2.8 Notificaciones

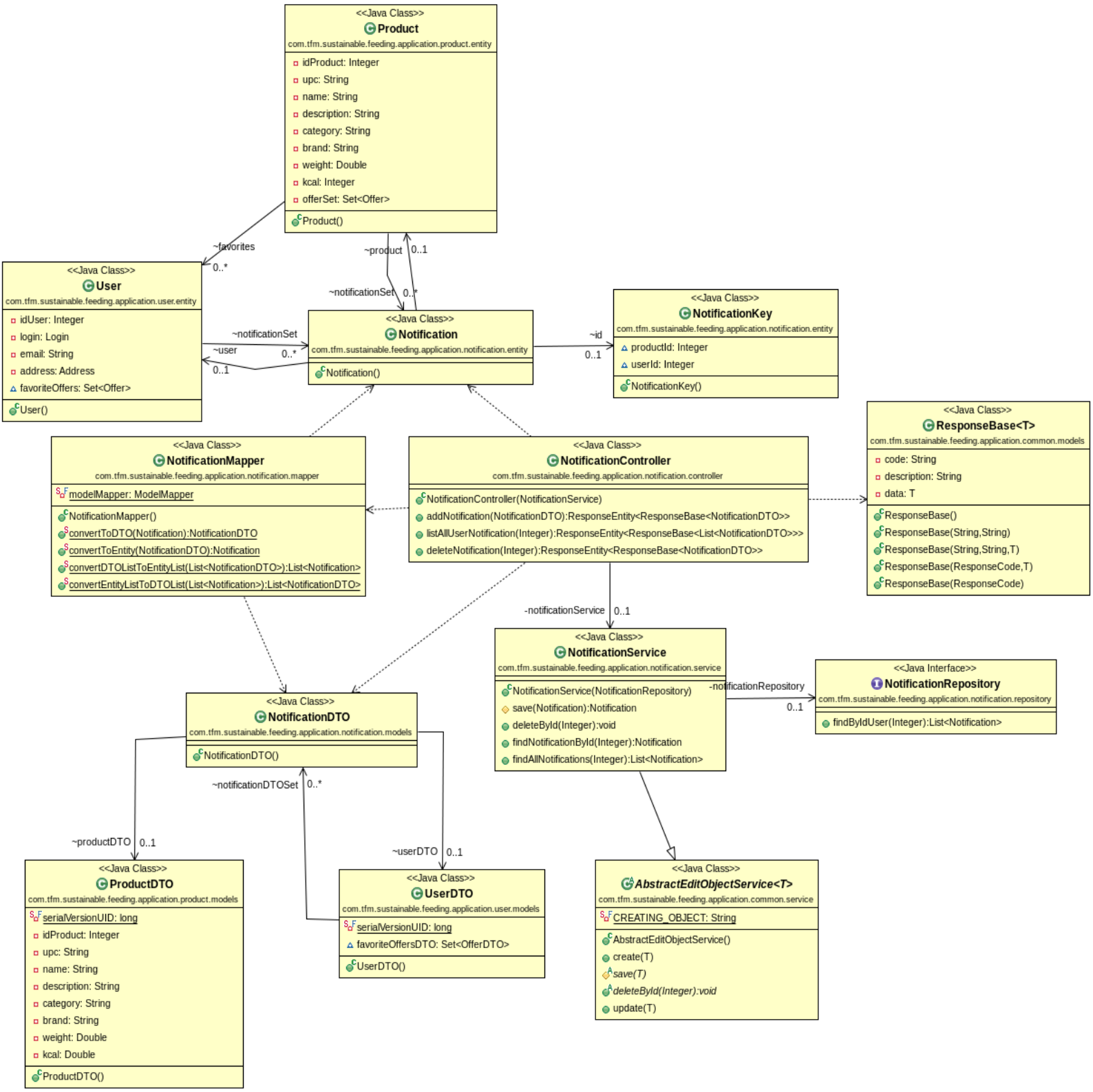


Figura 20: Diagrama de clases: Notificaciones

15. Prototipos

15.1 Lo-Fi

A continuación se presenta el prototipo de baja fidelidad de las diferentes ventanas de las que constará la aplicación de alimentación sostenible:

15.1.1 Página: Inicial

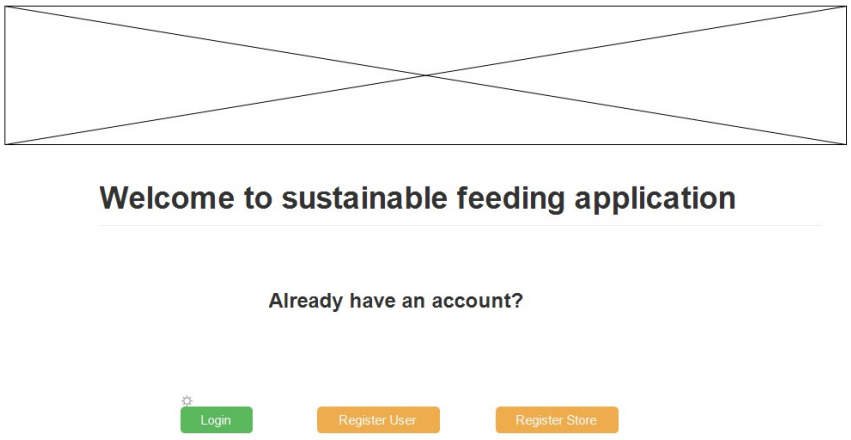


Figura 23: Lo-Fi - Página: Inicial

15.1.2 Página: Login

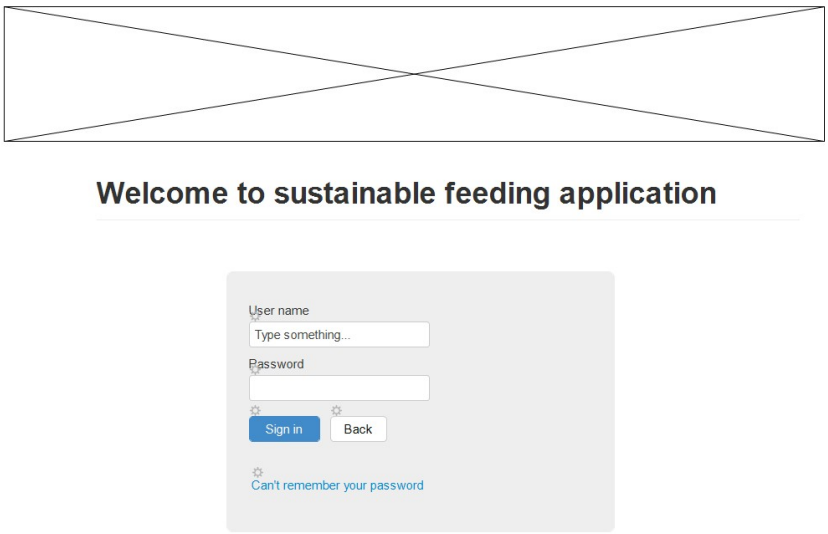


Figura 24: Lo-Fi - Página de login

15.1.3 Página: Registro de usuario

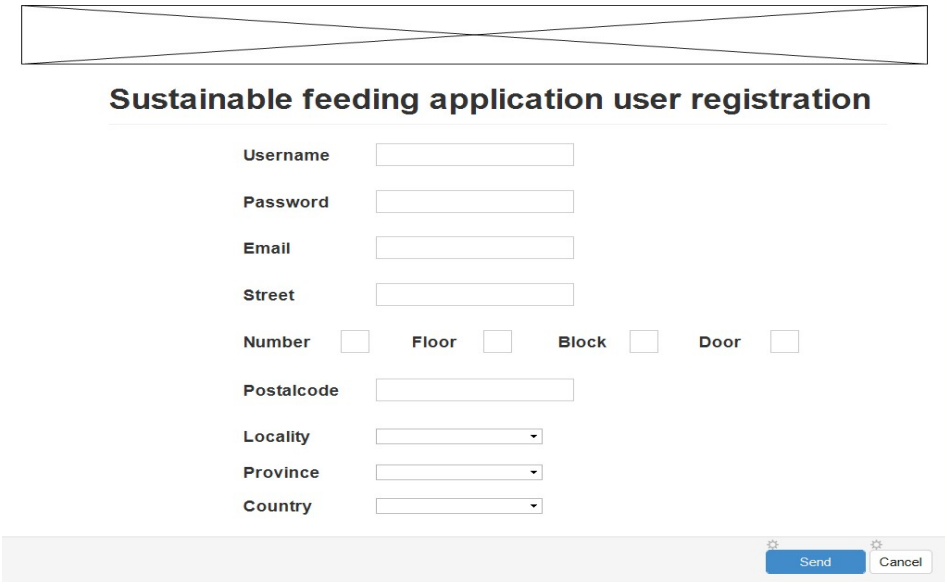


Figura 25: Lo-Fi - Página: Registro de usuario

15.1.4 Página: Registro de tienda

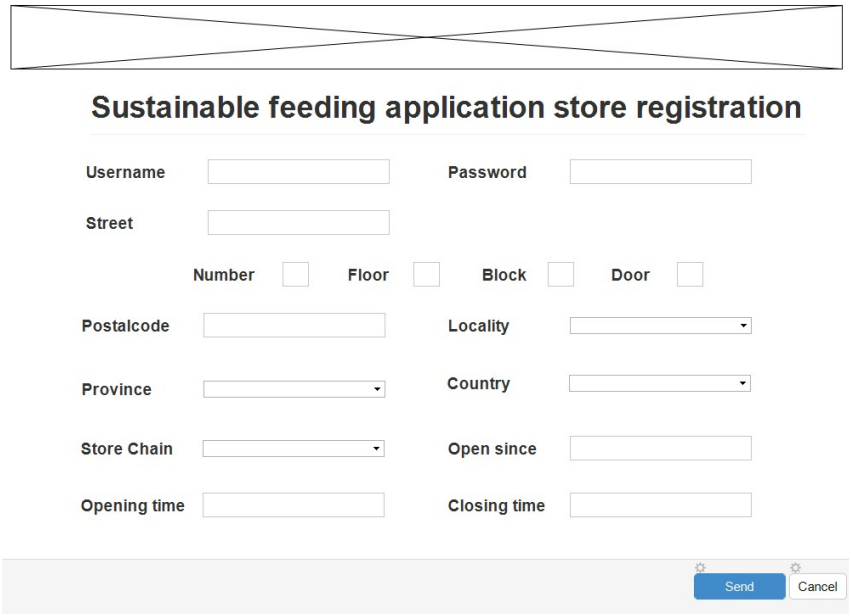


Figura 26: Lo-Fi - Página: Registro de tienda

15.1.5 Página: Home

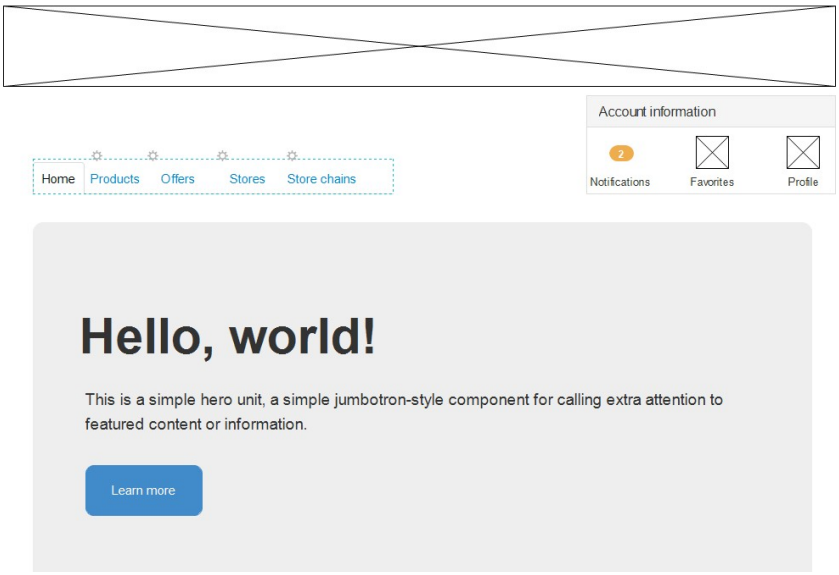


Figura 27: Lo-Fi - Página: Home

15.1.6 Página: Productos

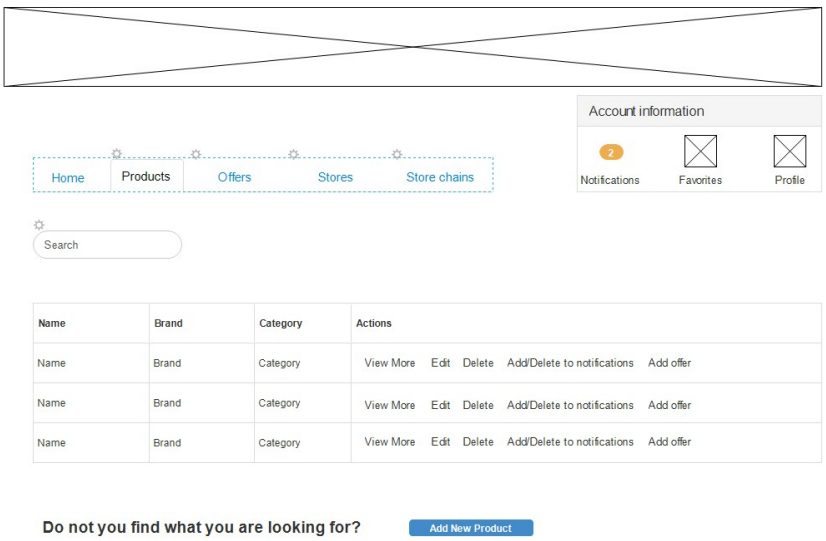


Figura 28: Lo-Fi - Página: Productos

15.1.7 Ventana modal: Añadir Oferta de un producto

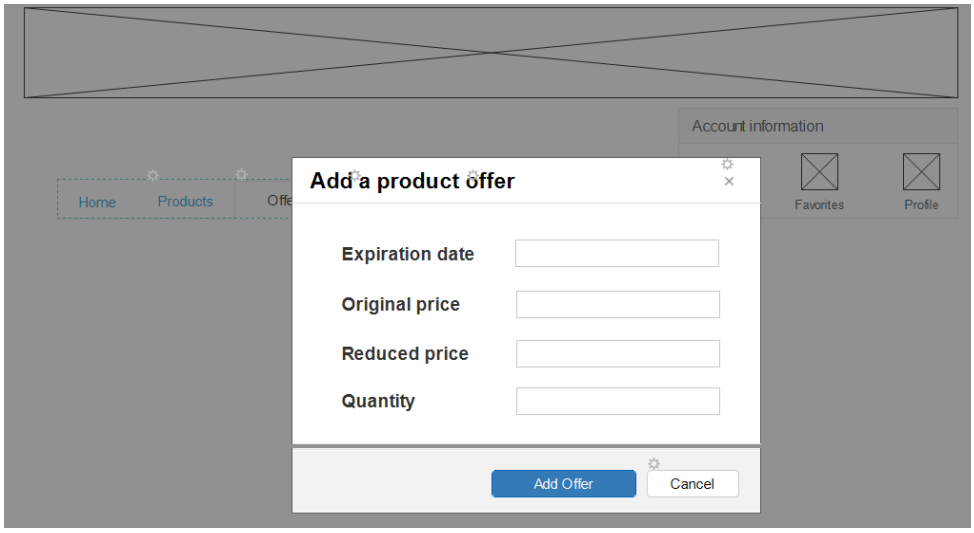


Figura 29: Lo-Fi - Ventana Modal: añadir oferta de producto

15.1.8 Página: Ver producto

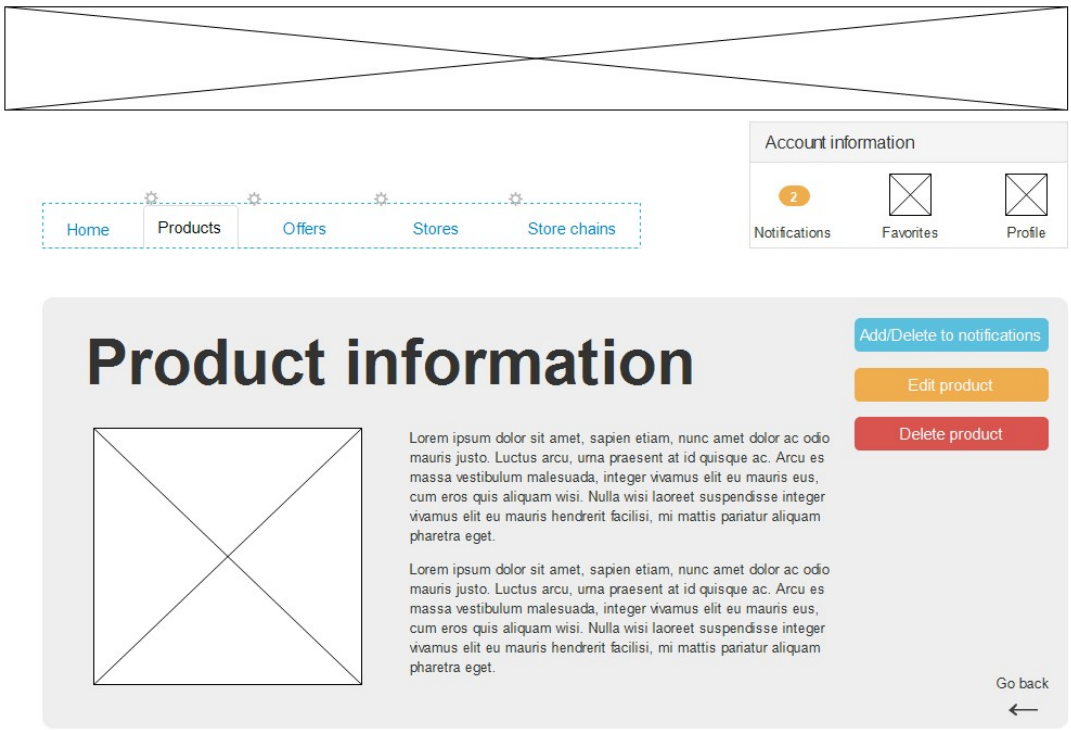


Figura 30: Lo-Fi - Página: Ver producto

15.1.9 Página: Añadir producto

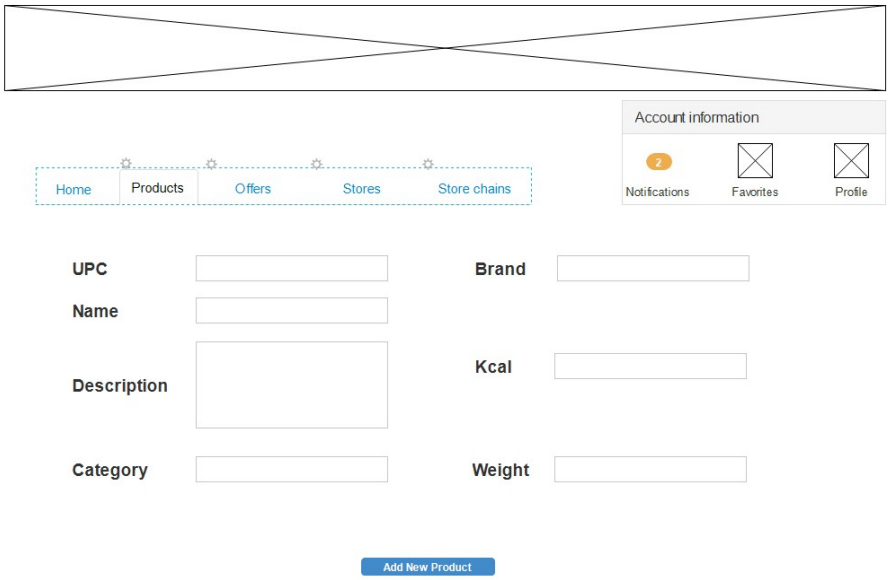


Figura 31: Lo-Fi - Página: Añadir producto

15.1.10 Página: Editar producto

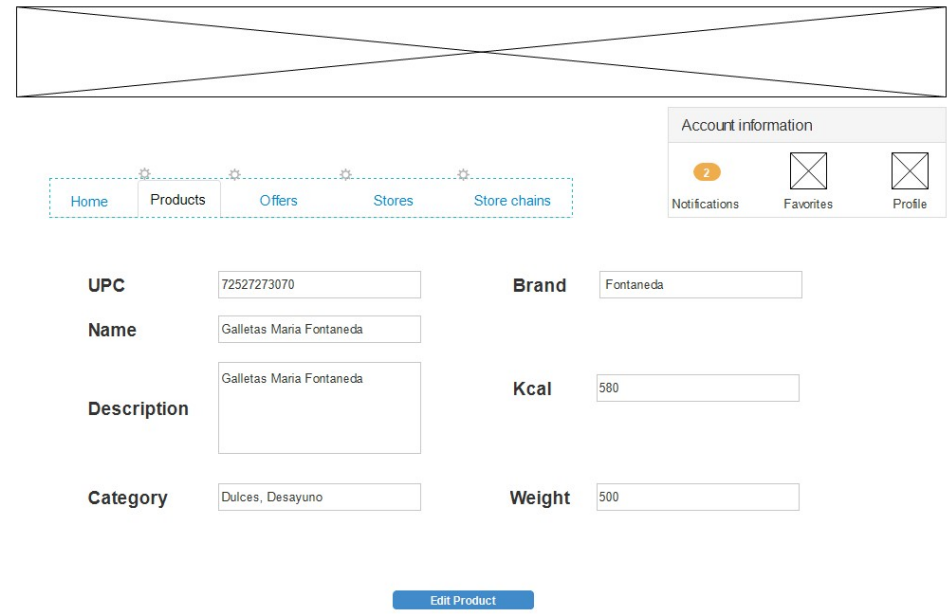


Figura 32: Lo-Fi - Página: Editar producto

15.1.11 Página: Ofertas

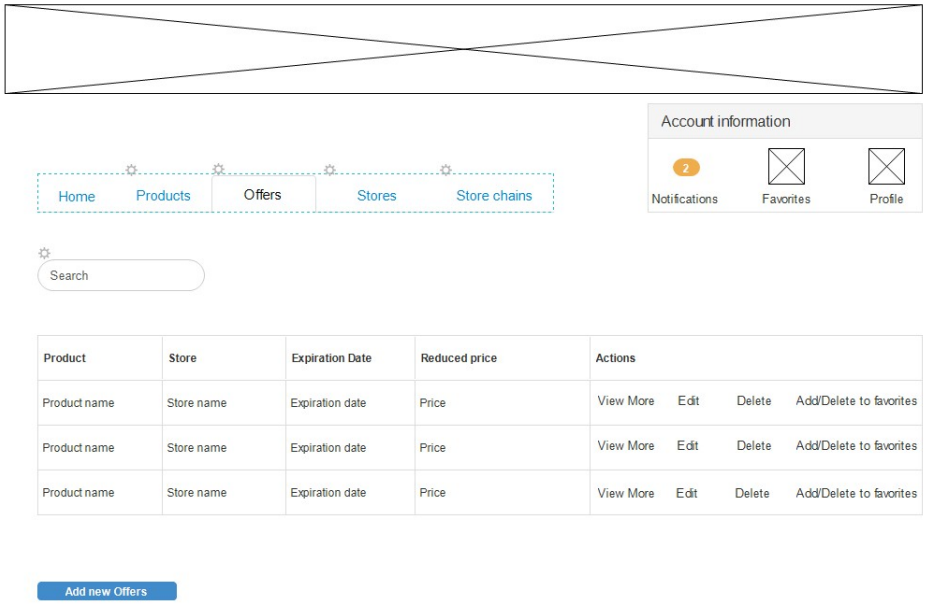


Figura 33: Lo-Fi - Página: Ofertas

15.1.12 Página: Ver oferta

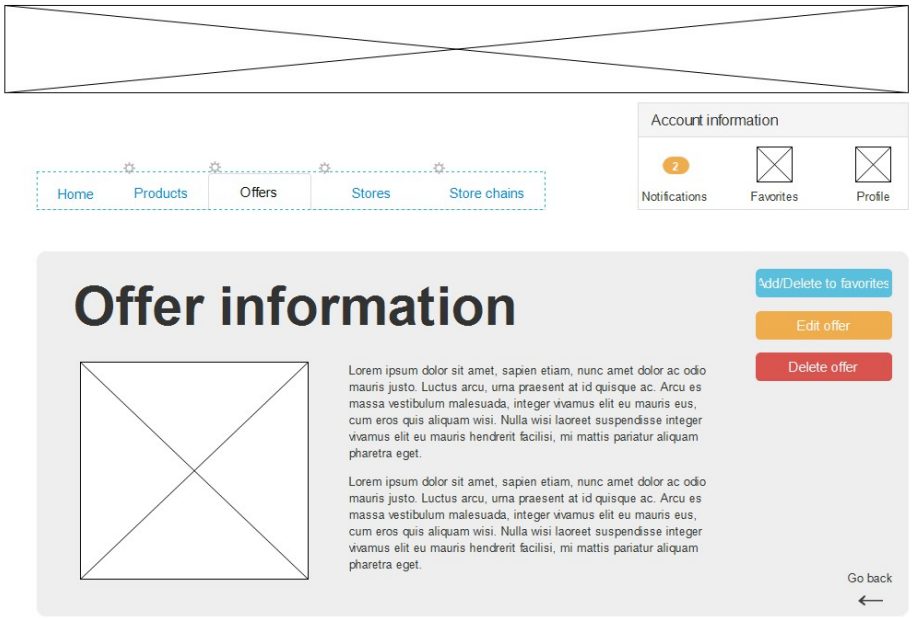


Figura 34: Lo-Fi - Página: Ver Ofertas

15.1.13 Página: Añadir lista de ofertas

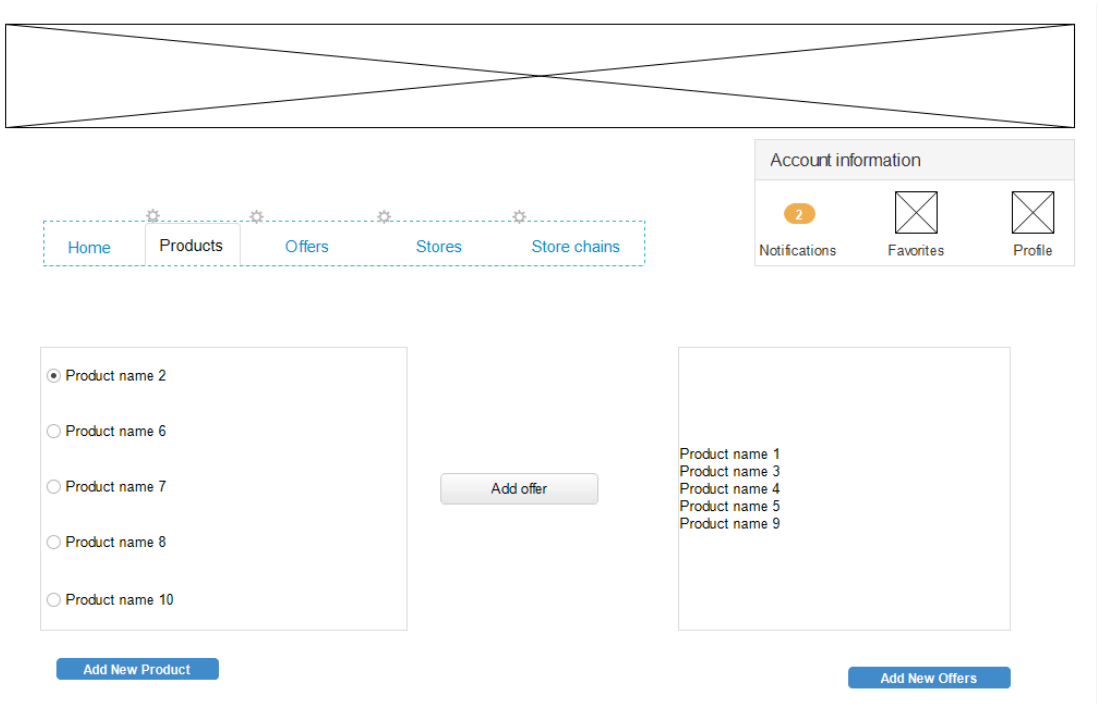


Figura 35: Lo-Fi - Página: Añadir lista de ofertas

15.1.14 Página: Editar oferta

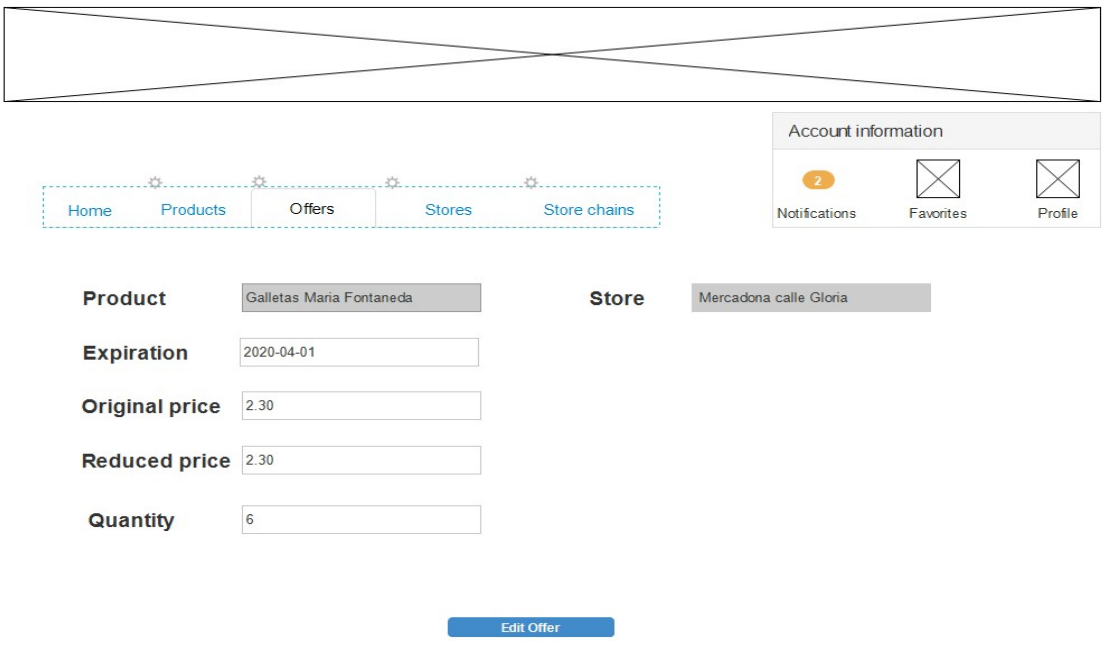


Figura 36: Lo-Fi - Página: Editar oferta

15.1.15 Página: Tiendas

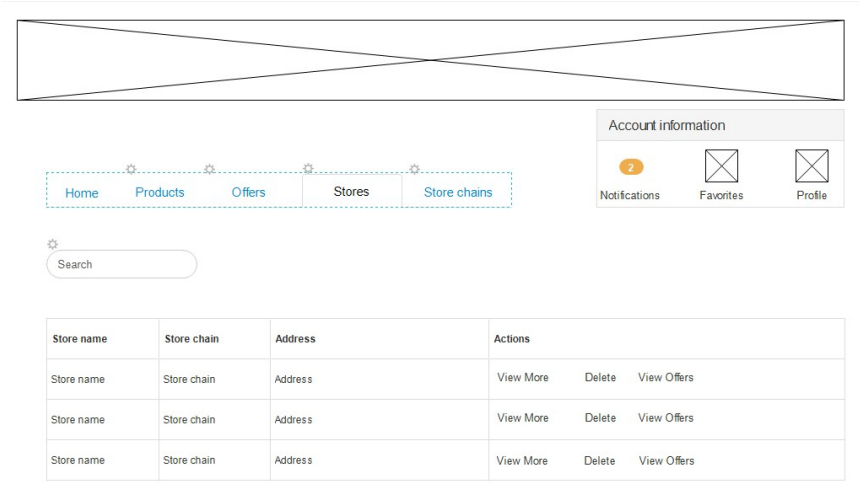


Figura 37: Lo-Fi - Página: Tiendas

15.1.16 Página: Ver tienda

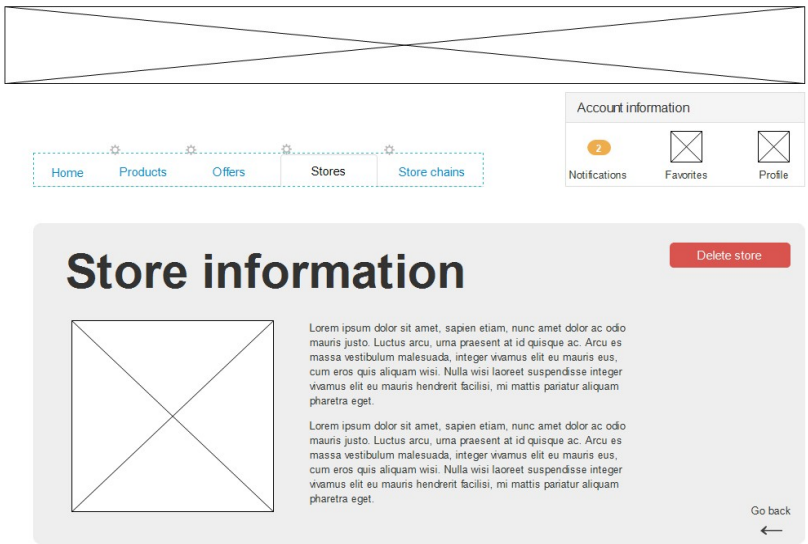


Figura 38: Lo-Fi - Página: Ver tienda

15.1.17 Página: Cadenas de tiendas

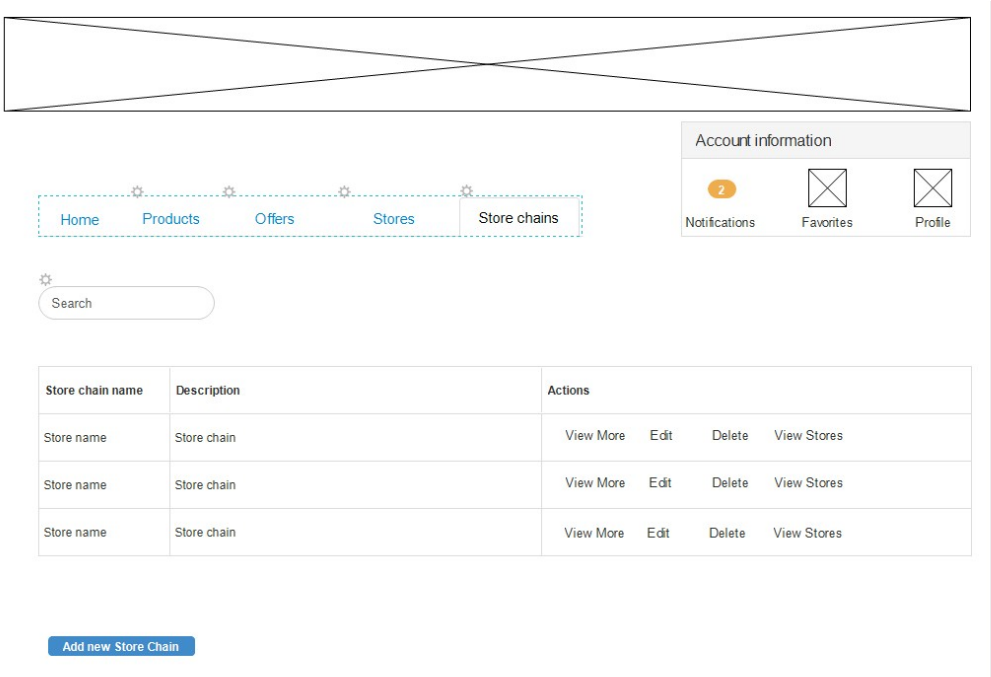


Figura 39: Lo-Fi - Página: Cadena de tiendas

15.1.18 Página: Ver cadena de tiendas

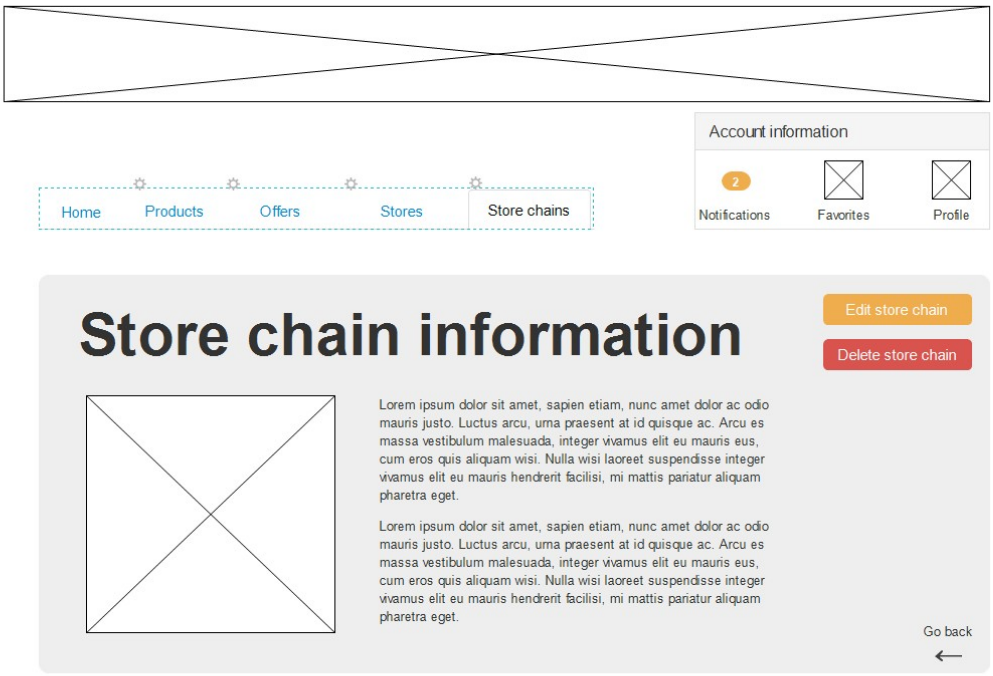


Figura 40: Lo-Fi - Página: Ver cadena de tiendas

15.1.19 Página: Añadir cadena de tiendas

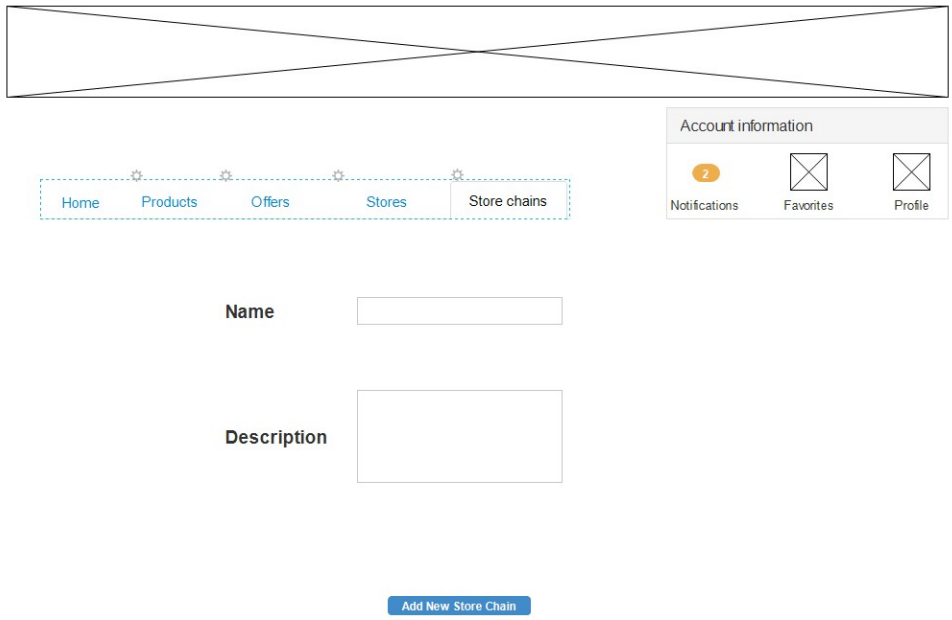


Figura 41: Lo-Fi - Página: Añadir cadena de tiendas

15.1.20 Página: Editar cadena de tiendas

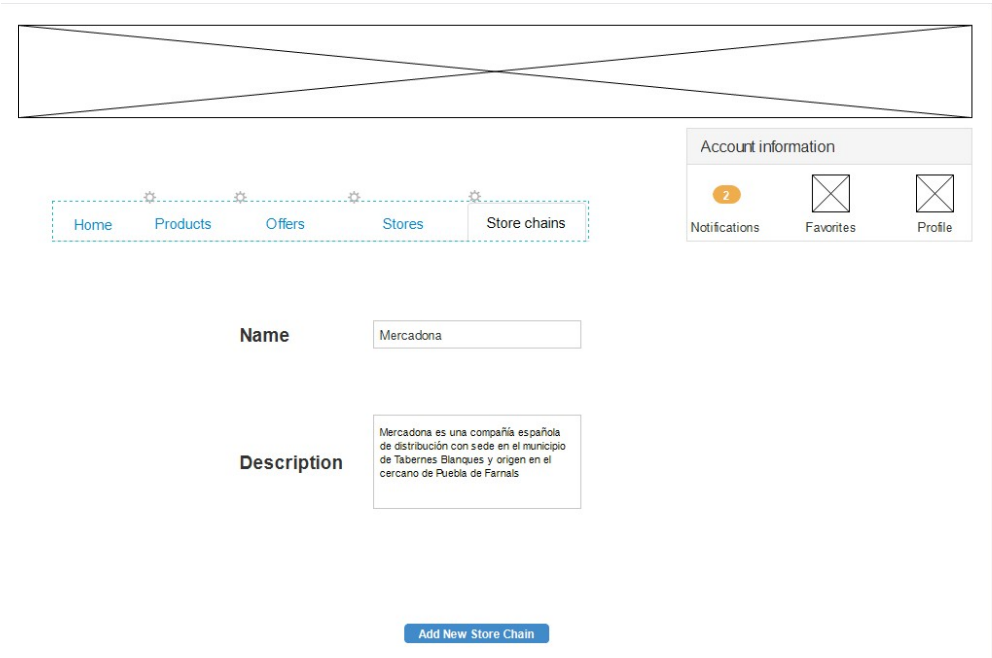


Figura 42: Lo-Fi - Página: Editar cadena de tiendas

15.1.21 Página: Ver notificaciones

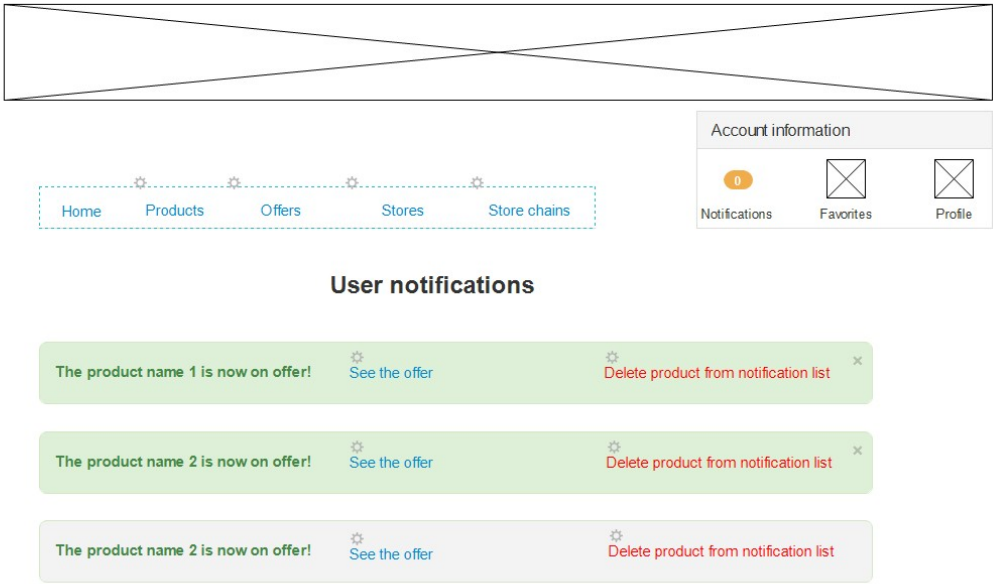


Figura 43: Lo-Fi - Página: Ver notificaciones

15.1.22 Página: Ver ofertas favoritas

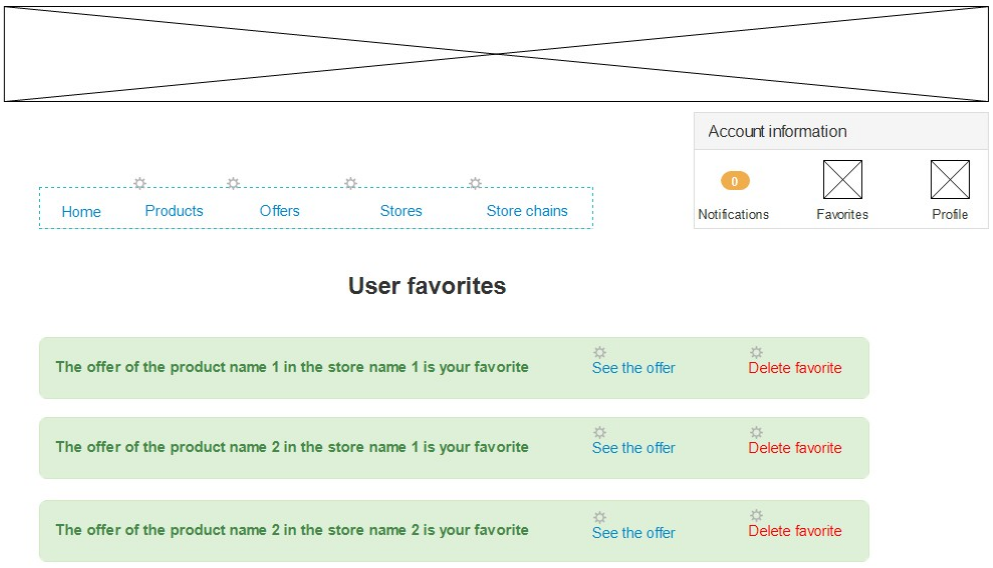


Figura 44: Lo-Fi - Página: Ver ofertas favoritas

15.1.23 Página: Ver perfil de cuenta de usuario

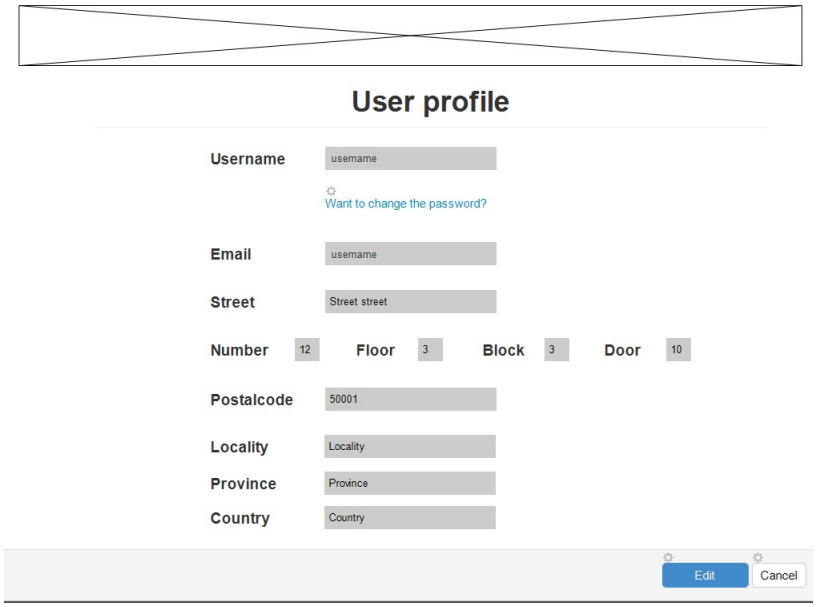


Figura 45: Lo-Fi - Página: Ver perfil de cuenta de usuario

15.1.24 Página: Editar perfil de cuenta de usuario

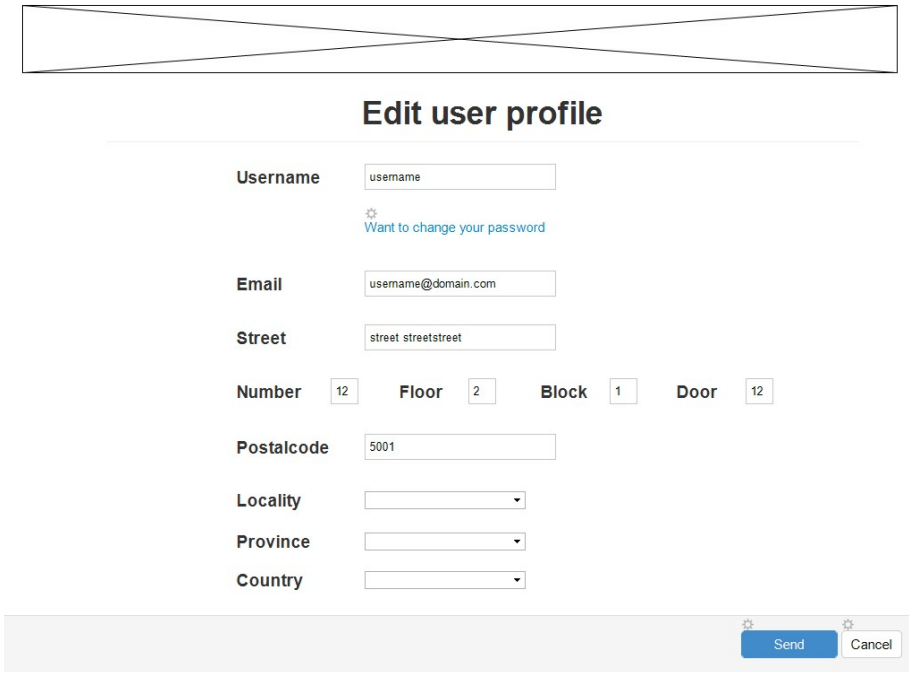


Figura 46: Lo-Fi - Página: Editar perfil de cuenta de usuario

15.1.25 Página: Ver perfil de cuenta de tienda

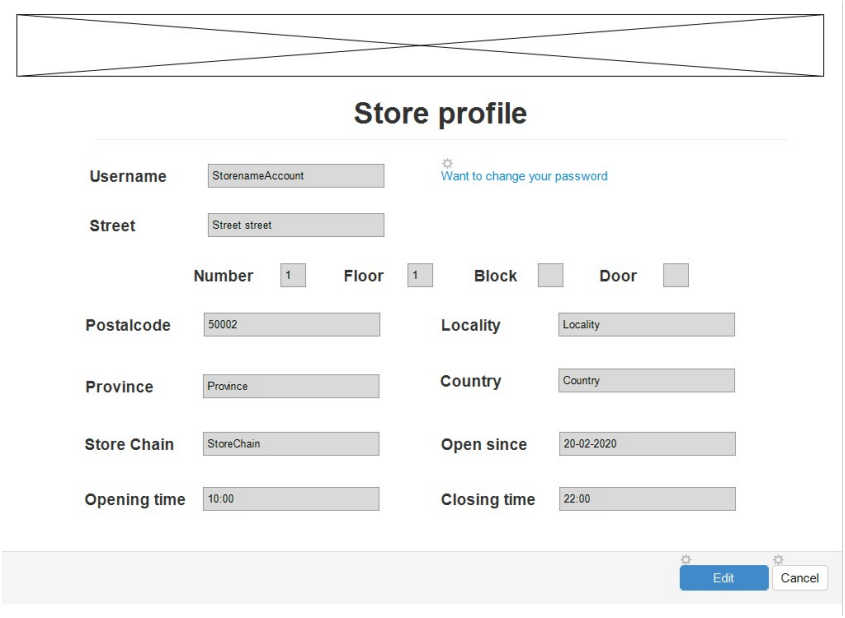


Figura 47: Lo-Fi - Página: Ver perfil de cuenta de tienda

15.1.26 Página: Editar perfil de cuenta de tienda

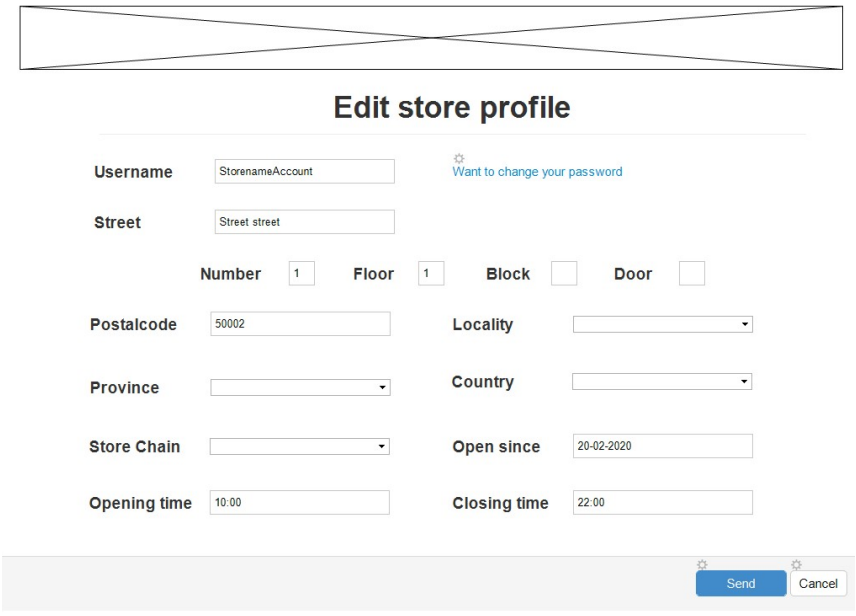


Figura 48: Lo-Fi - Página: Editar perfil de cuenta de tienda

15.2 Hi-Fi

A continuación se presenta el prototipo de alta fidelidad de las diferentes ventanas de las que constará la aplicación de alimentación sostenible:

15.2.1 Página: Inicial

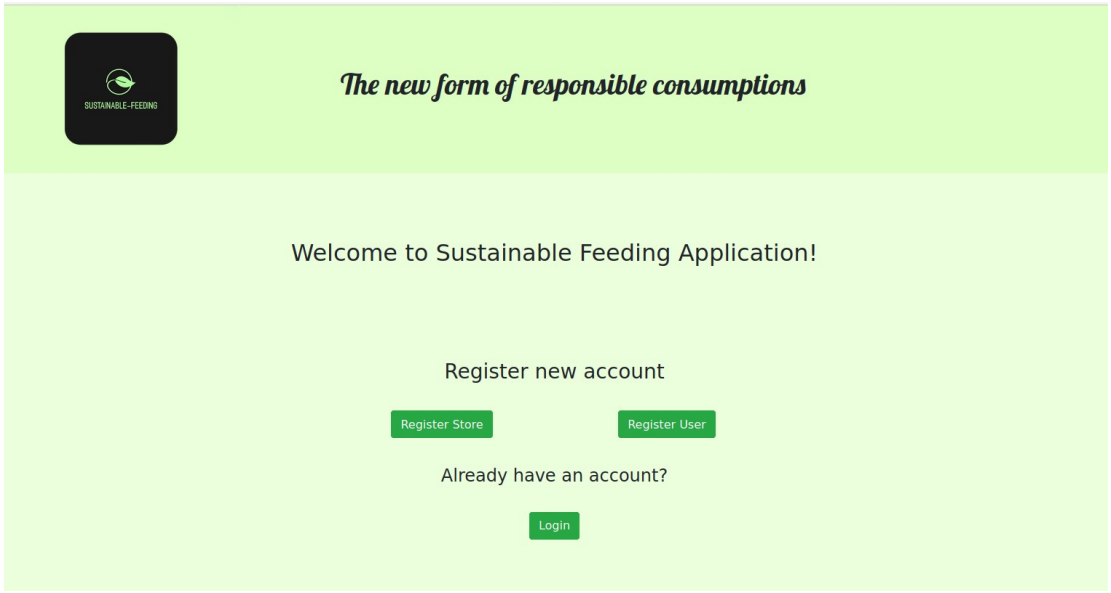


Figura 49: Hi-Fi - Página: Inicial

15.2.2 Página: Login

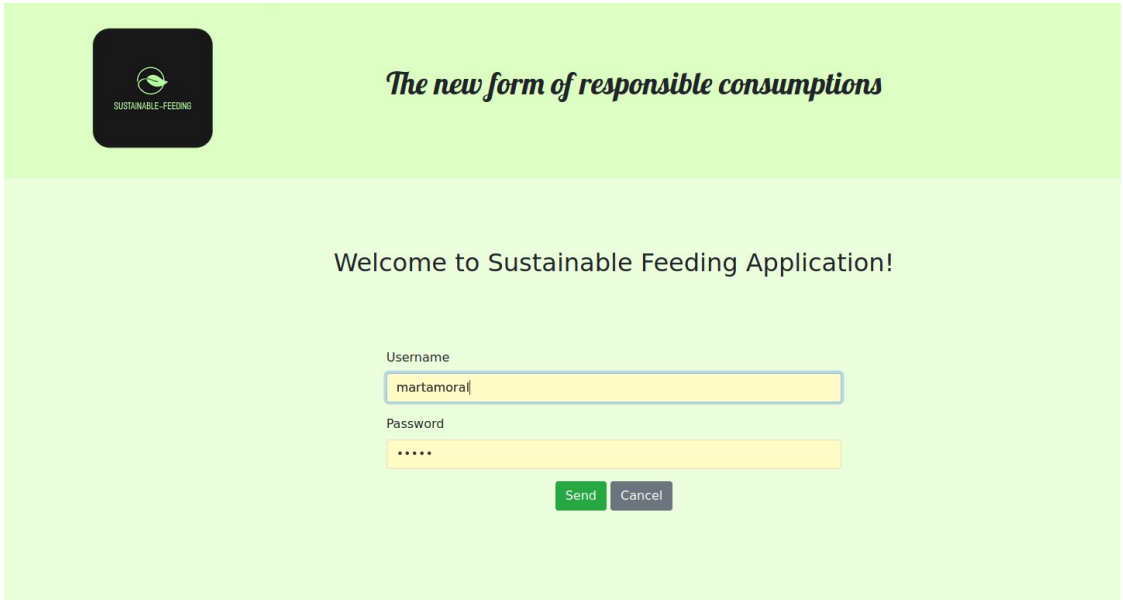


Figura 50: Hi-Fi - Página de login

15.2.3 Página: Registro de usuario

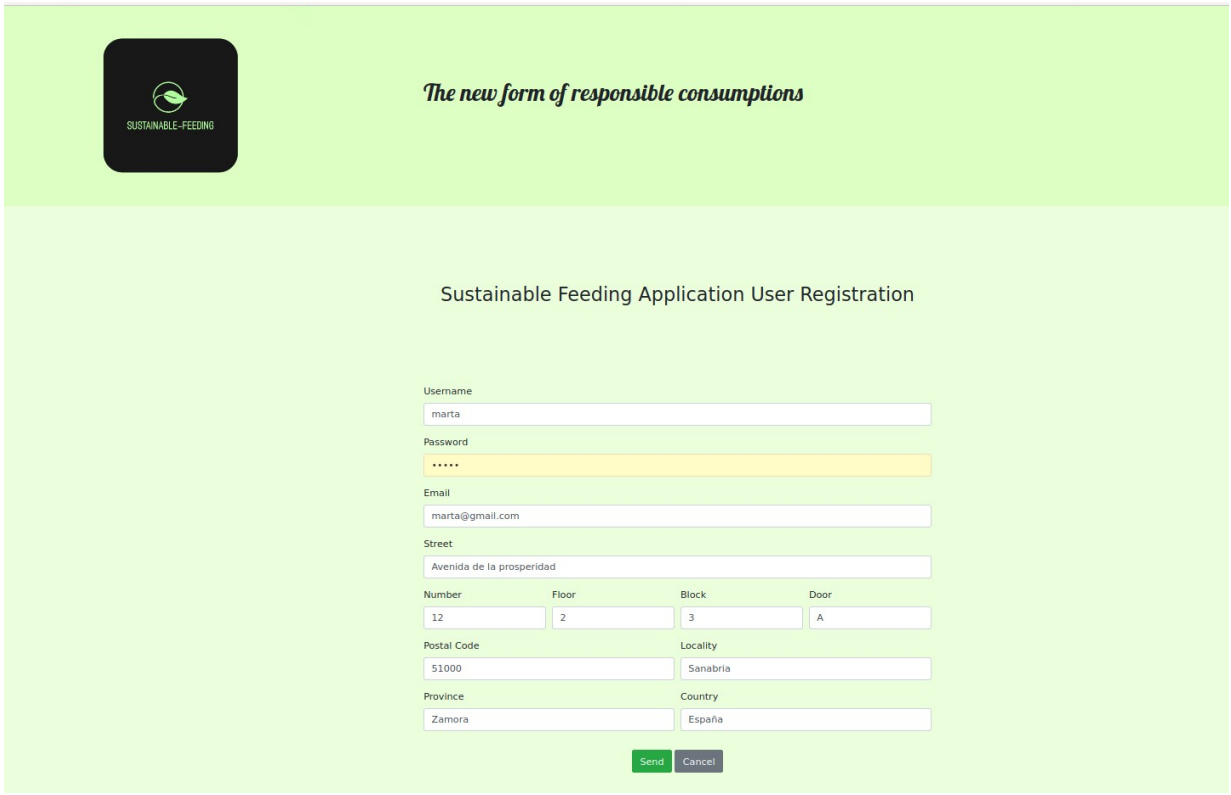


Figura 51: Hi-Fi - Página: Registro de usuario

15.2.4 Página: Registro de tienda

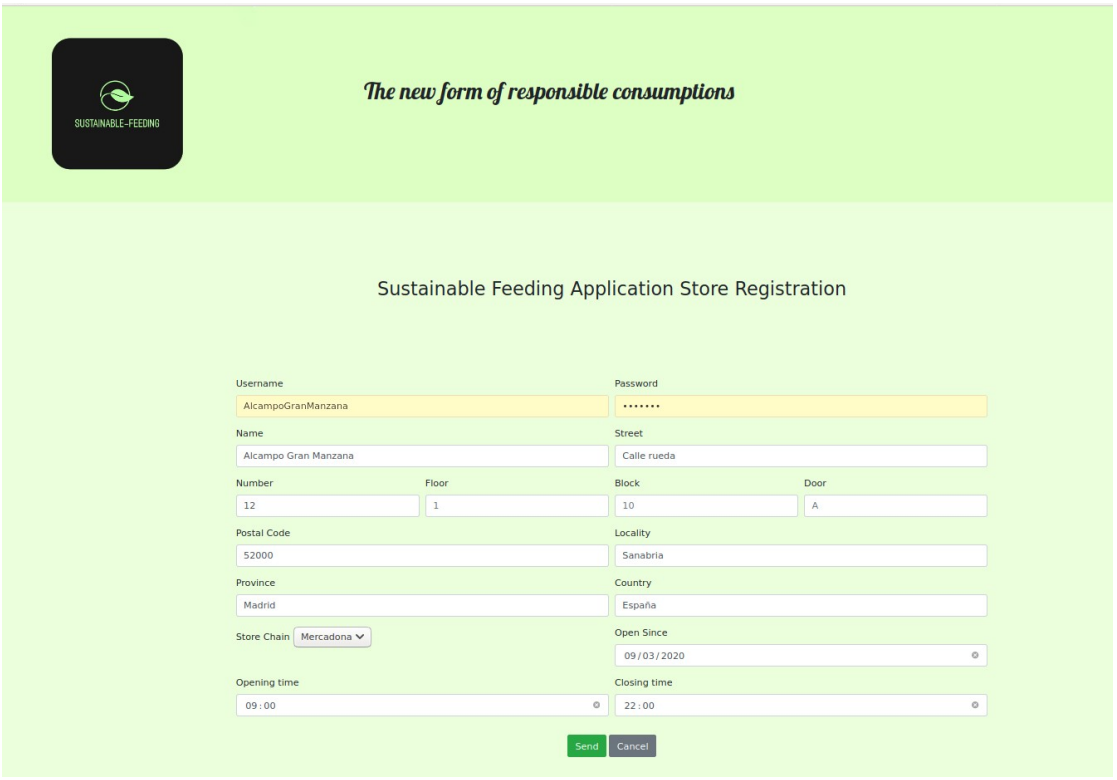


Figura 52: Hi-Fi - Página: Registro de tienda

15.2.5 Página: Home

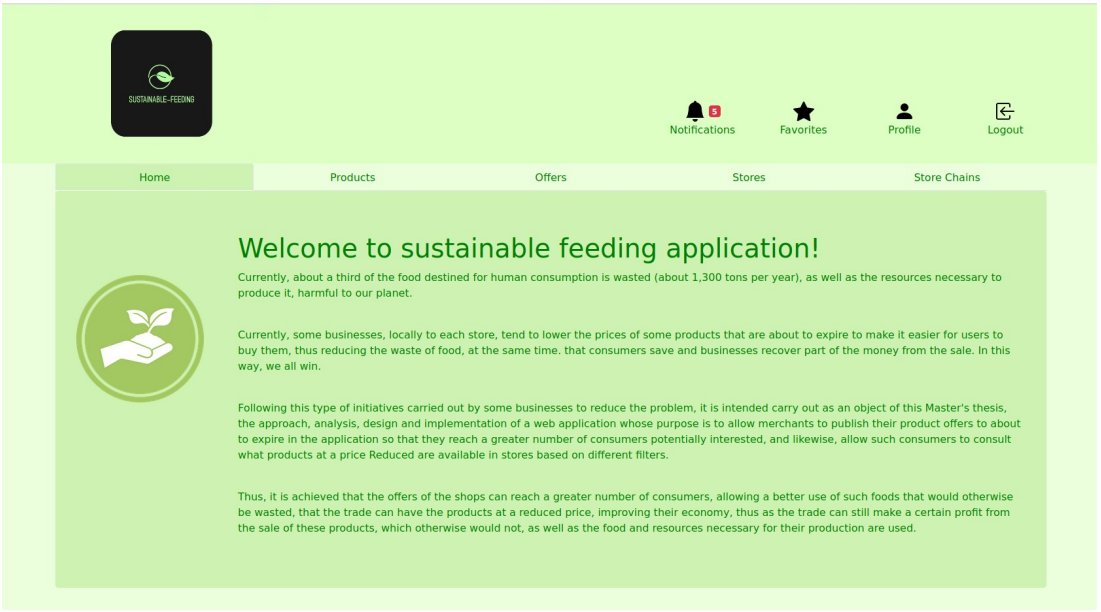


Figura 53: Hi-Fi - Página: Home

15.2.6 Página: Productos

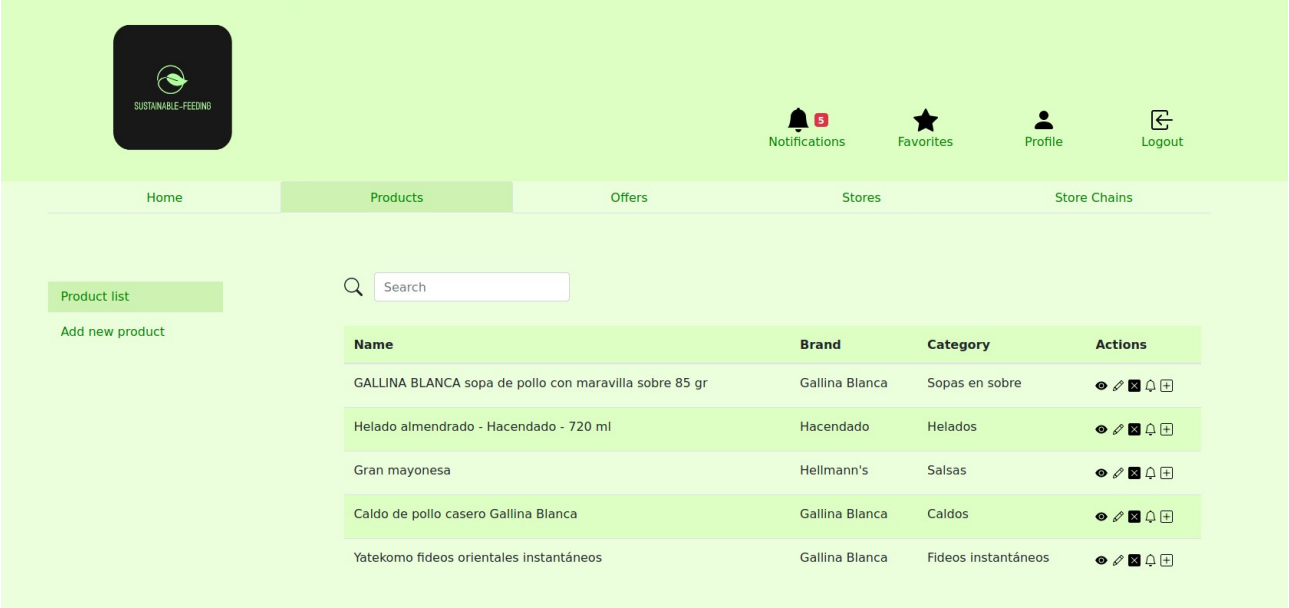


Figura 54: Hi-Fi - Página: Productos

15.2.7 Ventana modal: Añadir Oferta de un producto

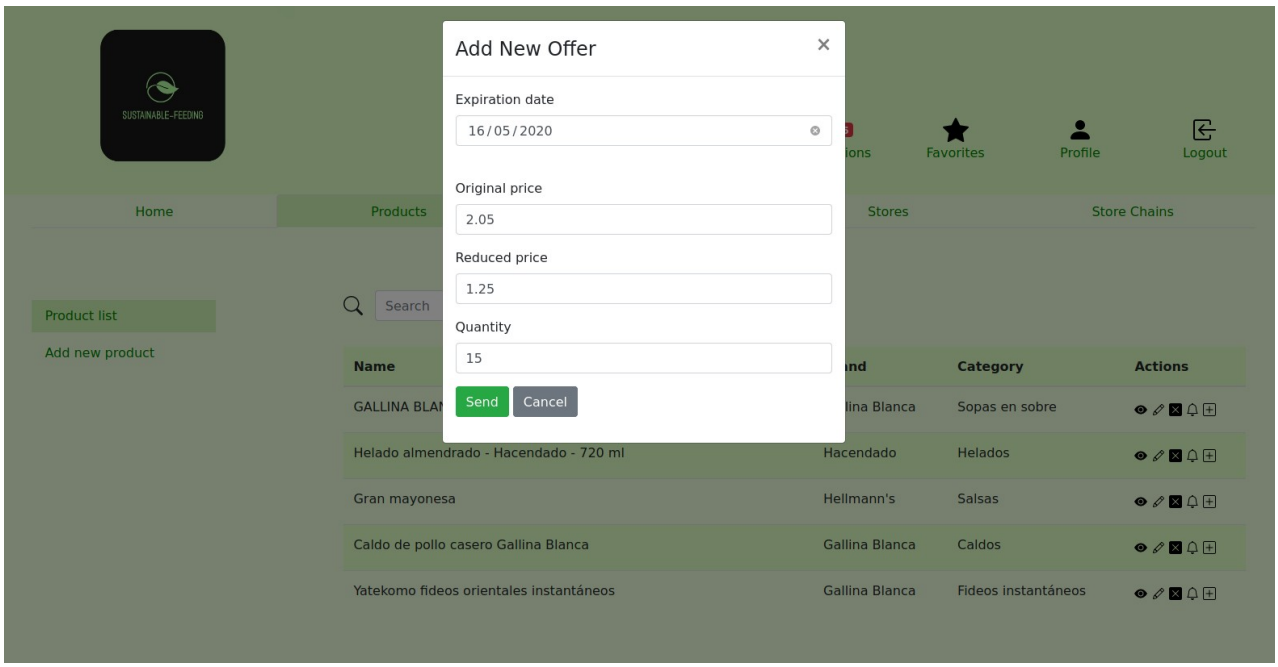


Figura 55: Hi-Fi - Ventana Modal: añadir oferta de producto

15.2.8 Página: Ver producto

Figura 56: Hi-Fi - Página: Ver producto

15.2.9 Página: Añadir producto

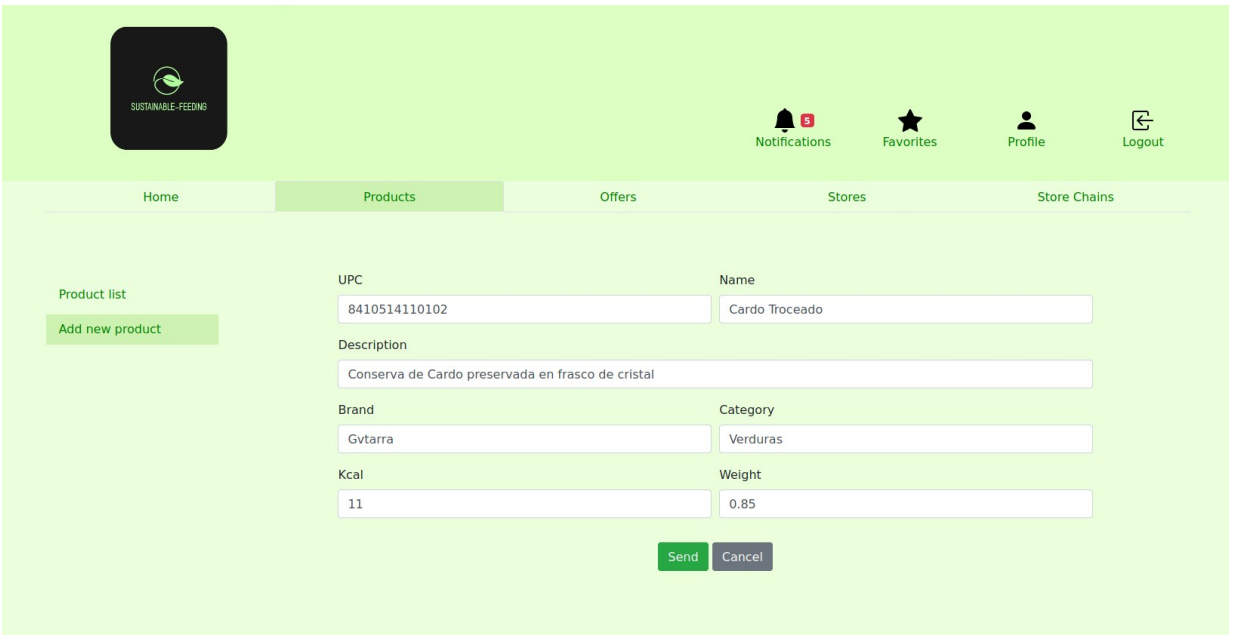


Figura 57: Hi-Fi - Página: Añadir producto

15.2.10 Página: Editar producto

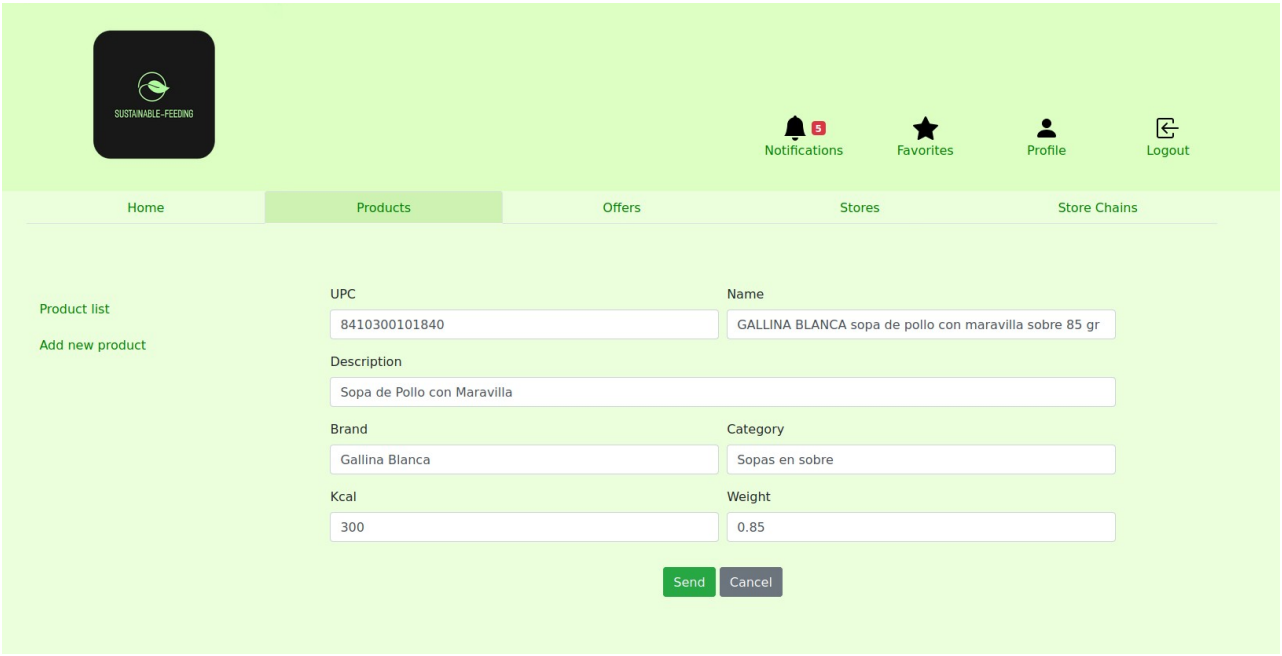


Figura 58: Hi-Fi - Página: Editar producto

15.2.11 Página: Ofertas

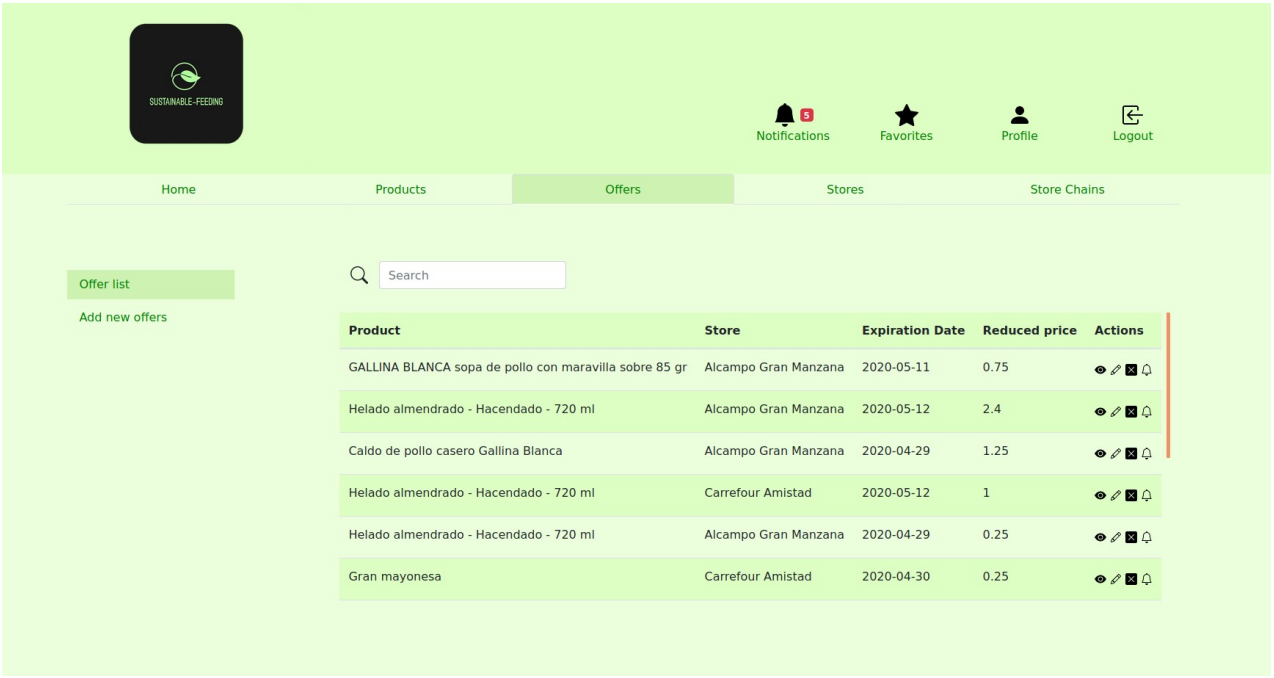


Figura 59: Hi-Fi - Página: Ofertas

15.2.12 Página: Ver oferta

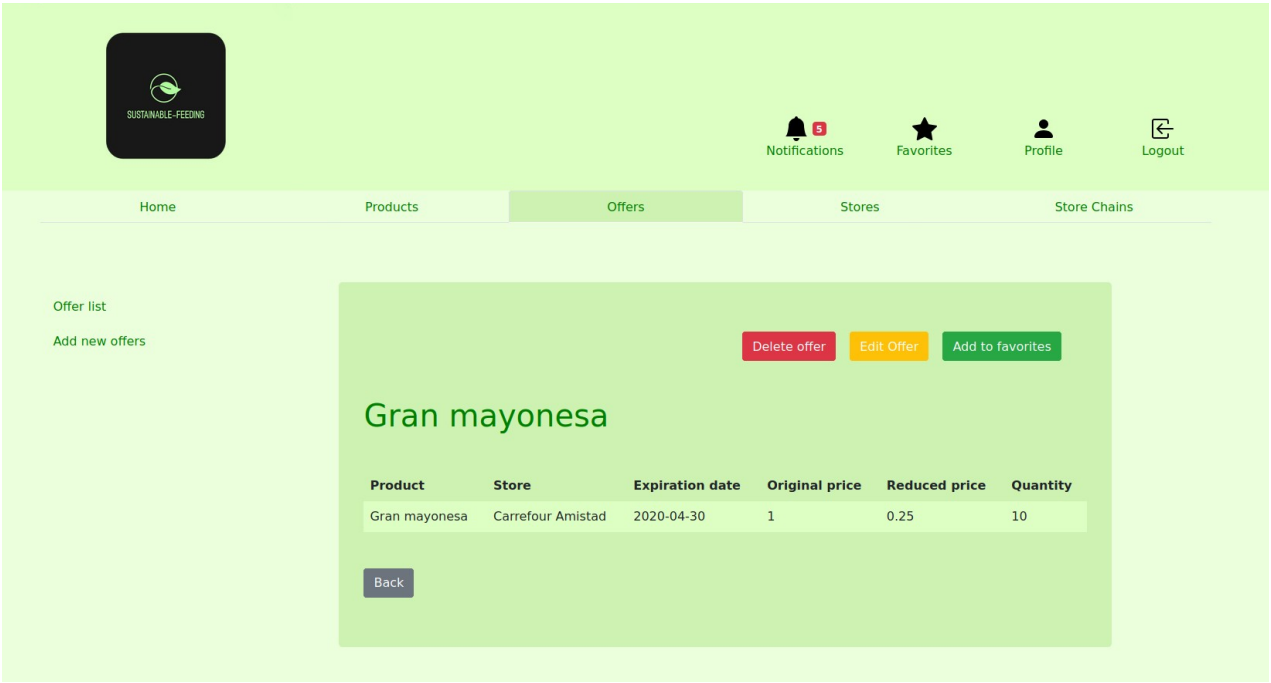


Figura 60: Hi-Fi - Página: Ver Ofertas

15.2.13 Página: Añadir lista de ofertas

Figura 61: Hi-Fi - Página: Añadir lista de ofertas

15.2.14 Página: Editar oferta

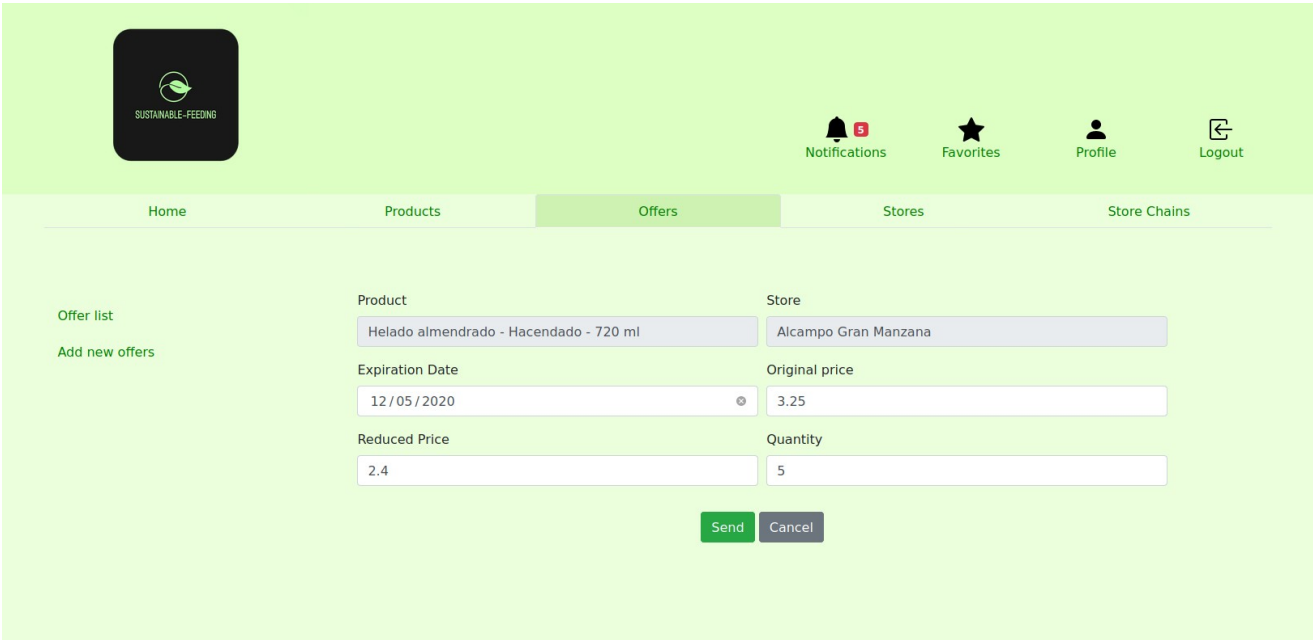


Figura 62: Hi-Fi - Página: Editar oferta

15.2.15 Página: Tiendas

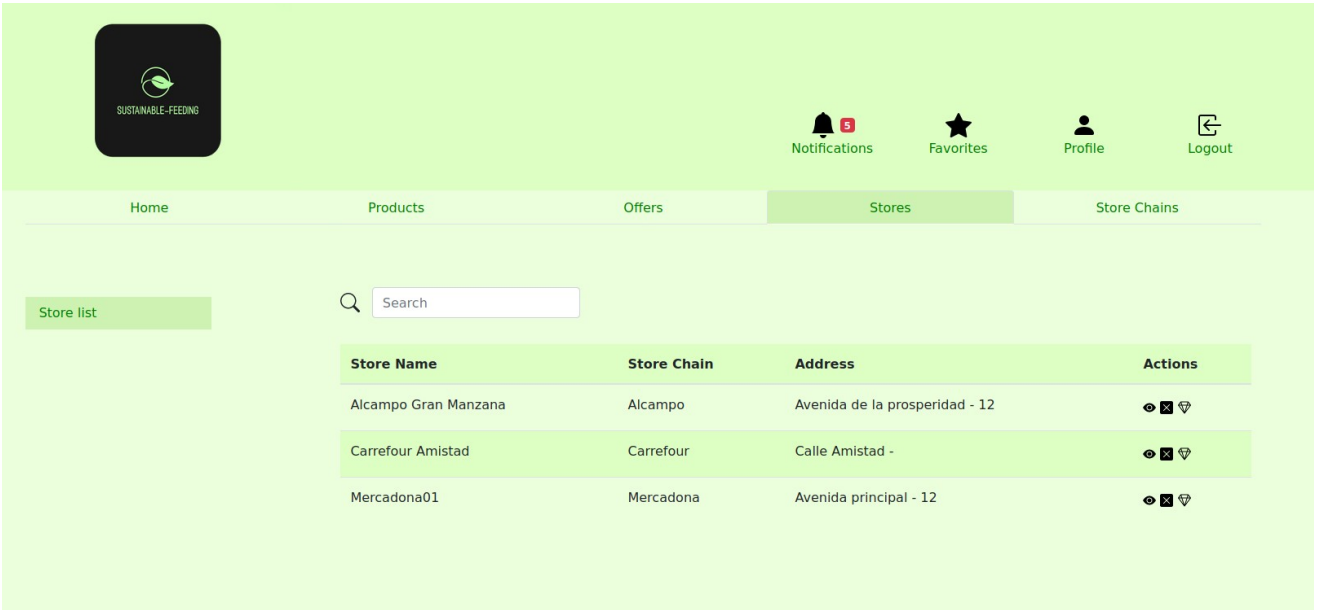


Figura 63: Hi-Fi - Página: Tiendas

15.2.16 Página: Ver tienda

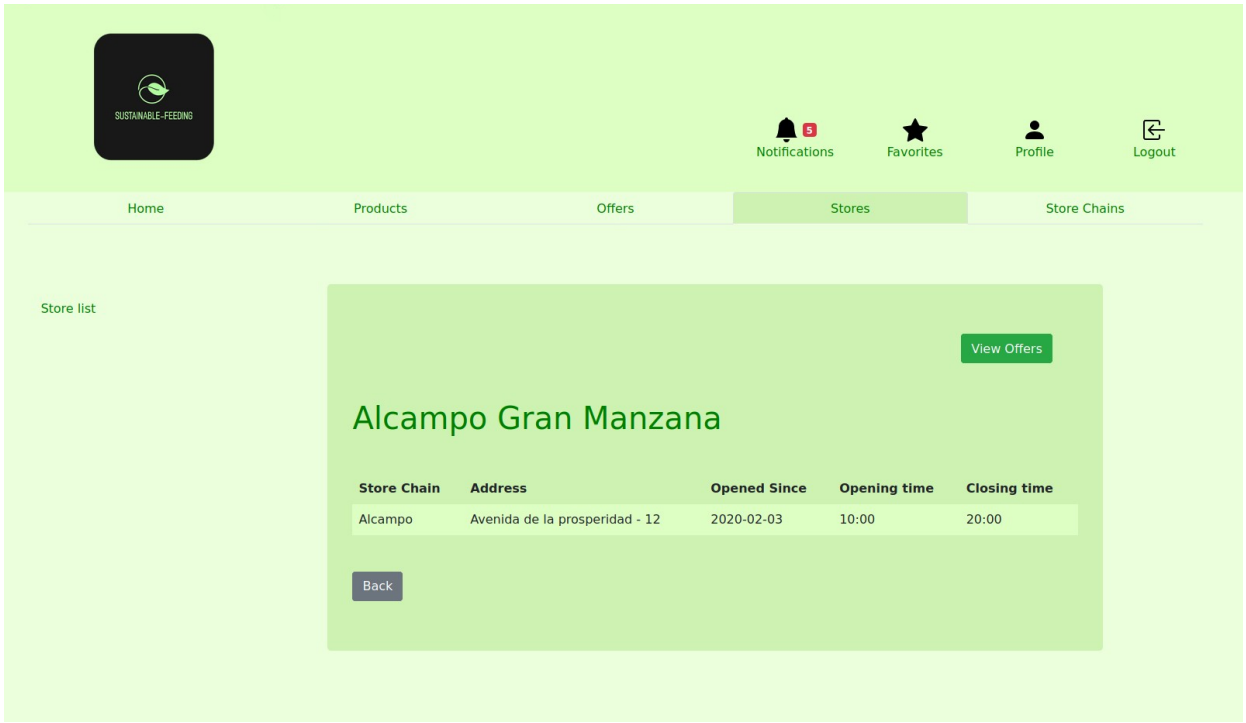


Figura 64: Hi-Fi - Página: Ver tienda

15.2.17 Página: Cadenas de tiendas

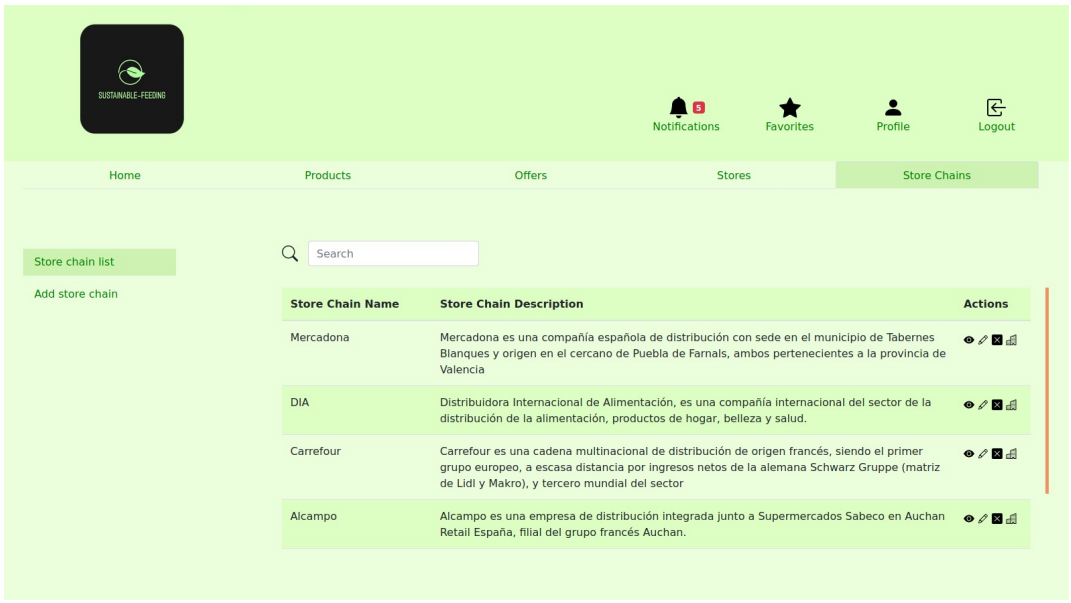


Figura 65: Hi-Fi - Página: Cadena de tiendas

15.2.18 Página: Ver cadena de tiendas

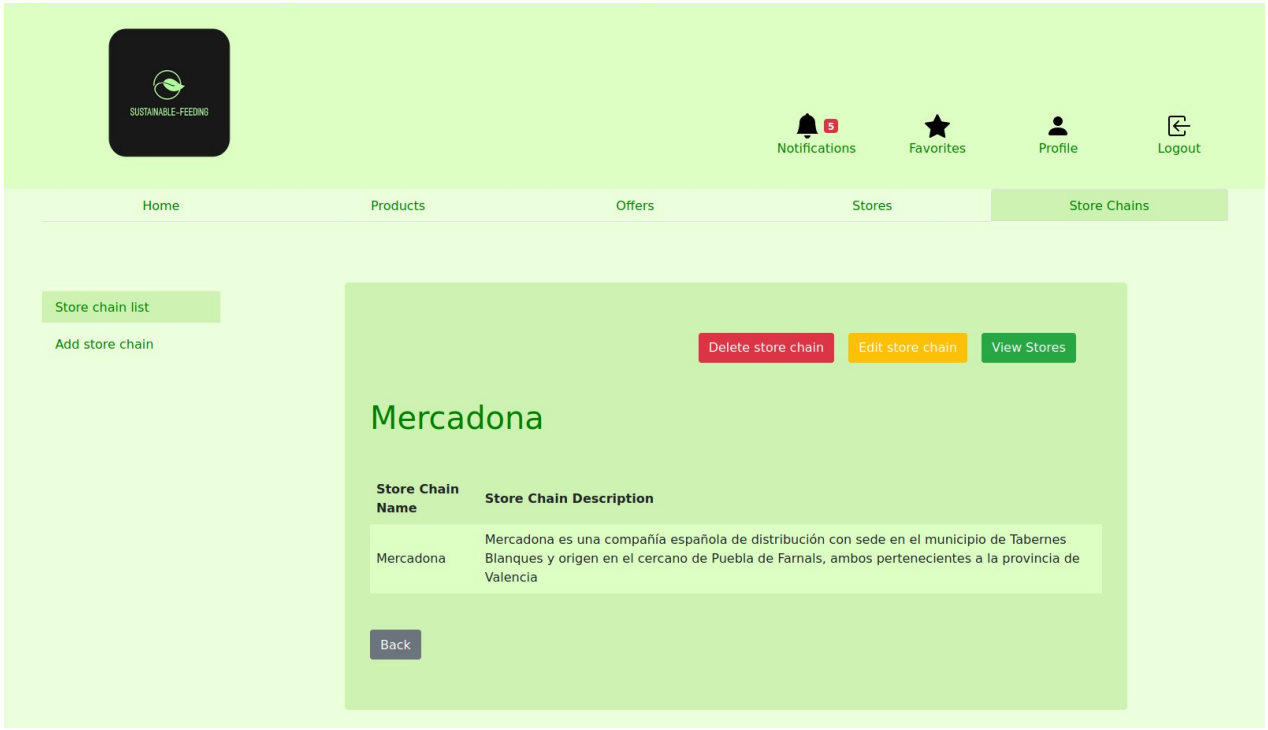


Figura 66: Hi-Fi - Página: Ver cadena de tiendas

15.2.19 Página: Añadir cadena de tiendas

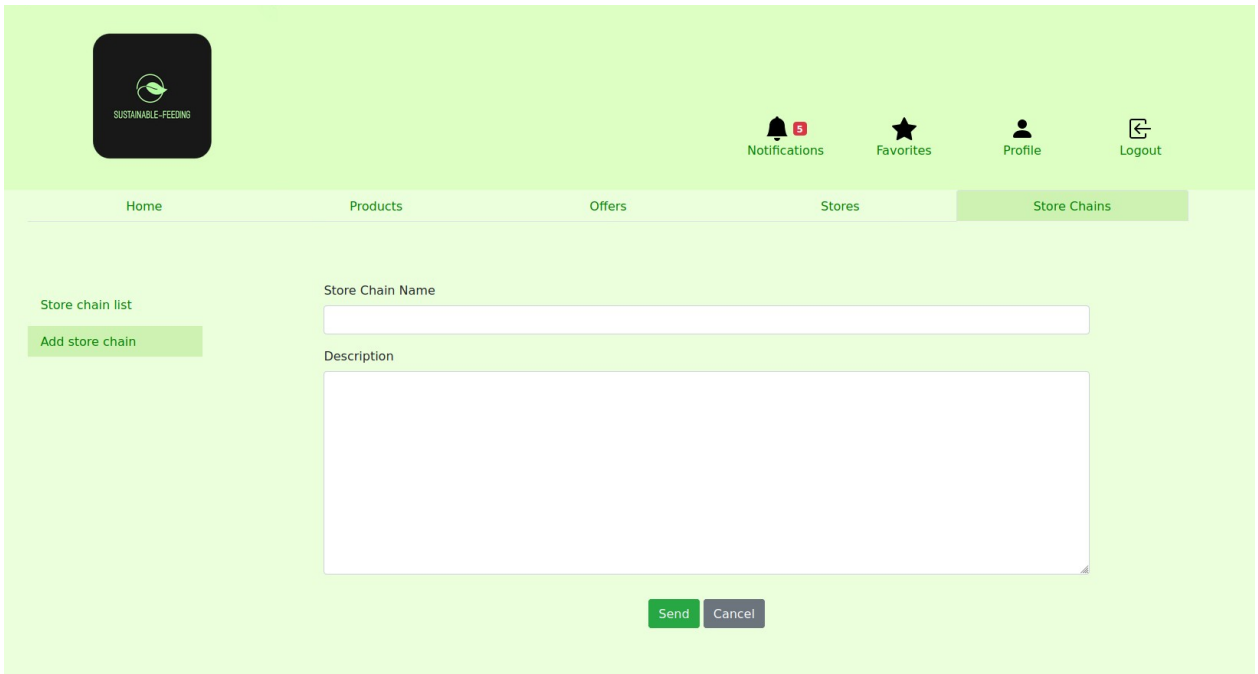


Figura 67: Hi-Fi - Página: Añadir cadena de tiendas

15.2.20 Página: Editar cadena de tiendas

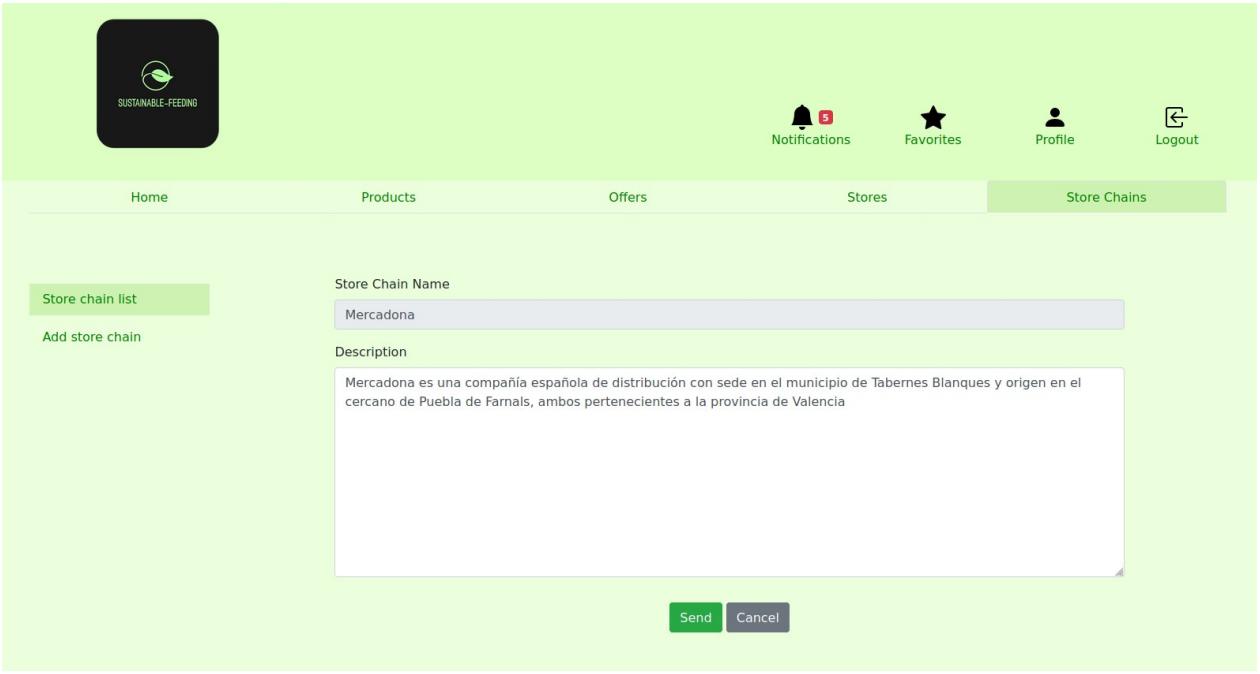


Figura 68: Hi-Fi - Página: Editar cadena de tiendas

15.2.21 Página: Ver notificaciones

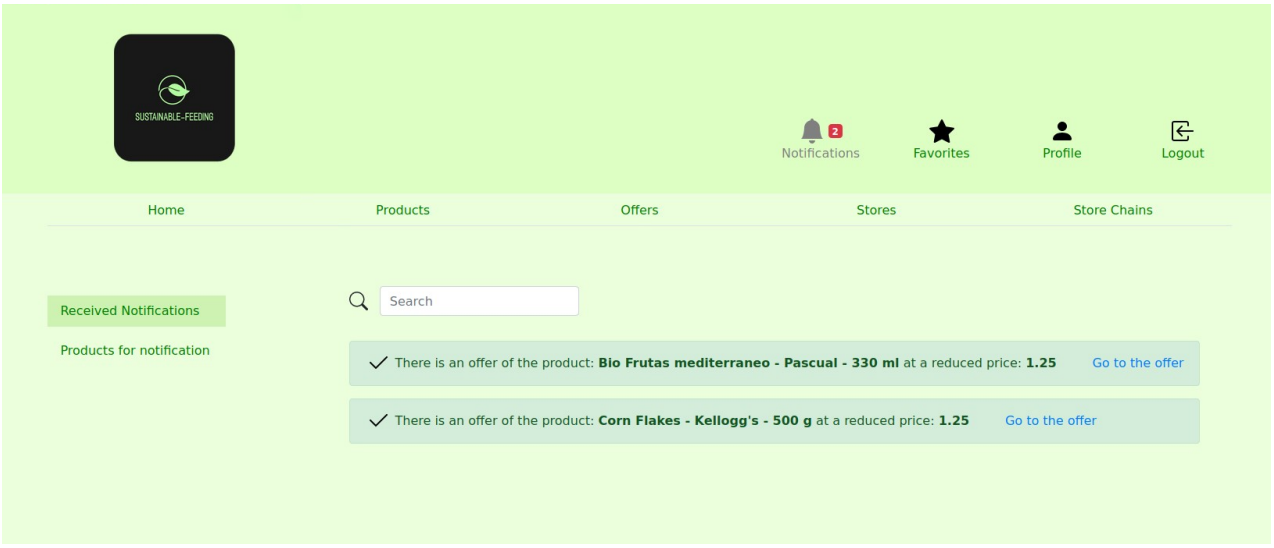


Figura 69: Hi-Fi - Página: Ver notificaciones

15.2.22 Página: Ver ofertas favoritas

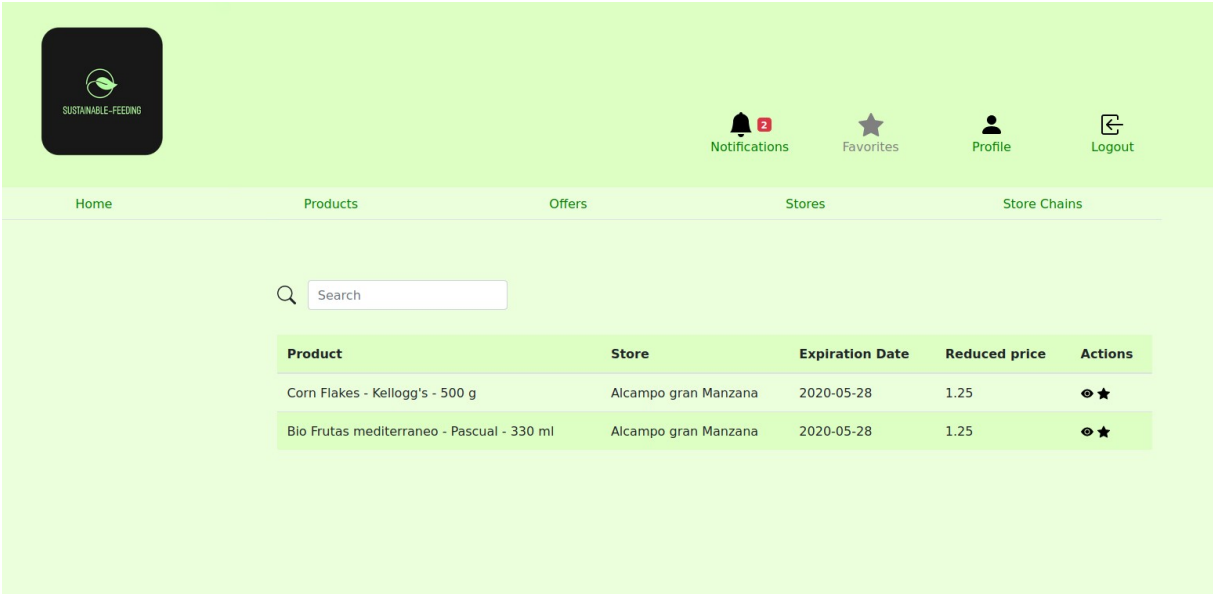


Figura 70: Hi-Fi - Página: Ver ofertas favoritas

15.2.23 Página: Ver perfil de cuenta de usuario

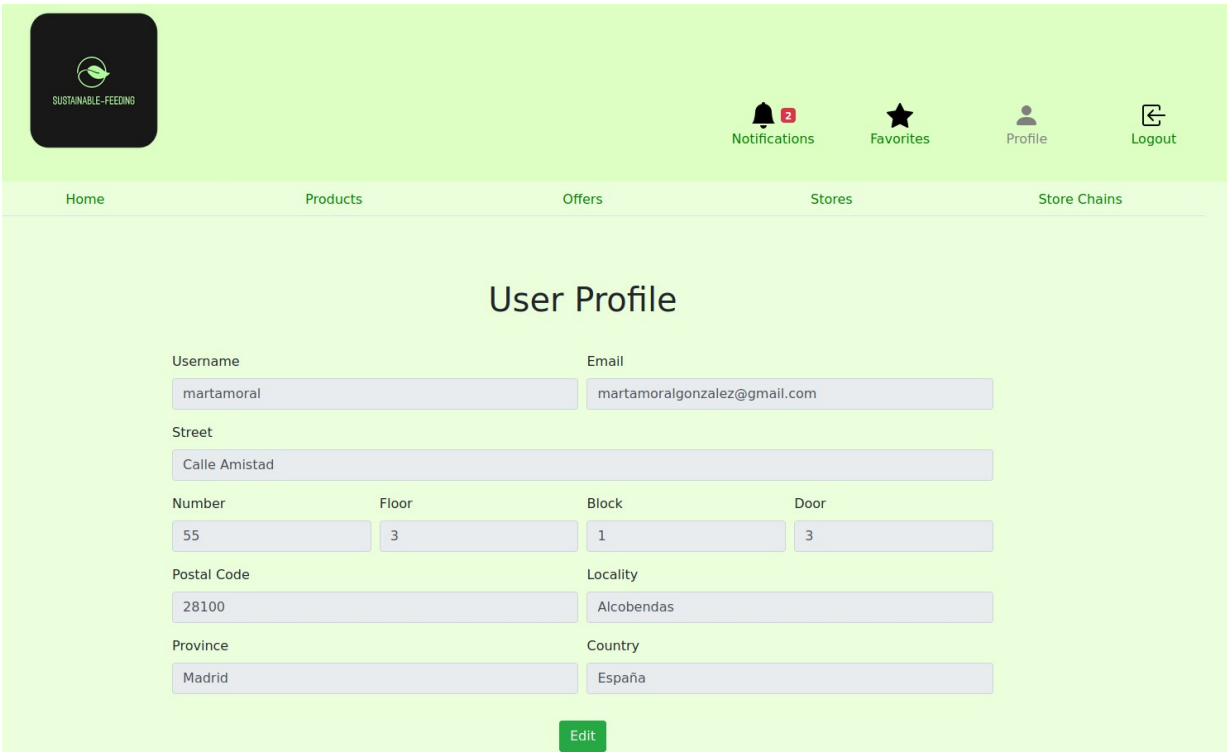


Figura 71: Hi-Fi - Página: Ver perfil de cuenta de usuario

15.2.24 Página: Editar perfil de cuenta de usuario

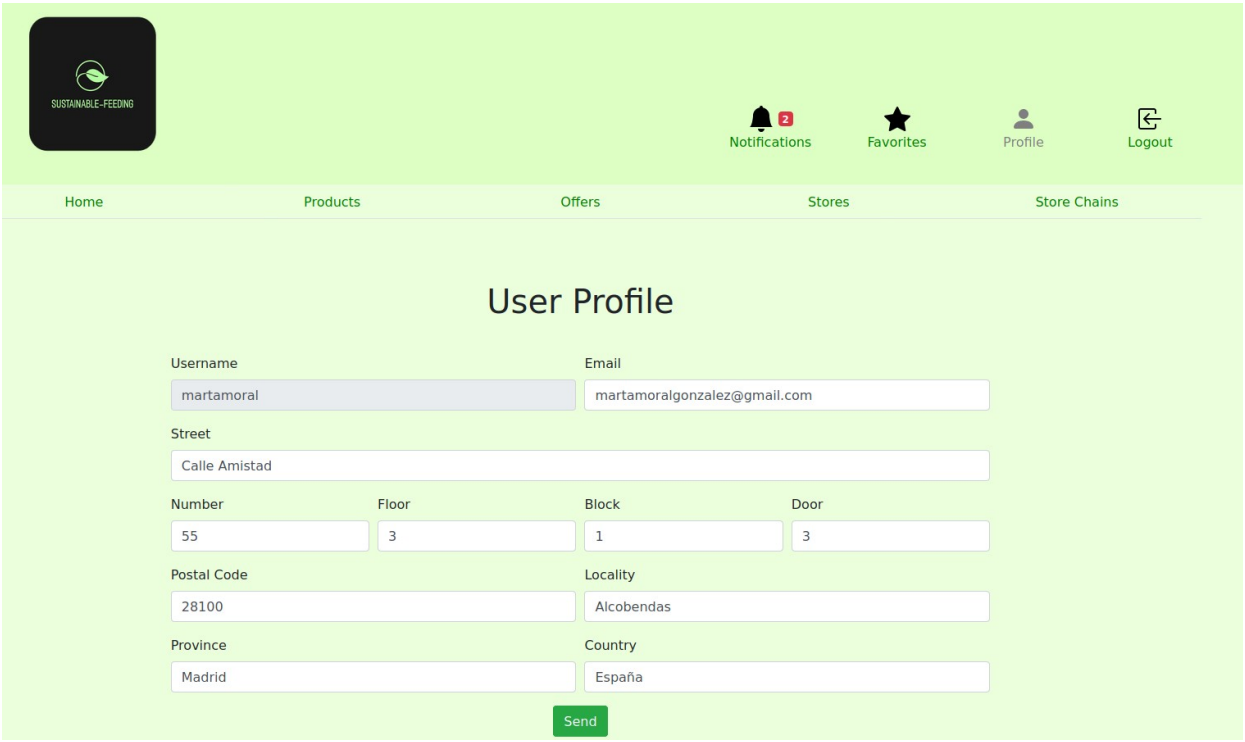


Figura 72: Hi-Fi - Página: Editar perfil de cuenta de usuario

15.2.25 Página: Ver perfil de cuenta de tienda

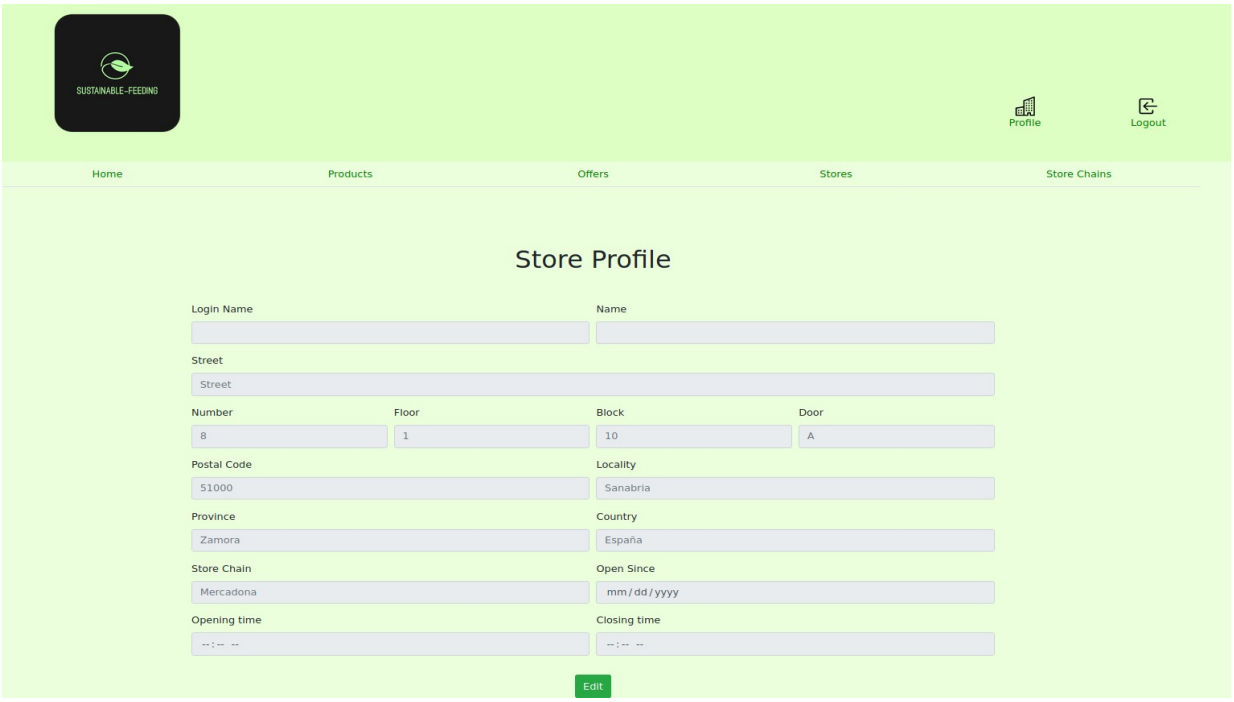


Figura 73: Hi-Fi - Página: Ver perfil de cuenta de tienda

15.2.26 Página: Editar perfil de cuenta de tienda

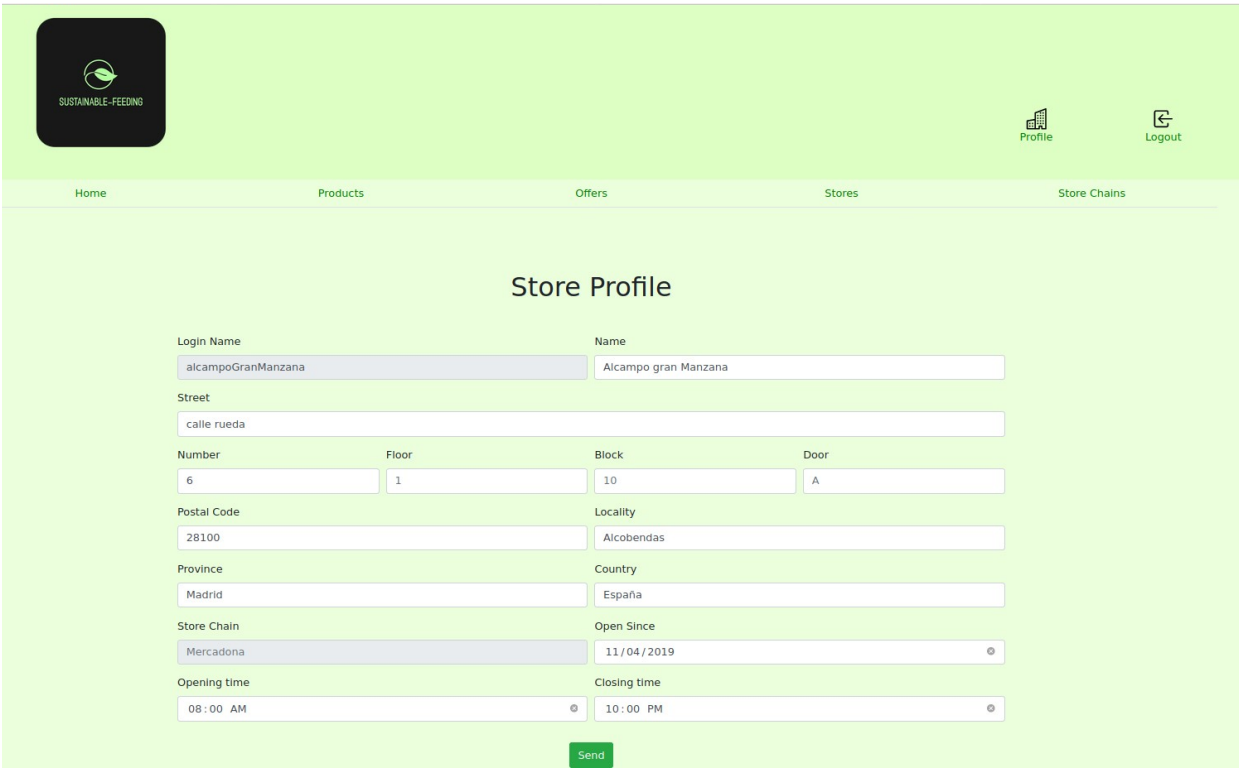


Figura 74: Hi-Fi - Página: Editar perfil de cuenta de tienda

16. Guiones

No se han creado ningún guión para la realización del presente TFM.

17. Perfiles de usuario

Dentro de la aplicación se contemplan 3 perfiles diferentes:

Perfil Usuario

Es el perfil correspondiente a los consumidores que se registren en la aplicación de alimentación sostenible para encontrar productos con fecha de caducidad próxima que les sean más asequibles económicamente.

Se espera que el perfil de los usuarios de la aplicación sea el de personas jóvenes que estén independizadas y sean independientes, pues deben hacer la compra ellos mismos y valorarán el ahorro que pueda suponer el comprar productos con fecha de caducidad próxima.

También, es el colectivo más informado y concienciado acerca del consumo sostenible y la contaminación, y además tienen un amplio dominio de las nuevas tecnologías.

Por tanto, se espera que el grueso de los consumidores que harán uso de la aplicación de alimentación sostenible sean personas entre los 25 y los 45 años, independizados e independientes, con amplio dominio de las nuevas tecnologías y con independencia de su estado civil, sexo y número de familiares a su cargo.

Perfil Tienda

Es el perfil correspondiente a las tiendas y comercios que se registran en la aplicación de alimentación sostenible con el fin de publicar ofertas de productos cuya fecha de caducidad está próxima.

Se espera que cualquier tienda pueda participar en la iniciativa, tanto tiendas pertenecientes a grandes cadenas comerciales (Mercadona, Alcampo, Carrefour, etc) como pequeños comercios y tiendas de alimentación de barrio.

Perfil Administrador

Es el perfil correspondiente al rol de administrador de la aplicación, el cual será el superusuario que tendrá acceso a las operaciones de gestión y mantenimiento de la información de la aplicación.

18. Usabilidad/UX

18.1 Navegación

A continuación se presentan las diferentes ventanas y la navegación entre las mismas de las que consta la aplicación web de alimentación sostenible.

18.1.1 Página: Inicial

Es la ventana inicial de la aplicación, a la cual tanto los consumidores como los comercios acceden tanto si tienen cuenta en la aplicación como si no.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.1 y 15.2.1 Página: Inicial* del apartado *15. Prototipos.*

Desde esta ventana puede accederse a las ventanas de Acceso a la aplicación, de Registro de usuario y de Registro de tienda, seleccionando cada uno de los botones respectivamente.

18.1.2 Página: Login

Es la ventana en la cual los usuarios consumidores y los establecimientos que ya se encuentren registrados en la aplicación pueden acceder introduciendo sus credenciales.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.2 y 15.2.2 Página: Login* del apartado *15. Prototipos.*

Desde esta ventana se accederá al Home de la aplicación si las credenciales proporcionadas son correctas.

18.1.3 Página: Registro de usuario

Es la ventana en la cual los usuarios consumidores pueden crear una cuenta en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.3 y 15.2.3 Página: Registro de usuario* del apartado *15. Prototipos.*

Desde esta ventana se accederá a la ventana Home si todos los datos son correctos.

18.1.4 Página: Registro de tienda

Es la ventana en la cual los comercios pueden crear una cuenta en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.4 y 15.2.4 Página: Registro de tienda* del apartado *15. Prototipos.*

Desde esta ventana se creará una cuenta de tienda y se accederá a la ventana Home si todos los datos son correctos.

18.1.5 Página: Home

Es la ventana principal de la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.5 y 15.2.5 Página: Home* del apartado *15. Prototipos.*

Desde esta ventana puede accederse a todas las funcionalidades de la aplicación a través de los menús.

18.1.6 Página: Productos

Página central de los productos de la aplicación desde la cual se puede ver un listado de los productos existentes, un buscador de productos así como añadir un nuevo producto.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.6 y 15.2.6 Página: Productos* del apartado *15. Prototipos.*

Sobre cada producto de la lista, podrán realizarse las acciones, en función de los permisos del perfil con el que se acceda: ampliar información del producto, editar producto, borrar producto, añadir o eliminar el producto de las notificaciones deseadas y añadir oferta.

Si sobre esta ventana se desea dar de alta una oferta sobre uno de los productos de la lista, mediante la acción Añadir Oferta, habiendo accedido con un perfil de tienda, se mostrará una ventana modal cuyos prototipos pueden verse en las secciones: *15.1.7 y 15.2.7 Ventana Modal: añadir oferta de producto* del apartado *15. Prototipos.*

Y tras completar los campos del formulario y pulsar el botón para añadir oferta, esta será añadida como oferta del producto en la tienda con la que se haya accedido.

18.1.7 Página: Ver producto

Página en la que se puede observar toda la información detallada acerca de un producto existente en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.8 y 15.2.8 Página: Ver Producto* del apartado *15. Prototipos.*

Desde esta ventana se puede, en función de los permisos del perfil con el que se acceda, Añadir el producto a notificaciones si no existe o si ya existe eliminarlo de notificaciones, Editar la información del producto o Borrar el producto.

18.1.8 Página: Añadir producto

Página desde la cual se puede añadir un nuevo producto a la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.9 y 15.2.9 Página: Añadir Producto* del apartado *15. Prototipos.*

18.1.9 Página: Editar producto

Página desde la cual se puede editar un producto existente en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.10 y 15.2.10 Página: Editar Producto* del apartado *15. Prototipos.*

18.1.10 Página: Ofertas

Página central de las ofertas de la aplicación desde la cual se puede ver un listado de las ofertas existentes, un buscador de ofertas así como añadir una nueva oferta.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.11 y 15.2.11 Página: Ofertas* del apartado *15. Prototipos.*

Sobre cada producto de la lista, podrán realizarse las acciones, en función de los permisos del perfil con el que se acceda: ampliar información de la oferta, editar oferta, borrar oferta y añadir o eliminar oferta a/de favoritos.

18.1.11 Página: Ver oferta

Página en la que se puede observar toda la información detallada acerca de una oferta existente en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.12 y 15.2.12 Página: Ver Oferta* del apartado *15. Prototipos.*

Desde esta ventana se puede, en función de los permisos del perfil con el que se acceda, Añadir la oferta a favoritos o eliminarla si ya se había añadido previamente, Editar la información de la oferta o Borrar la oferta.

18.1.12 Página: Añadir lista de ofertas

Página desde la cual se pueden añadir productos a la lista de ofertas de la tienda con la que se haya accedido.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.13 y 15.2.13 Página Añadir lista de ofertas* del apartado *15. Prototipos.*

Se seleccionará el producto a añadir en la lista de la izquierda y al pulsar el botón Añadir oferta, se mostrará la ventana modal cuyo prototipo puede verse en las secciones: *15.1.7 y 15.2.7 Ventana modal Añadir Oferta de un producto* del apartado *15. Prototipos.*

Y tras completar la información de la oferta y pulsar enviar, la oferta sobre el producto se mostrará en la lista de la derecha. Cuando se hayan introducido todas las ofertas deseadas en la lista de la derecha, al pulsar el botón Añadir nuevas ofertas, la lista de ofertas a dar de alta en la aplicación será enviada.

18.1.13 Página: Editar oferta

Página desde la cual se puede editar una oferta existente en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.14 y 15.2.14 Página: Editar oferta* del apartado *15. Prototipos.*

18.1.14 Página: Tiendas

Página central de las tiendas de la aplicación desde la cual se puede ver un listado de las tiendas existentes y un buscador de tiendas.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.15 y 15.2.15 Página: Tiendas* del apartado *15. Prototipos.*

Sobre cada tienda de la lista, podrán realizarse las acciones, en función de los permisos del perfil con el que se acceda: ampliar información de la tienda, borrar tienda y ver las ofertas de esa tienda.

18.1.15 Página: Ver tienda

Página en la que se puede observar toda la información detallada acerca de una tienda existente en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.16 y 15.2.16 Página: Ver Tienda* del apartado *15. Prototipos.*

18.1.16 Página: Cadenas de tiendas

Página central de las cadenas de tiendas de la aplicación desde la cual se puede ver un listado de las cadenas de tiendas existentes, un buscador de tiendas así como el acceso para crear una nueva cadena de tiendas.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.17 y 15.2.17 Página: Cadenas de tiendas* del apartado *15. Prototipos.*

Sobre cada tienda de la lista, podrán realizarse las acciones, en función de los permisos del perfil con el que se acceda: ampliar información de la cadena de tiendas, borrar cadena de tiendas y ver las tiendas pertenecientes esa cadena de tiendas.

18.1.17 Página: Ver cadena de tiendas

Página en la que se puede observar toda la información detallada acerca de una tienda existente en la aplicación, así como acceder a la ventana de edición de cadena de tiendas y borrar la cadena de tiendas.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.18 y 15.2.18 Página: Ver Cadena de tiendas* del apartado *15. Prototipos.*

18.1.18 Página: Añadir cadena de tiendas

Página desde la cual se puede añadir una nueva cadena de tiendas a la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.19 y 15.2.19 Página: Añadir cadena de tiendas* del apartado *15. Prototipos.*

18.1.19 Página: Editar cadena de tiendas

Página desde la cual se puede editar una cadena de tiendas existente en la aplicación.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.20 y 15.2.20 Página: Editar cadena de tiendas* del apartado *15. Prototipos.*

18.1.20 Página: Ver notificaciones

Página accesible a través del contador de notificaciones del panel derecho de la ventana, y muestra las notificaciones que existen actualmente para el usuario, acerca de productos que tenía seleccionados para notificar y de los cuales se ha añadido una nueva oferta.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.21 y 15.2.21 Página: Ver Cadena de tiendas* del apartado *15. Prototipos.*

Desde cada notificación es posible posible ver la oferta del producto, eliminar el producto de la lista de notificaciones deseadas y marcar la notificación como vista.

18.1.21 Página: Ver ofertas favoritas

Página accesible a través del icono de favoritos del panel derecho de la ventana, y muestra las ofertas favoritas para el usuario.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.22 y 15.2.22 Página: Ver ofertas favoritas* del apartado *15. Prototipos.*

Desde cada notificación es posible posible ver la oferta del producto y eliminar la oferta de la lista de favoritos.

18.1.22 Página: Ver perfil de cuenta de usuario

Página accesible a través del icono profile del panel derecho de la ventana, y los datos de la cuenta del usuario, así como un botón para acceder a la ventana de edición del perfil del usuario.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.23 y 15.2.23 Página: Ver perfil de cuenta de usuario* del apartado *15. Prototipos.*

18.1.23 Página: Editar perfil de cuenta de usuario

Página a través de la cual el usuario puede modificar sus datos de perfil.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.24 y 15.2.24 Página: Editar perfil de cuenta de usuario* del apartado *15. Prototipos.*

18.1.24 Página: Ver perfil de cuenta de tienda

Página accesible a través del icono profile del panel derecho de la ventana, y los datos de la cuenta de la tienda, así como un botón para acceder a la ventana de edición del perfil de la tienda.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.25 y 15.2.25 Página: Editar perfil de cuenta de usuario* del apartado *15. Prototipos.*

18.1.25 Página: Editar perfil de cuenta de tienda

Página a través de la cual el comercio puede modificar sus datos de perfil.

Pueden verse los prototipos de esta ventana en las secciones: *15.1.26 y 15.2.26 Página: Editar perfil de cuenta de tienda* del apartado *15. Prototipos.*

19. Seguridad

Para securizar la aplicación de alimentación sostenible se han implementado los siguientes mecanismos de seguridad:

19.1 Autenticación

Para la autenticación de los usuarios que hacen uso de la aplicación, se utiliza JWT, cuyo funcionamiento teórico se detalló en el apartado: *5.2.1 JWT.*
Para implementar este mecanismo de autenticación se ha desarrollado una clase filtro llamada *AuthenticationManager*, que es la encargada de comprobar que los datos del login introducidos por el usuario son correctos y, en caso afirmativo, construir el token JWT que identificará al usuario como autenticado durante toda la navegación y devolverlo a la capa cliente en las cabecera de respuesta: *Authorization*.

En la clase de configuración *WebSecurityConfig* se configuran los filtros personalizados para la realización de los mecanismos de autenticación y autorización y se indica que todas las peticiones deben ser autenticadas salvo las que cumplan ciertos patrones, que en el caso de la aplicación de alimentación sostenible son las páginas de login, de registro de tiendas y usuarios, el favicon, y los recursos estáticos css, js, html e imágenes.

De esta forma, todas las peticiones que se realicen a la capa servidora, salvo las que apunten a las operaciones de login y registro, necesitarán tener informado un token JWT válido en las cabeceras de la petición para poder ejecutarse sin errores, en caso contrario, se devolverá un error indicando que las credenciales no son correctas.

19.2 Autorización

Para la autorización de los usuarios para que puedan ejecutar únicamente el subconjunto de operaciones a las que deben tener acceso, se ha implementado un filtro para tal fin, *JWTAuthorizationFilter*, que se registra en la clase de configuración *WebSecurityConfig,* y que se encarga de interceptar todas las peticiones que se realizan al API REST para comprobar que contienen un token válido y establecer el rol del usuario.
Este rol es utilizado junto con la anotación *@PreAuthorize* para permitir o denegar la ejecución de las operaciones expuestas por el API REST en función del rol que tenga el usuario, obteniendo un error en el caso de que el usuario intente ejecutar una operación para la cual no tenga autorización.

Para las autorizaciones en la capa ciente de la aplicación de alimentación sostenible, se ha implementado un método en el API REST llamado */authorized*, encargado de comprobar si el usuario está autenticado en la aplicación y en caso afirmativo, devolver el rol que tiene el usuario. Esta información es utilizada en las distintas páginas web que forman la parte cliente de la aplicación, para, utilizando las directivas y condicionales de VueJS mostrar al usuario únicamente las opciones que está autorizado a ejecutar en función de su rol.

19.3 Acceso a páginas HTML por usuario sin autenticar

Como se ha indicado previamente, a nivel de *WebSecurityConfig* se ha establecido que se permita el acceso a los html de la aplicación sin que sea necesario un token JWT. Sin embargo, esto no era lo que a nivel de funcionamiento de la aplicación se desea, pues, a pesar de que no podría ejecutar ninguna petición a la capa servidora, no se desea ni siquiera que el usuario pueda llegar a ver las páginas de la aplicación sin estar autenticado en la misma. Para ello, en cada una de las ventanas, durante la fase *beforeCreate* del ciclo de vida establecido por VueJS, se realiza una petición al método del API */authorized* para comprobar si el usuario que realiza la petición se encuentra autenticado en la misma, y en caso negativo, se redirige la navegación a una página de error, impidiendo así que el usuario vea las páginas web de la aplicación sin esta autenticado en la misma.

19.4 Envío y almacenamiento del password de los usuarios

Para proteger el password de los usuarios, cuando éste es enviado desde la parte cliente a la parte servidora, por ejemplo durante las operaciones de login o de registro, lo que viaja realmente es una hash de la password de forma que su valor real se encuentre protegido y no viaje en ningún momento. Para esta operación se utiliza la librería javascript: bcrypt.js

En la parte servidora, se comprueba y almacena el valor de la hash de la password de los usuarios, de tal forma que en ningún caso se almacena el valor real de la password de los usuarios, quedando protegida ante un robo de información en la base de datos utilizada por la aplicación.

20. Tests

Para probar que el funcionamiento de la aplicación es el correcto, se han realizado pruebas de cada una de sus funcionalidades en relación a los diferentes perfiles con los que puede accederse a la aplicación y se ha obtenido la siguiente tabla con el resultado de cada una de las pruebas realizada:

Funcionalidad	Perfil	Resultado
Registrar nuevo usuario	Usuario	Satisfactorio
Obtener la información personal de una cuenta de usuario	Usuario	Satisfactorio
Editar información de usuario	Usuario	Satisfactorio
Registrar nueva tienda	Tienda	Satisfactorio
Obtener información de la cuenta de tienda	Tienda	Satisfactorio
Obtener lista de tiendas	Usuario Tienda Administrador	Satisfactorio
Buscar tienda	Usuario Tienda Administrador	Satisfactorio
Editar información de tienda	Tienda	Satisfactorio
Eliminar tienda	Administrador	Satisfactorio
Añadir un producto	Tienda Administrador	Satisfactorio
Obtener la lista de todos los productos	Usuario Tienda Administrador	Satisfactorio
Obtener producto	Usuario Tienda Administrador	Satisfactorio
Buscar producto	Usuario Tienda Administrador	Satisfactorio
Editar producto	Administrador	Satisfactorio
Eliminar producto	Administrador	Satisfactorio
Añadir una oferta	Tienda	Satisfactorio

Añadir una lista de ofertas	Tienda	Satisfactorio
Obtener la lista de todas las ofertas	Usuario Tienda Administrador	Satisfactorio
Obtener las ofertas de una tienda	Usuario Tienda Administrador	Satisfactorio
Obtener una oferta	Usuario Tienda Administrador	Satisfactorio
Buscar oferta	Usuario Tienda Administrador	Satisfactorio
Editar una oferta	Tienda	Satisfactorio
Eliminar una oferta	Tienda Administrador	Satisfactorio
Añadir una oferta a favoritos	Usuario	Satisfactorio
Eliminar una oferta de favoritos	Usuario	Satisfactorio
Obtener la lista de ofertas favoritas	Usuario	Satisfactorio
Añadir una cadena de tiendas	Administrador	Satisfactorio
Obtener una cadena de tiendas	Usuario Tienda Administrador	Satisfactorio
Obtener la lista de todas las cadenas de tiendas	Usuario Tienda Administrador	Satisfactorio
Obtener las tiendas pertenecientes a una cadena de tiendas	Usuario Tienda Administrador	Satisfactorio
Editar cadena de tiendas	Administrador	Satisfactorio
Eliminar cadena de tiendas	Administrador	Satisfactorio
Añadir notificación de un producto	Usuario	Satisfactorio
Obtener lista de notificaciones	Usuario	Satisfactorio
Eliminar notificación	Usuario	Satisfactorio

Tabla 3: Pruebas de funcionalidad realizadas

Además de comprobar que el funcionamiento de las diferentes funcionalidades de la aplicación se comportan correctamente, sería muy recomendable realizar un test de usabilidad con usuarios para comprobar la experiencia de usuario que la aplicación ofrece y el grado de satisfacción que ésta genera en los usuarios, sin embargo, esto se acometerá en un futuro, como se indica en el apartado: *26. Proyección a futuro*.

21. Versiones de la aplicación/servicio

Durante la realización del presente TFM se han ido realizando iteraciones sobre las tareas a realizar con objeto de implementar las diferentes funcionalidades de las que consta la aplicación de alimentación sostenible.

Así, podría decirse que la primera versión Beta de la aplicación fue aquella que se entregó junto con la PEC 3 del presente proyecto, y cuyo funcionamiento se mostró en el video que acompañaba a tal entrega.

Y, por tanto, la versión 1.0, es la que se entrega junto con la entrega final del presente TFM, quedando siempre abiertas las versiones para continuar añadiendo y mejorando nuevas funcionalidades tal y como se verá en el apartado: *26. Proyección a Futuro.*

22. Requisitos de instalación/implantación/uso

La aplicación web de alimentación sostenible puede ser desplegada en cualquier plataforma que ofrezca el despliegue de aplicaciones web, por ejemplo en cualquier plataforma de provisión de servicios en la nube, como AWS o Azure.

Sin embargo, en esta fase se propone trabajar con el propio jar generado vía `maven package`, el cual, al ser ejecutado a través del comando `java -jar` en cualquier ordenador que cuente con Java instalado que tenga correctamente establecidas las variables de entorno JAVA_HOME y PATH, autodespliega en la máquina la aplicación web de alimentación sostenible y se puede trabajar con ella de esta forma en cualquier parte.

23. Instrucciones de instalación/implantación

Junto con el presente documento se entrega un archivo .zip llamado *sustainable-application-with-data.zip* con la aplicación de alimentación sostenible. Para arrancar la aplicación web en cualquier ordenador que cuente con Java instalado, bastará con descomprimir el contenido del fichero en la carpeta que se desee, pongamos que se extraer en /home/marta/test, de forma que deberá quedar de la siguiente forma:

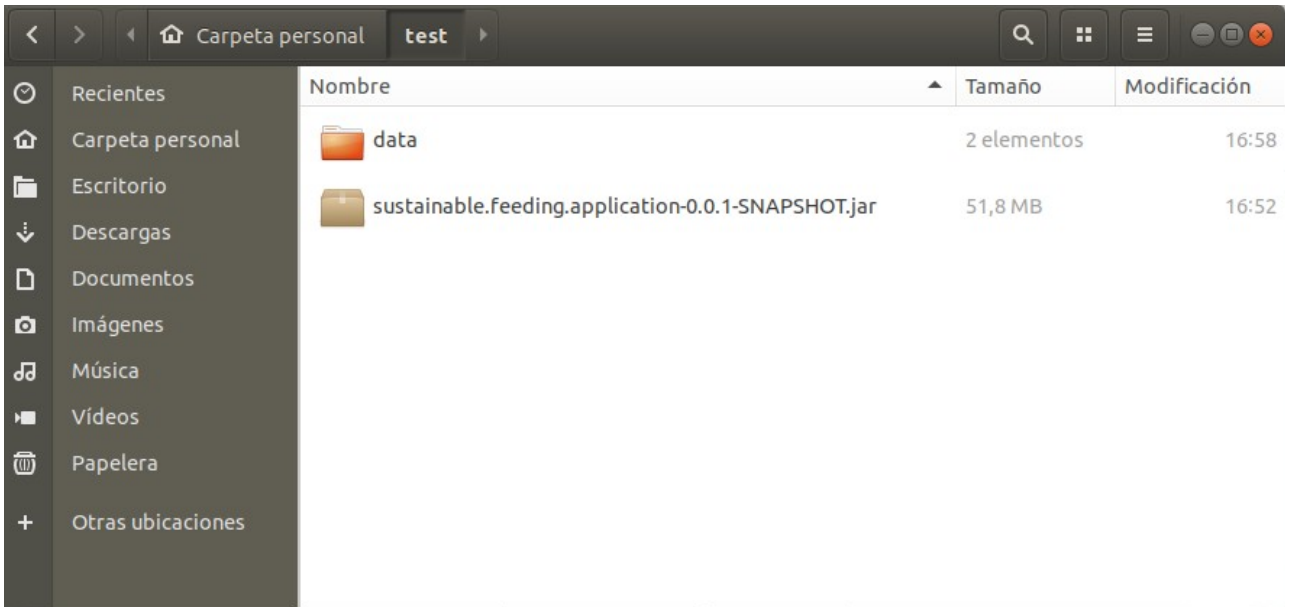


Figura 75: Descompresión de paquete de aplicación

Tras esta operación, se deberá abrir una ventana en línea de comandos o terminal, del sistema operativo en que vaya a realizarse la ejecución e introducir el comando:

```
java -jar sustainable.feeding.application-0.0.1-SNAPSHOT.jar
```

Y se observará que la aplicación comienza a arrancar:

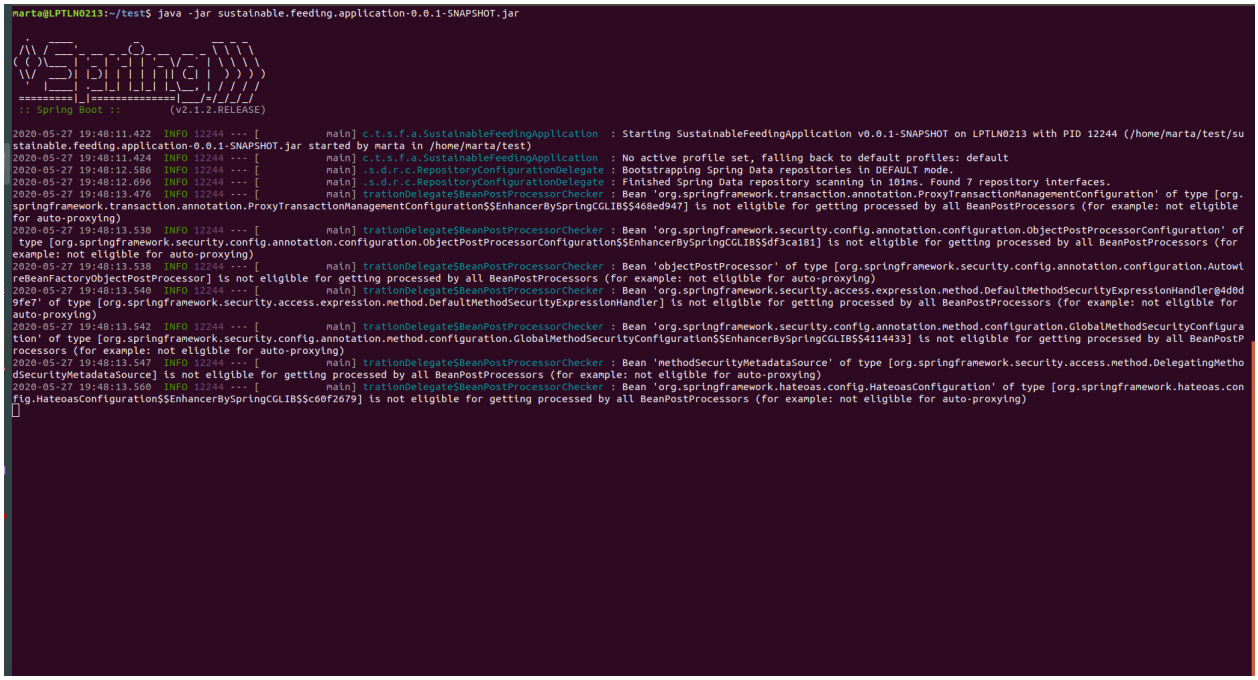


Figura 76: Inicio de arranque de la aplicación

Se considera que la aplicación se encuentra arrancada cuando en la pantalla se muestre el mensaje:

```
2020-05-27 19:48:22.030  INFO 12244 --- [           main] c.t.s.f.a.SustainableFeedingApplication  : Started SustainableFeedingApplication in 10.963 seconds (JVM running for 11.448)
```

A continuación, ya puede accederse a la aplicación web, para ello se debe de abrir el navegador que se prefiera e introducir en la barra de direcciones la siguiente url:
http://localhost:8080/initial.html

Y ya se estaría dentro de la aplicación web de alimentación sostenible, para más información acerca del uso de la aplicación puede consultarse el apartado: *24. Instrucciones de uso*.

24. Instrucciones de uso

Pueden encontrarse tanto la documentación del API REST como la guía de usuario de la aplicación en el apartado: *Anexo 5. Guía de usuario*

25. Bugs

Como en cualquier proyecto de desarrollo de software se han ido detectando errores y *bugs*, que se han ido corrigiendo en las diferentes iteraciones y entregas del producto, sin embargo, no se cuenta con un inventario de todos los errores detectados y corregidos en la aplicación, por lo que se describen a continuación únicamente algunos de los más relevantes:

- Error de integridad al eliminar un producto del cual existe una oferta
- Error de integridad al eliminar un producto añadido a las notificaciones de un usuario
- Error de integridad al eliminar una tienda que tiene dadas de alta ofertas
- Error de integridad al eliminar una oferta que está en la lista de favoritos de un usuario
- Al incluir los mecanismos de seguridad, en concreto el envío de token JWT, la petición para obtener la lista de cadenas de tiendas en el registro de una nueva tienda (cuando no hay aún autenticación) produce un error de permiso denegado
- Las peticiones de listado de ofertas realizadas por un usuario tienda, muestran todas las ofertas de todas las tiendas, en lugar de mostrar únicamente las ofertas dadas de alta por su tienda.
- El usuario administrador no podía acceder a la opción de borrado de producto.

26. Proyección a futuro

El trabajo futuro relativo a la aplicación de alimentación sostenible, desarrollada como objetivo del presente TFM, contará con las siguientes vías de trabajo:

- Despliegue y publicación de la aplicación en un dominio público, para ello, se deberá buscar un servicio de hosting o un proveedor de servicios en la nube que permitan el despliegue de una aplicación SpringBoot, así como su publicación y establecimiento de un dominio público accesible a través de Internet para todos los usuarios que lo deseen.

- A pesar de los mecanismos de seguridad que ya se han implementado en la versión actual de la aplicación de alimentación sostenible, en un futuro se llevará a cabo la Implementación de la utilización del protocolo HTTPS para proteger la integridad y la confidencialidad en el intercambio de datos entre la capa cliente y la capa servidora de la aplicación de alimentación sostenible.

- Realización de estudio de usabilidad y experiencia de usuario, para ello, se realizarán test de usuarios de la aplicación y los resultados obtenidos se utilizarán para mejorar la usabilidad y UX de las futuras versiones de la aplicación.

- Implementación de una versión móvil de la aplicación sostenible que permita a los usuarios interactuar con la misma desde sus *smartphones*.

- Desarrollo de un API público que se pondrá a disposición de los desarrolladores que deseen hacer uso del mismo que permita consultar las ofertas publicadas en la aplicación por cada comercio, de forma que esta funcionalidad pueda ser utilizada en otras aplicaciones web o aplicaciones móviles.

27. Presupuesto

Para el desarrollo de la aplicación de alimentación sostenible, dado que el equipo de desarrollo ha estado formado únicamente por una persona, la autora del presente TFM, y que, previamente a comenzar con el desarrollo del proyecto, ya se contaba con el equipamiento técnico que ha sido necesario para acometerlo, el presupuesto sería:

Teniendo en cuenta la planificación del proyecto que puede consultarse en el apartado: *11. Planificación* se calcula que se han dedicado un total de 111 días (sin contar el tiempo dedicado a la defensa del mismo). Suponiendo que, se ha trabajado en este proyecto, de media 4 horas al día, y que el precio de cada hora de trabajo se pagase a 12,5 euros, el coste total del desarrollo del proyecto de la aplicación de alimentación sostenible es: 111*4*12,5=5550 euros.

Días de trabajo	111 días (sin contar defensa)
Horas dedicadas por día de trabajo	4 horas/día
Precio por hora del desarrollador	12,5
Presupuesto aplicacion alimentación sostenible	5550 euros

Tabla 4: Presupuesto

28. Marketing y Ventas

Enfoque del trabajo en el plano de su branding y plan de promoción así como su política de precios y estrategia de venta.

29. Conclusión/-es

El presente TFM ha tenido como objetivo desarrollar un proyecto de desarrollo del software, en concreto, el desarrollo de una aplicación web, haciendo uso de las capacidades otorgadas por la realización del Máster Universitario en Ingeniería Informática, tales como: Ingeniería del software, Sistemas distribuidos, Gestión avanzada de proyectos, Ingeniería de la usabilidad, Dirección estratégica de sistemas de la información y Técnicas avanzadas de ingeniería del software.

Se ha elegido desarrollar una aplicación de alimentación sostenible debido a que actualmente vivimos en una época de derroche de recursos y consumismo que no puede mantenerse a lo largo del tiempo, y ya comienza a afectar a la calidad de vida que tenemos en nuestro planeta. Por este motivo, se ha pensado que podría aportarse un pequeño granito de arena para potenciar y facilitar tanto a los comercios como a los consumidores el consumo responsable de alimentos y el aprovechamiento de productos a punto de caducar que de otro modo se echarían a perder.

Las tecnologías y frameworks utilizados para la implementación de la aplicación de alimentación sostenible han permitido profundizar en el diseño y la creación de un API REST con los mecanismos de seguridad de autenticación y autorización basados en token JWT como se ha detallado en apartados anteriores, de manera sencilla y rápida, gracias a la utilización de SpringBoot. El uso del framework VueJS, ha permitido conocer las bondades de este conjunto de librerías y ponerlas en práctica, ya que en el momento en que se se comenzó este Trabajo Fin de Máster se desconocía su funcionamiento.

Para el desarrollo del proyecto a largo plazo, se han planteado líneas de trabajo relativas al despliegue de la misma como a mejoras en la seguridad, la usabilidad y la penetración en el mercado, que se acometerán en un futuro.

En resumen, el desarrollo de la aplicación web de alimentación sostenible ha sido enriquecedor pues ha permitido realizar un proyecto de ingeniería del software desde sus fases iniciales: planteamiento de la idea, análisis, selección de tecnologías a utilizar, diseño, implementación, pruebas, documentación, corrección de bugs e incidencias, hasta la fase final de entrega, por tanto, se ha tenido posibilidad de enfrentar todas las fases de trabajo de un proyecto, lo cual es muy positivo y supone un buen aprendizaje como punto final a la realización del Máster Universitario en Ingeniería Informática.

Anexo 1. Entregables del proyecto

Lista de archivos entregados y su descripción.

Anexo 2. Código fuente (extractos)

Anexo 2.1 Clase WebSecurityConfig

Esta clase es el lugar donde se configuran los diferentes aspectos de la seguridad de la aplicación Spring, y su código fuente se muestra a continuación:

```
package com.tfm.sustainable.feeding.application;

/*imports*/

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

	private UsuarioDetailsServiceImpl usuarioDetailsServiceImpl;

	public WebSecurityConfig(UsuarioDetailsServiceImpl userDetailsService) {
	this.usuarioDetailsServiceImpl = userDetailsService;
	}

	@Bean
	public PasswordEncoder bCryptPasswordEncoder() {
	return new BCryptPasswordEncoder();
	}

	@Override
	protected void configure(HttpSecurity http) throws Exception {

	http
			.csrf()
			.disable()
			.authorizeRequests()

	.antMatchers("/login","/user/registration","/store/registration","/
register/storechain/list","/favicon.ico","/h2/*","/*.html","/css/**", "/js/**",
"/images/**").permitAll()
			.anyRequest().authenticated().and()
```

```
                .addFilter(new
JWTAuthenticationFilter(authenticationManager()))
                .addFilter(new
JWTAuthorizationFilter(authenticationManager()))
                .headers().frameOptions().sameOrigin();
    }

    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception
{


auth.userDetailsService(usuarioDetailsServiceImpl).passwordEncoder(bCryptPasswor
dEncoder());
    }

}
```

Anexo 2.2 Clase JWTAuthenticationFilter

Esta clase es en donde se realiza el proceso de autenticación de la aplicación:

```
package com.tfm.sustainable.feeding.application.security;

/*imports*/

public class JWTAuthenticationFilter extends
UsernamePasswordAuthenticationFilter {

    private AuthenticationManager authenticationManager;

    public JWTAuthenticationFilter(AuthenticationManager
authenticationManager) {
    this.authenticationManager = authenticationManager;
    }


    @Override
```

```
	public Authentication attemptAuthentication(HttpServletRequest request,
HttpServletResponse response)
			throws AuthenticationException {
	try {
			LoginDTO credenciales = new
ObjectMapper().readValue(request.getInputStream(), LoginDTO.class);

			byte[] pass =
Base64.getDecoder().decode(credenciales.getPassword());
			String decodedString = new String(pass);
			return authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(
					credenciales.getLoginName(), decodedString, new
ArrayList<>()));
	} catch (IOException e) {
			throw new RuntimeException(e);
	}
	}

	@Override
	protected void successfulAuthentication(HttpServletRequest request,
HttpServletResponse response, FilterChain chain,
						Authentication auth) throws
IOException, ServletException {

	final String authorities = auth.getAuthorities().stream()
				.map(GrantedAuthority::getAuthority)
				.collect(Collectors.joining(","));


	String token = Jwts.builder().setIssuedAt(new Date())
				.setSubject(((User)auth.getPrincipal()).getUsername())
				.claim(JWTConstants.ROLE_CTE, authorities)
				.setExpiration(new Date(System.currentTimeMillis() +
JWTConstants.TOKEN_EXPIRATION_TIME))
				.signWith(SignatureAlgorithm.HS512,
JWTConstants.SUPER_SECRET_KEY).compact();
```

```
	response.addHeader(JWTConstants.HEADER_AUTHORIZACION_KEY,
JWTConstants.TOKEN_BEARER_PREFIX + " " + token);

	}
}
```

Anexo 2.3 Clase JWTAuthorization

Esta clase es en donde se realiza el proceso de autorización de la aplicación:

```
package com.tfm.sustainable.feeding.application.security;

/*imports*/

public class JWTAuthorizationFilter extends BasicAuthenticationFilter {

	public JWTAuthorizationFilter(AuthenticationManager authManager) {
	super(authManager);
	}

	@Override
	protected void doFilterInternal(HttpServletRequest req,
HttpServletResponse res, FilterChain chain)
		throws IOException, ServletException {
	String header = req.getHeader(JWTConstants.HEADER_AUTHORIZACION_KEY);
	if (header == null || !
header.startsWith(JWTConstants.TOKEN_BEARER_PREFIX)) {
		chain.doFilter(req, res);
		return;
	}

	UsernamePasswordAuthenticationToken authentication =
getAuthentication(req);
	if(authentication == null) {
		 res.sendError(HttpServletResponse.SC_UNAUTHORIZED);
		 return;
	}

	SecurityContextHolder.getContext().setAuthentication(authentication);
```

```
	req.setAttribute("user", (String)authentication.getPrincipal());
	chain.doFilter(req, res);
	}

	private UsernamePasswordAuthenticationToken
getAuthentication(HttpServletRequest request) {
	String token = request.getHeader(JWTConstants.HEADER_AUTHORIZACION_KEY);
	if (token != null) {

		// Se procesa el token y se recuperan usuario y role.
		final Claims claims = Jwts.parser()
			.setSigningKey(JWTConstants.SUPER_SECRET_KEY)

	.parseClaimsJws(token.replace(JWTConstants.TOKEN_BEARER_PREFIX, ""))
			.getBody();

		Date tokenDate = claims.getExpiration();
		long token_time = tokenDate.getTime()/1000;
		long current_time = new Date().getTime() / 1000;
		if ( token_time < current_time) {
		 	return null;
		}

		String user = claims.getSubject();

		final Collection<SimpleGrantedAuthority> authorities =

	Arrays.stream(claims.get(JWTConstants.ROLE_CTE).toString().split(","))
				.map(SimpleGrantedAuthority::new)
				.collect(Collectors.toList());

		if (user != null) {
			return new UsernamePasswordAuthenticationToken(user, null,
authorities);
		}
		return null;
	}
	return null;
	}
```

```
}
```

Anexo 2.4 Componente VueJS: Comprobación de que el usuario se encuentra autenticado

En este fragmento de código se muestra la petición GET que se realiza al método /authorized del API para comprobar si el usuario se encuentra autenticado en la aplicación y en caso afirmativo obtener su rol, y en caso negativo redireccionar a una página de error. Además, en el caso de que el usuario tenga como rol 'USER' se refrescarán las notificaciones pendientes.

```
const vm = new Vue({
  el: "#element",
  data: {
      role: "NONE",
      errors: [],
      notifications: 0,
      interval: ""
  },
  beforeCreate: function () {
      axios
      .get("http://localhost:8080/authorized", {
      headers: { Authorization: localStorage.getItem("Authorization") },
      })
      .then((response) => {
      this.role = response.data.data;
      if (this.role === "USER") this.refreshNotifications();
      })
      .catch(function (error) {
      let err = error.response.data.description
            ? error.response.data.description
            : error;
      location.href = "error.html?" + "error=" + encodeURIComponent(err);
      });
  }
...
```

Anexo 2.5 Renderización condicional en función del rol del usuario

En este fragmento de código se muestra la forma en que se renderiza el contenido de las ventanas de las aplicación en función del rol del usuario.

```
<div class="col-md-3 center">
   <a href="./userProfile.html" v-if="role === 'USER'">
  <svg class="bi bi-person-fill icons" width="1em" height="1em" viewBox="0 0 16
16"
fill="currentColor" xmlns="http://www.w3.org/2000/svg">
 <path fill-rule="evenodd"
d="M3 14s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1H3zm5-6a3 3 0 100-6 3 3 0 000 6z"
clip-rule="evenodd"/>
  </svg>
  <br>
  <p class="greenText">Profile
  <p></p></a>
   <a href="storeProfile.html" v-if="role === 'STORE'">
  <svg class="bi bi-building icons" width="1em" height="1em" viewBox="0 0 16 16"
fill="currentColor" xmlns="http://www.w3.org/2000/svg">
 <path fill-rule="evenodd" d="M15.285.089A.5.5 0 0115.5.5v15a.5.5 0 01-.5.5h-
3a.5.5 0 01-.5-.5V14h-1v1.5a.5.5 0 01-.5.5H1a.5.5 0 01-.5-.5v-6a.5.5 0
01.418-.493l5.582-.93V3.5a.5.5 0 01.324-.468l8-3a.5.5 0 01.46.057zM7.5
3.846V8.5a.5.5 0 01-.418.493l-5.582.93V15h8v-1.5a.5.5 0 01.5-.5h2a.5.5 0
01.5.5V15h2V1.222l-7 2.624z" clip-rule="evenodd"/>
 <path fill-rule="evenodd" d="M6.5 15.5v-7h1v7h-1z" clip-rule="evenodd"/>
 <path d="M2.5 11h1v1h-1v-1zm2 0h1v1h-1v-1zm-2 2h1v1h-1v-1zm2 0h1v1h-1v-1zm6-
10h1v1h-1V3zm2 0h1v1h-1V3zm-4 2h1v1h-1V5zm2 0h1v1h-1V5zm2-2h1v1h-1V3zm2
0h1v1h-1V3zm-4 0h1v1h-1V3zm0 2h1v1h-1V5zm2 0h1v1h-1V5zm2-2h1v1h-1V3zm-4
2h1v1h-1v-1zm2 0h1v1h-1v-1zm2 0h1v1h-1v-1z"/>
  </svg>
  <br>
  <p class="greenText">Profile
  <p></p></a><br>
</div>
```

Anexo 2.6 Controlador: Operaciones asociadas a una cuenta de usuario

A continuación se muestra el controlador de usuario, a través del cual se ejecutan las diferentes operaciones asociadas a la cuenta de usuario. Se publica el código fuente de este controlador a modo de ejemplo, pues el resto de los controladores de la aplicación son similares, siempre adaptados a su ámbito de uso.

Cabe destacar la anotación @PreAuthorize("hasRole('ROLE_USER')"), que es la que, junto con los mecanismos de seguridad implementados y descritos en el apartado: *19. Seguridad*, realiza el control de acceso a las operaciones en función del rol de los usuarios, en este caso solo se dejará ejecutar la operación a aquellos usuarios con el rol 'USER'.

```
package com.tfm.sustainable.feeding.application.user.controller;

/* imports */

@RestController
@Slf4j
@Api(value = "API to register, edit or delete users", tags = "USER CONTROLLER")
public class UserController {

	private final UserService userService;
	private final LoginService loginService;

	public UserController(UserService userService, LoginService loginService)
{
	this.userService = Objects.requireNonNull(userService);
	this.loginService = Objects.requireNonNull(loginService);
}

@ApiOperation(httpMethod = "POST", value = "Register a new user", response =
ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "The registered user",
response = ResponseBase.class),@ApiResponse(code = 500, message = "Error on the
server side"),@ApiResponse(code = 400, message = "Bad request error")
})
@PostMapping("/user/registration")
public ResponseEntity<ResponseBase<UserPersonalInfoDTO>>
userRegistration(@ApiParam(value = "User information to be registered in the
application", required = true)@RequestBody UserRegistrationDTO userDTO) {
```

```
	userDTO.getLoginDTO().setRole("USER");
	UserPersonalInfoDTO result =
UserMapper.convertToUserInfoDTO(userService.create(UserMapper.convertToUserRegis
trationEntity(userDTO)));
	ResponseBase<UserPersonalInfoDTO> responseBase = new
ResponseBase<>(ResponseCode.OK, result);
	return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "GET", value = "Get the personal information about a
user by its id", response = ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "The user personal
information", response = ResponseBase.class),@ApiResponse(code = 500, message =
"Error on the server side"),@ApiResponse(code = 400, message = "Bad request
error")
})
@GetMapping("/user/personalInfo")
public ResponseEntity<ResponseBase<UserPersonalInfoDTO>>
findUserInfoById(HttpServletRequest request) {

	String usuario = (String) request.getAttribute(("user"));
	Login login = loginService.findByUsername(usuario);
	User user = userService.findUserByLoginId(login.getIdLogin());
	UserPersonalInfoDTO userPersonalInfoDTO =
UserMapper.convertToUserInfoDTO(user);
	ResponseBase<UserPersonalInfoDTO> responseBase = new
ResponseBase<>(ResponseCode.OK, userPersonalInfoDTO);
	return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "GET", value = "Get all the information about a user
by its id", response = ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "All the user
information", response = ResponseBase.class),@ApiResponse(code = 500, message =
"Error on the server side"),@ApiResponse(code = 400, message = "Bad request
error")
})
```

```
@GetMapping("/user/{userid}")
public ResponseEntity<ResponseBase<UserDTO>> findUserById(@ApiParam(value = "The
id of the user", required = true) Integer userId) {
	User user = userService.findUserById(userId);
	UserDTO userDTO = UserMapper.convertToUserDTO(user);
	ResponseBase<UserDTO> responseBase = new ResponseBase<>(ResponseCode.OK,
userDTO);
	return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "POST", value = "Edit an existing user personal
information with the specified user information", response =
ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "The edited user",
response = ResponseBase.class),@ApiResponse(code = 500, message = "Error on the
server side"),@ApiResponse(code = 400, message = "Bad request error")
})
@PostMapping("/user/personalInfo/edit")
public ResponseEntity<ResponseBase> updateUserInfo(@ApiParam(value = "The new
information to set in the existing user", required = true)@RequestBody
UserPersonalInfoDTO userPersonalInfoDTO, HttpServletRequest request) {
	String usuario = (String) request.getAttribute(("user"));
	Login login = loginService.findByUsername(usuario);
	User user = userService.findUserByLoginId(login.getIdLogin());
	User userInfo = UserMapper.convertToUserInfoEntity(userPersonalInfoDTO);
	userInfo.setIdUser(user.getIdUser());
	userService.editPersonalInfo(userInfo);

	ResponseBase responseBase = new ResponseBase<>(ResponseCode.OK);
	return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "POST", value = "Change the password of the User
account", response = ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "Successful
operation", response = ResponseBase.class),@ApiResponse(code = 500, message =
```

```
"Error on the server side"),@ApiResponse(code = 400, message = "Bad request
error")
})
@PostMapping("/user/personalInfo/{userid}/password/change")
public ResponseEntity<ResponseBase> updateLoginInfo(@ApiParam(value = "The id of
the user", required = true) Integer userid,@ApiParam(value = "The last
password", required = true) String lastPassword,@ApiParam(value = "The new
password", required = true) String newPassword) {
      userService.updatePassword(userid, lastPassword, newPassword);
      ResponseBase responseBase = new ResponseBase<>(ResponseCode.OK, null);
      return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER') OR hasRole('ROLE_ADMIN')")
@ApiOperation(httpMethod = "POST", value = "Delete an existing user by its id",
response = ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "Successful
operation", response = ResponseBase.class),@ApiResponse(code = 500, message =
"Error on the server side"),@ApiResponse(code = 400, message = "Bad request
error")
})
@PostMapping("/user/delete/{userid}")
public ResponseEntity<ResponseBase> deleteUser(@ApiParam(value = "The id of the
user", required = true) Integer userid) {
      userService.deleteById(userid);
      ResponseBase responseBase = new ResponseBase<>(ResponseCode.OK, null);
      return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "GET", value = "Get all the user's favorite offers",
response = ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "All the user's
favorite offers", response = ResponseBase.class),@ApiResponse(code = 500,
message = "Error on the server side"),@ApiResponse(code = 400, message = "Bad
request error")
})
@GetMapping("/user/favorites")
```

```
public ResponseEntity<ResponseBase<List<OfferDTO>>>
findUserFavorites(HttpServletRequest request) {

	String usuario = (String) request.getAttribute(("user"));
	Login login = loginService.findByUsername(usuario);
	User user = userService.findUserByLoginId(login.getIdLogin());
	List<OfferDTO> userFavoritesDTO =
OfferMapper.convertEntityListToDTOList(user.getFavoriteOffers(),user.getFavorite
Offers());
	ResponseBase<List<OfferDTO>> responseBase = new
ResponseBase<>(ResponseCode.OK, userFavoritesDTO);
	return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "POST", value = "Add an offer to the user's favorite
list", response = ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "The offer added to
user's favorites", response = ResponseBase.class),@ApiResponse(code = 500,
message = "Error on the server side"),@ApiResponse(code = 400, message = "Bad
request error")
})
@PostMapping("/user/favorites/add/{offerid}")
public ResponseEntity<ResponseBase> addFavorite(@ApiParam(value = "The id of the
offer to be added", required = true) @PathVariable("offerid") Integer offerid,
HttpServletRequest request) {

	String usuario = (String) request.getAttribute(("user"));
	Login login = loginService.findByUsername(usuario);
	User user = userService.findUserByLoginId(login.getIdLogin());
	userService.addFavorite(user, offerid);
	ResponseBase responseBase = new ResponseBase<>(ResponseCode.OK, null);
	return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "POST", value = "Delete an offer from the user's
favorite list", response = ResponseEntity.class)
```

```
@ApiResponses(value = {@ApiResponse(code = 200, message = "Successful
operation", response = ResponseBase.class),@ApiResponse(code = 500, message =
"Error on the server side"),@ApiResponse(code = 400, message = "Bad request
error")
})
@PostMapping("/user/favorites/delete/{offerid}")
public ResponseEntity<ResponseBase> deleteFavorite(@ApiParam(value = "The id of
the offer to be added", required = true) @PathVariable("offerid") Integer
offerid, HttpServletRequest request) {

      String usuario = (String) request.getAttribute(("user"));
      Login login = loginService.findByUsername(usuario);
      User user = userService.findUserByLoginId(login.getIdLogin());
      userService.deleteUserFavorite(user, offerid);
      ResponseBase responseBase = new ResponseBase<>(ResponseCode.OK, null);
      return ResponseEntity.ok(responseBase);
}

@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "POST", value = "Add Notification", response =
ResponseEntity.class)
@ApiResponses(value = {@ApiResponse(code = 200, message = "The added
notification", response = ResponseBase.class),@ApiResponse(code = 500, message =
"Error on the server side"),@ApiResponse(code = 400, message = "Bad request
error")
})
@PostMapping("/user/notification/add/{productid}")
public ResponseEntity<ResponseBase> addNotification(@ApiParam(value =
"Notification information to add", required = true) @PathVariable("productid")
Integer productid, HttpServletRequest request) {

      String usuario = (String) request.getAttribute(("user"));
      Login login = loginService.findByUsername(usuario);
      User user = userService.findUserByLoginId(login.getIdLogin());
      userService.addNotification(user, productid);
      ResponseBase responseBase = new ResponseBase<>(ResponseCode.OK, null);
      return ResponseEntity.ok(responseBase);

}
```

```
@PreAuthorize("hasRole('ROLE_USER')")
@ApiOperation(httpMethod = "GET", value = "Get a user's notification list",
response = ResponseEntity.class) @ApiResponses(value = {@ApiResponse(code = 200,
message = "The user's notification list", response =
ResponseBase.class),@ApiResponse(code = 500, message = "Error on the server
side"),@ApiResponse(code = 400, message = "Bad request error")
})
@GetMapping("/user/product/notification/list")
public ResponseEntity<ResponseBase<List<ProductDTO>>>
listAllUserProductsForNotification(HttpServletRequest request) {
      String usuario = (String) request.getAttribute(("user"));
      Login login = loginService.findByUsername(usuario);
      User user = userService.findUserByLoginId(login.getIdLogin());
      List<ProductDTO> productDTOList =
ProductMapper.convertEntityListToDTOList(user.getNotificationList(),user.getNoti
ficationList());
      ResponseBase<List<ProductDTO>> responseBase = new
ResponseBase<>(ResponseCode.OK, productDTOList);
      return ResponseEntity.ok(responseBase);
}

@ApiOperation(httpMethod = "POST", value = "delete a user's notification",
response = ResponseEntity.class)
@PostMapping("/user/notification/delete/{productid}")
public ResponseEntity<ResponseBase<NotificationDTO>>
deleteNotification(@ApiParam(value = "The id of the product of the
notification", required = true) @PathVariable("productid") Integer productid,
HttpServletRequest request) {
      String usuario = (String) request.getAttribute(("user"));
      Login login = loginService.findByUsername(usuario);
      User user = userService.findUserByLoginId(login.getIdLogin());
      userService.deleteNotification(user, productid);
      ResponseBase<NotificationDTO> responseBase = new
ResponseBase<>(ResponseCode.OK, null);
      return ResponseEntity.ok(responseBase);
      }
}
```

Anexo 3. Librerías/Código externo utilizado

Información detallada acerca de qué librerías, código, archivos, y cualquier otra herramienta tecnológica desarrollada por terceros utilizada en el trabajo, y qué partes de los mismos han sido usadas y cómo.

Anexo 4. Capturas de pantalla

Las capturas de pantalla del presente trabajo se han mostrado en cada uno de los apartados correspondientes.

Anexo 5. Guía de usuario

El API REST de la aplicación web de alimentación sostenible tiene una documentación “viva” realizada a través de Swagger, cuya información a nivel genérico puede consultarse a continuación:

Anexo 5.1 Documentación del API REST

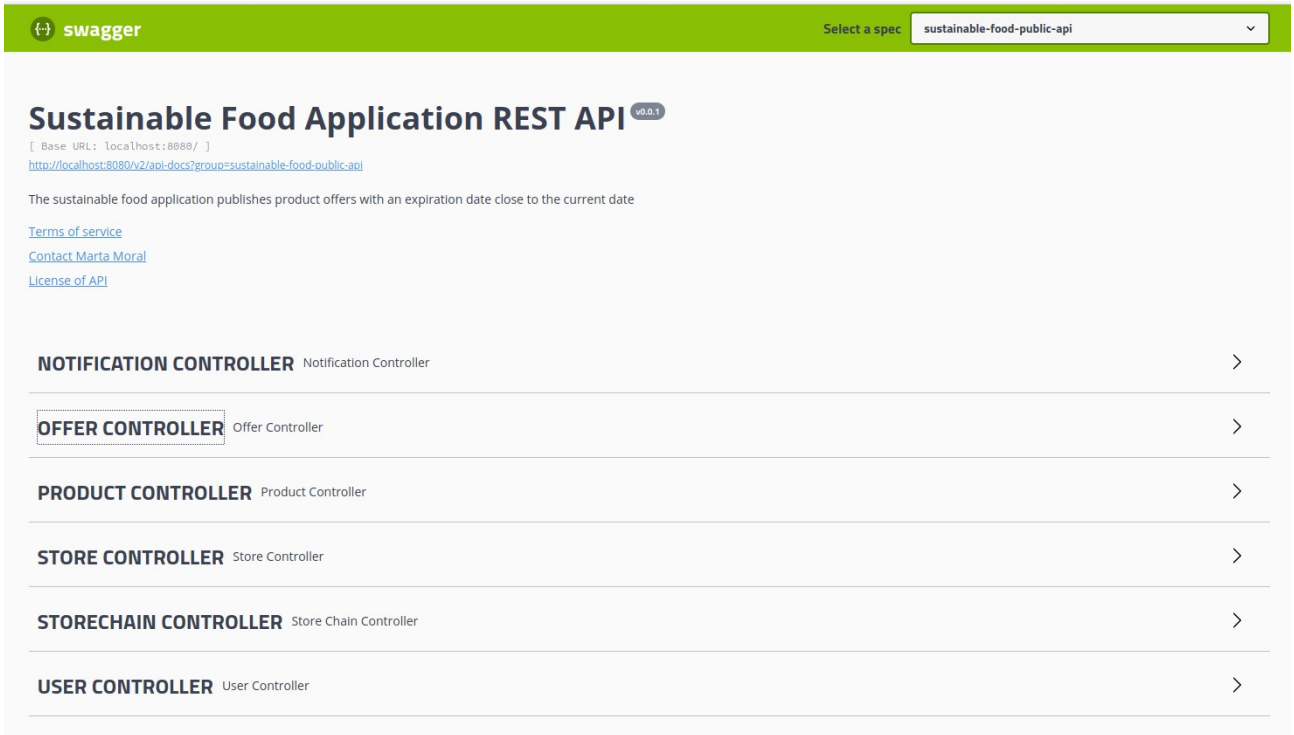


Figura 77: Documentación API Swagger: Controladores

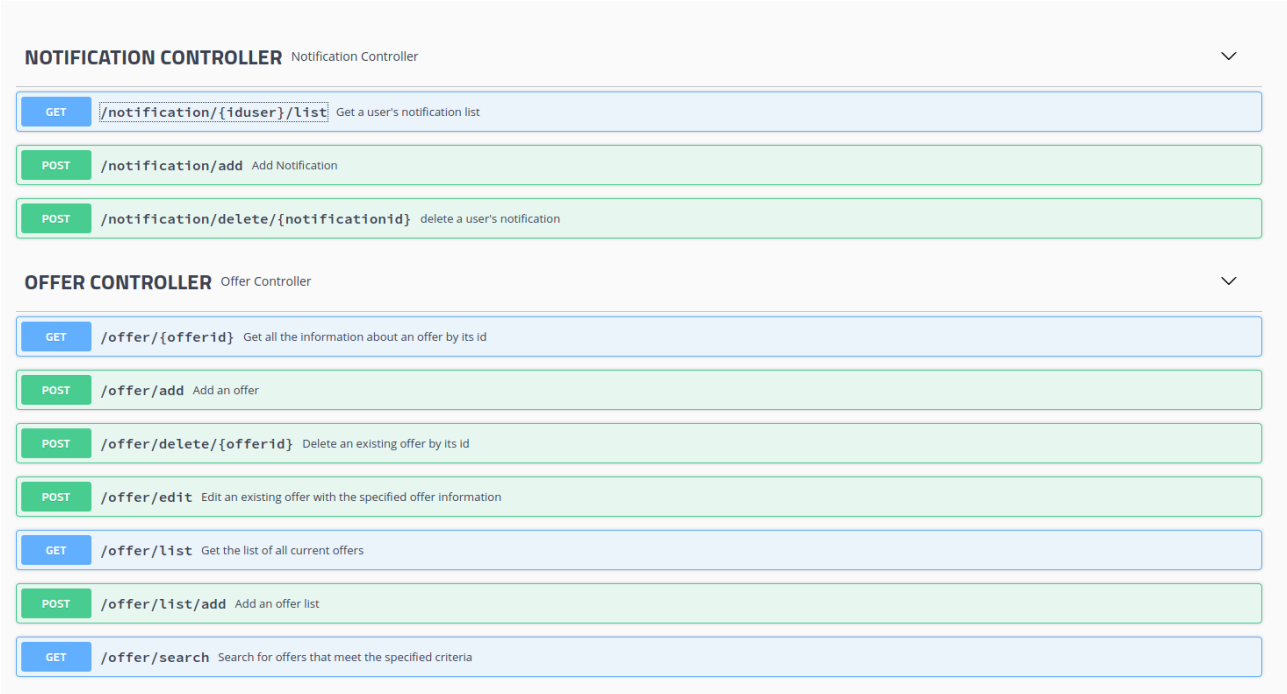


Figura 78: Documentación API Swagger: Notificaciones y oferta

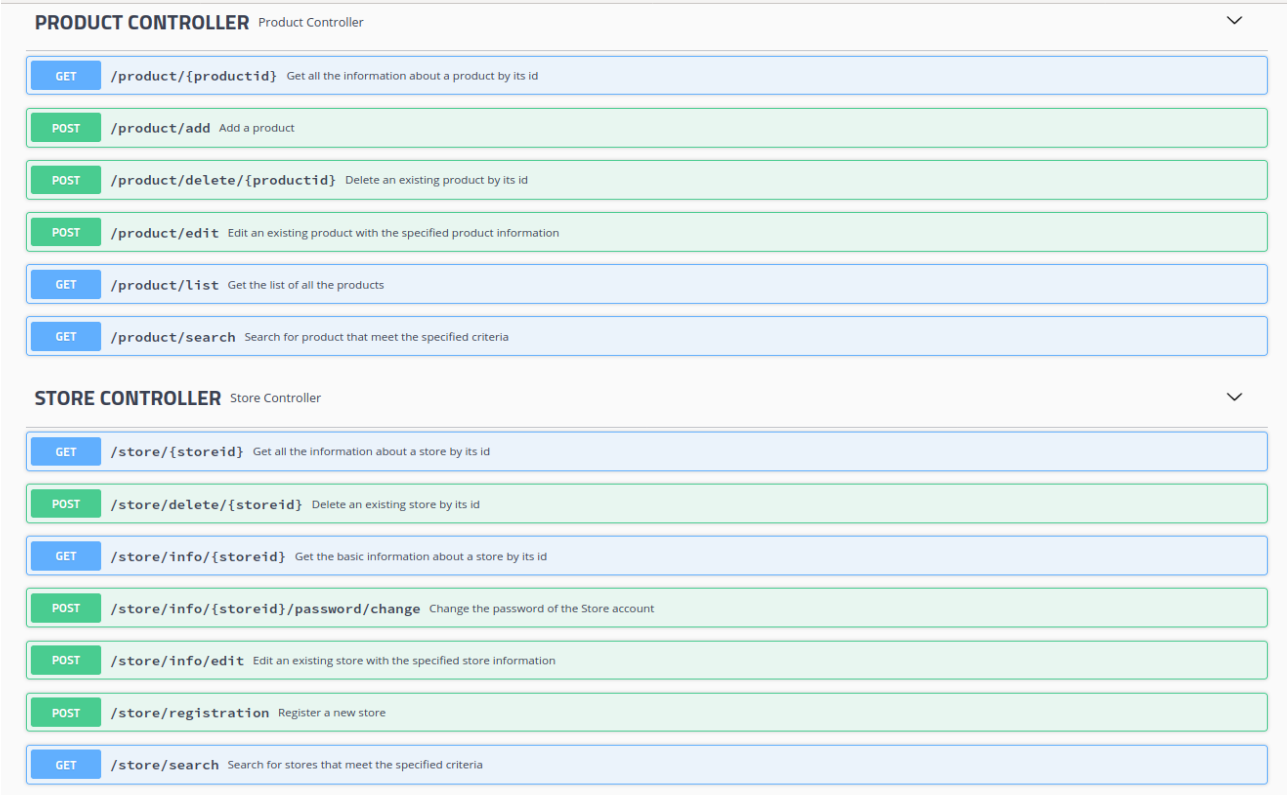


Figura 79: Documentación API Swagger: Producto y tienda

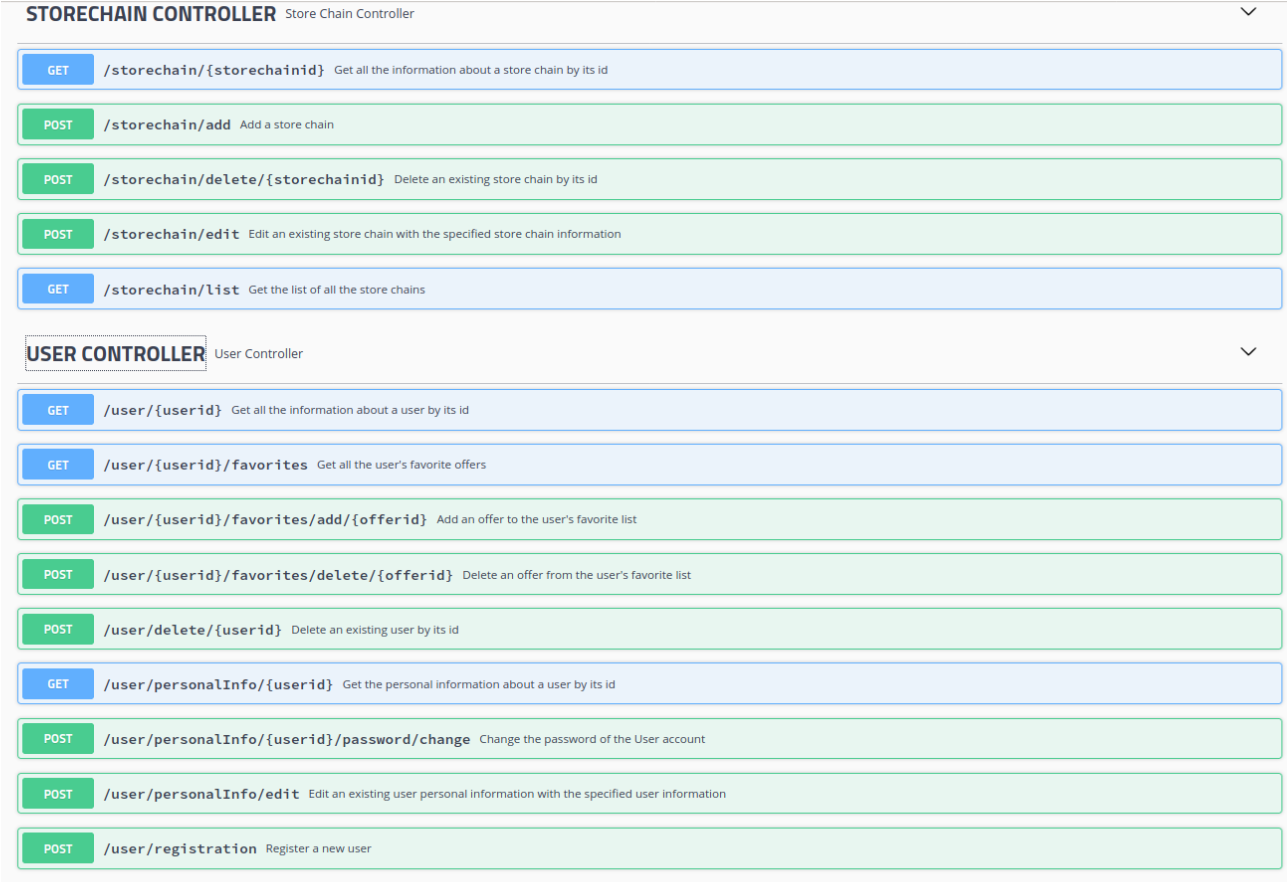


Figura 80: Documentación API Swagger: Cadena de tiendas y usuario

Puede encontrarse información detallada acerca de cada uno de los servicios en la ui de Swagger, que se encuentra (una vez arrancado el proyecto de Spring Boot), en la siguiente URL: http://localhost:8080/swagger-ui.html

Anexo 5.2 Guía de usuario

Para acceder a la aplicación de alimentación sostenible introduzca la dirección: http://localhost:8080/initial.html en el navegador de su ordenador, y podrá observar la siguiente ventana:

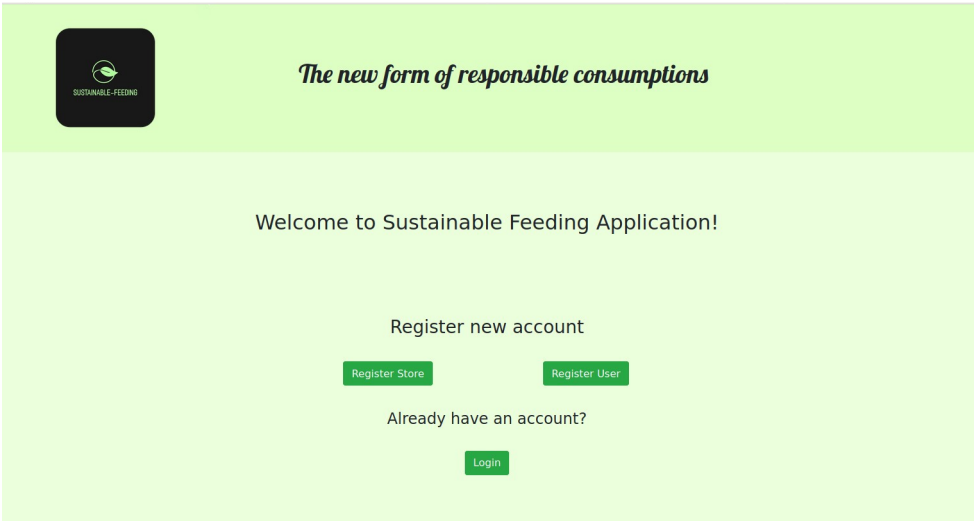


Figura 81: Ventana inicial

Si desea darse de alta en la aplicación como un comercio, pulse el botón “Register Store”, mientras que si desea darse de alta como usuario pulse el botón “Register User”. Ambas opciones le llevarán a los formularios de registro que se muestran a continuación:

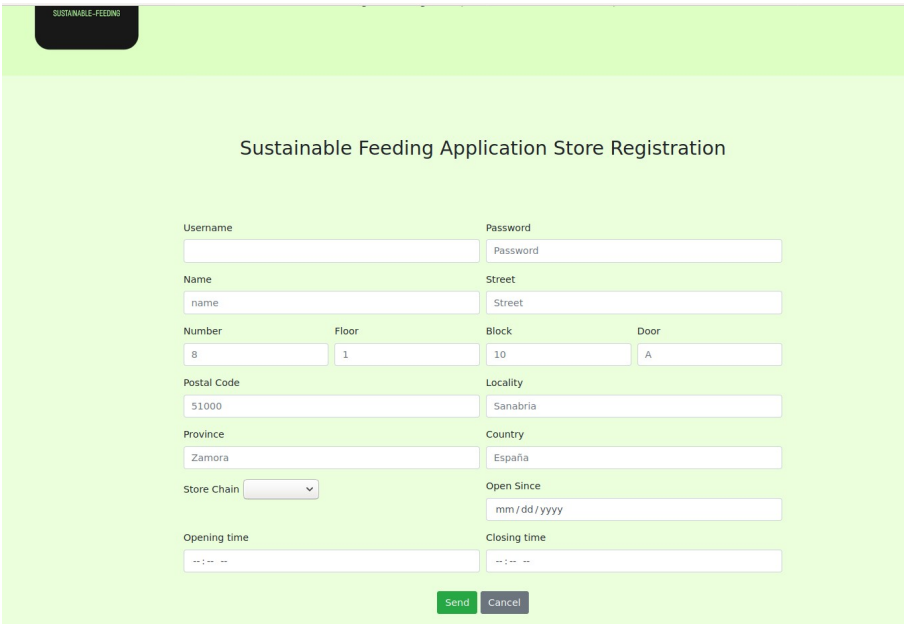


Figura 82: Ventana registro de comercio

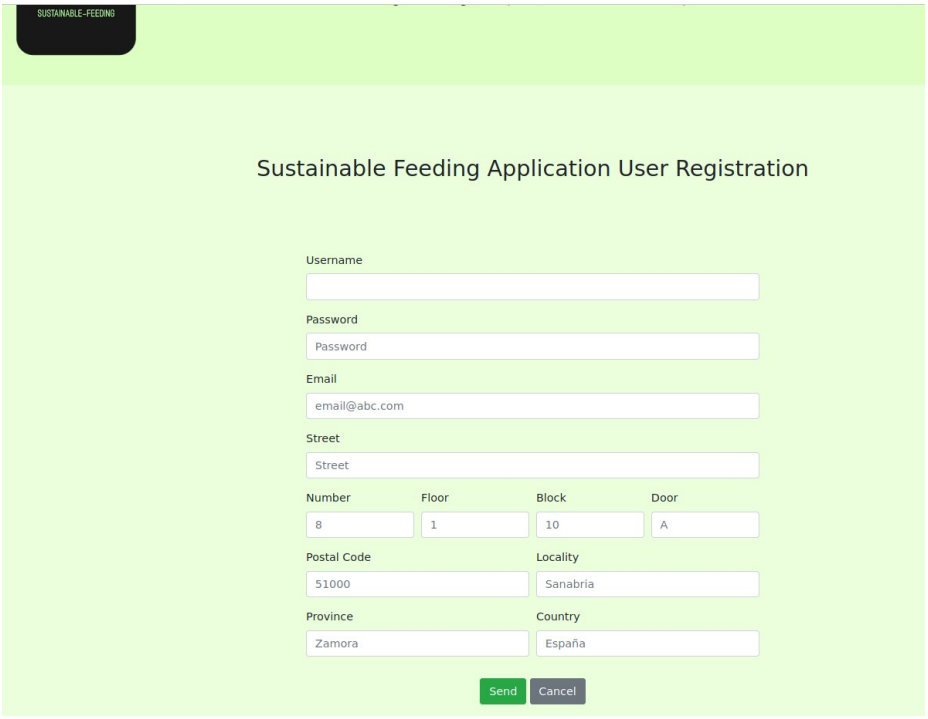


Figura 83: Ventana registro de usuario

Si ya tiene cuenta en la aplicación, basta con pulsar el botón “Login”, mostrándose la siguiente ventana, en la cual únicamente tendrá que añadir sus credenciales.

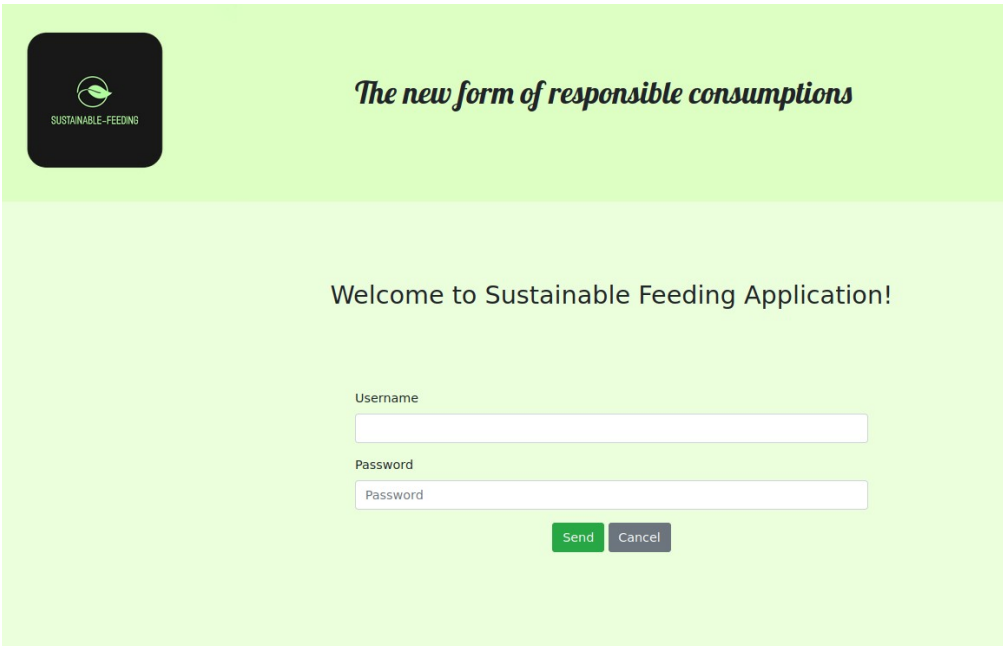


Figura 84: Ventana registro de usuario

Si las credenciales introducidas no han sido correctas, se mostrará un mensaje de error, mientras que si han sido correctas, se mostrará la ventana principal de la aplicación.

Las opciones disponibles para los comercios y para los usuarios son diferentes, por lo que se mostrarán primero las opciones disponibles para los usuarios y posteriormente para los comercios.

Anexo 5.2.1 Opciones disponibles para usuarios

Opciones de la cabecera superior

A continuación se muestra la página home cuando se accede a la aplicación con una cuenta de usuario.

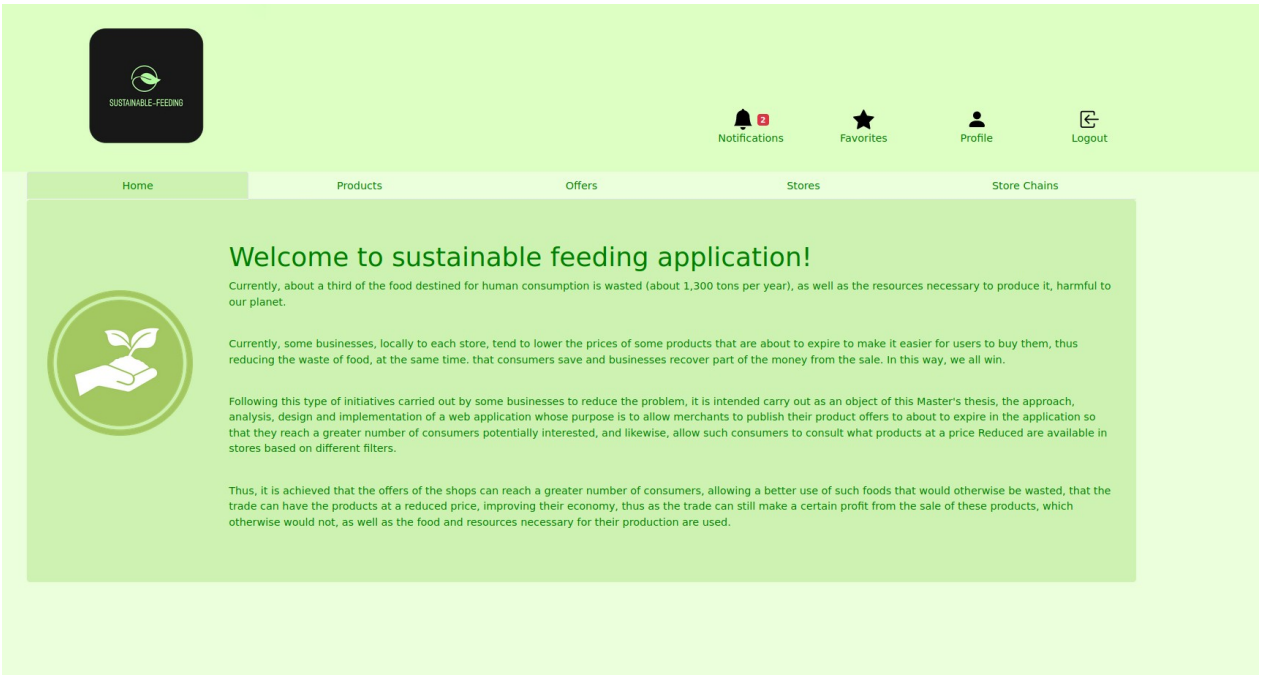


Figura 85: Ventana home para usuario

En la cabecera superior pueden observarse las operaciones de Notificaciones, Favoritos, Perfil y Salida de la aplicación.

Si se desean ver las notificaciones pendientes, basta con pulsar el icono de la campana, que le conducirá a la siguiente ventana:

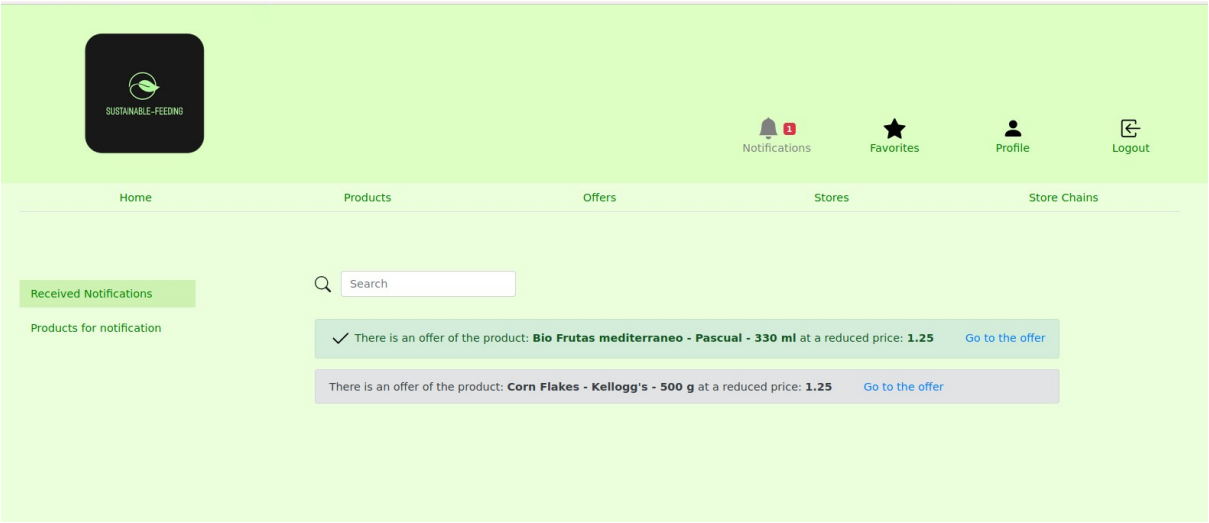


Figura 86: Ventana notificaciones de usuario

En la cual, aparecerán las notificaciones existentes acerca de ofertas de productos marcados como deseados para recibir notificaciones de los mismos. Las primeras notificaciones que se muestran serán aquellas que aún no han sido leídas, seguidas de aquellas que ya han sido leídas. Para marcar como leída una notificación se debe seleccionar el icono con el *Check* a la derecha de cada notificación.
Desde cada una de las notificaciones es posible acceder al detalle de cada oferta pulsando sobre el enlace "*Go to the offer*".

Si en la lista de la izquierda se selecciona la opción "*Products to notification*" se podrán ver y gestionar la lista de productos de los que se desea recibir notificaciones:

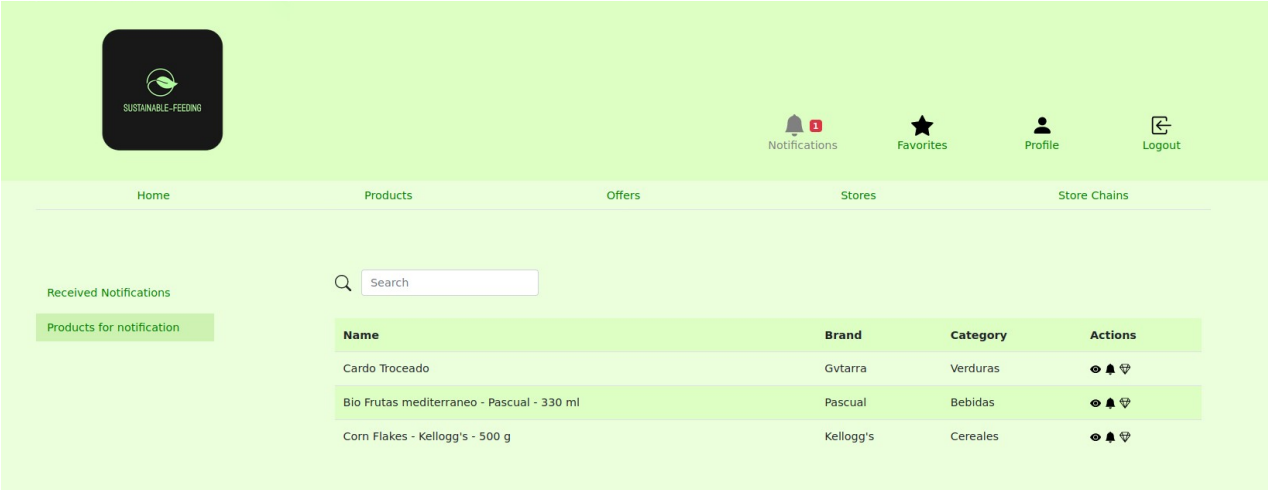


Figura 87: Ventana lista de productos de los que se desean notificaciones

Desde cada elemento de la lista de productos puede accederse a la ventana de detalle del producto, seleccionando el icono del ojo, eliminar el producto como deseado para recibir notificaciones del mismo, pulsando icono de la campana, y acceder a las ofertas sobre tal producto pulsando el icono del diamante.

Si se desea ver las ofertas guardadas como favoritas, se debe seleccionar el icono de la estrella en la cabecera superior de la página, accediendo así a la siguiente ventana:

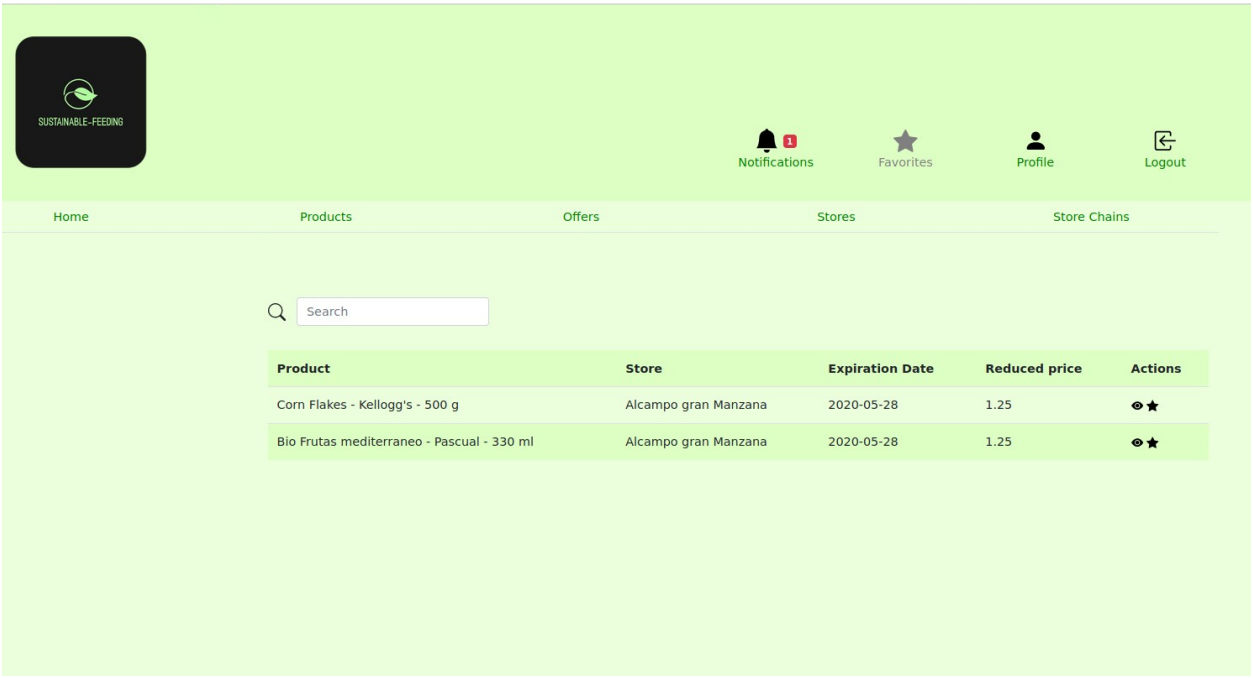


Figura 88: Ventana ofertas favoritas

En cada oferta de la lista puede accederse a ver el detalle de la oferta pulsando sobre el icono del ojo, y eliminar la oferta de favoritos pulsando el icono de la estrella.

Si se desea ver los datos de perfil del usuario, se debe seleccionar el icono del usuario en la cabecera superior de la página, accediendo así a la siguiente ventana:

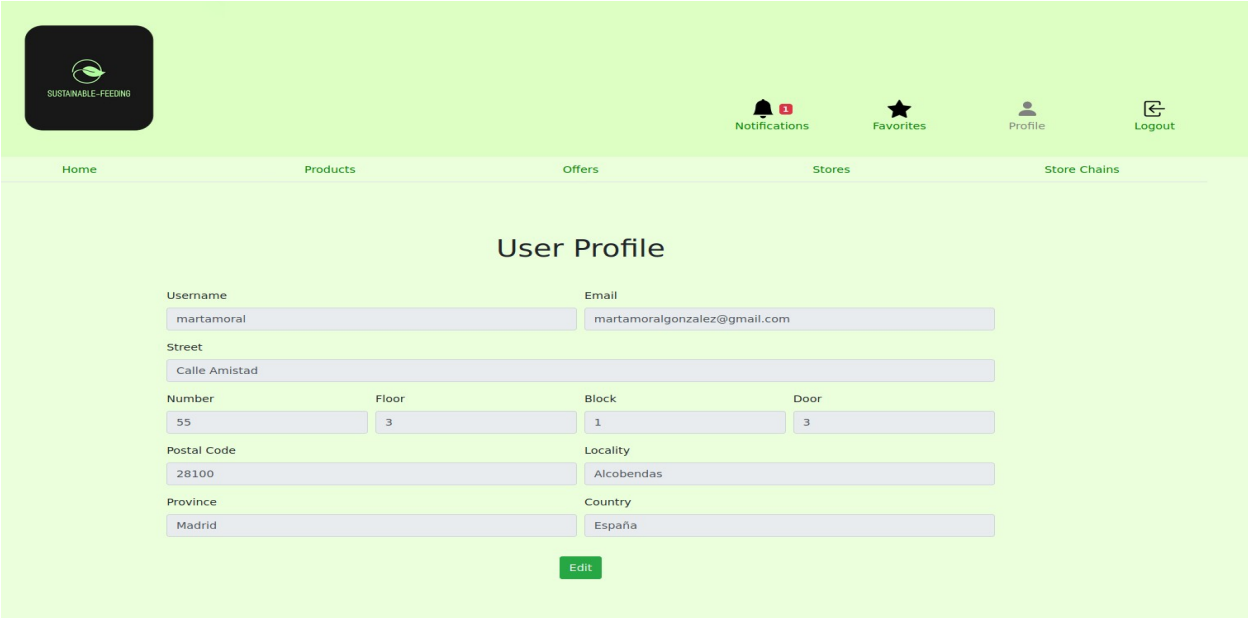


Figura 89: Ventana perfil de usuario

Si se quisiera modificar alguno de los datos de perfil, basta con seleccionar el botón "*Edit*", llegandose a la ventana para editar datos de perfil:

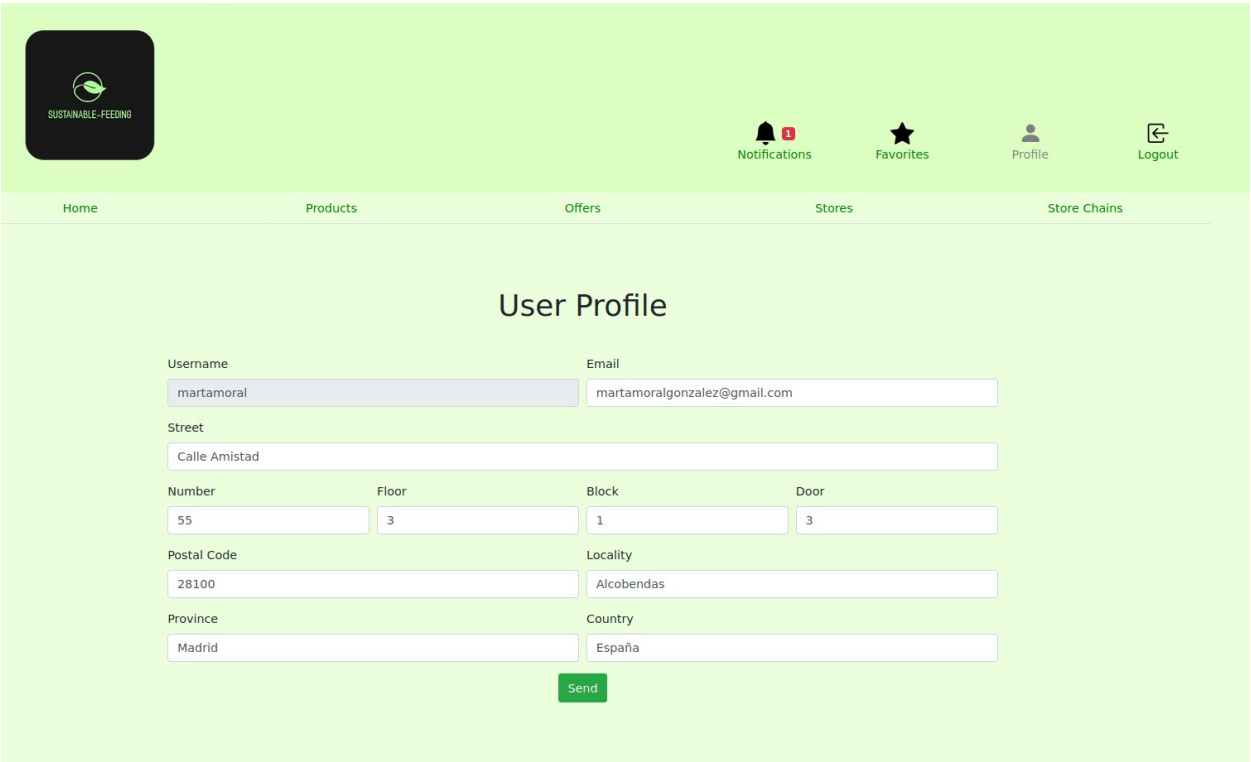


Figura 90: Ventana editar perfil de usuario

Menú central: Productos

Si se desea acceder a la información de los productos dados de alta en la aplicación, se deberá seleccionar "*Products*" en el menú central, accediendo a la siguiente ventana:

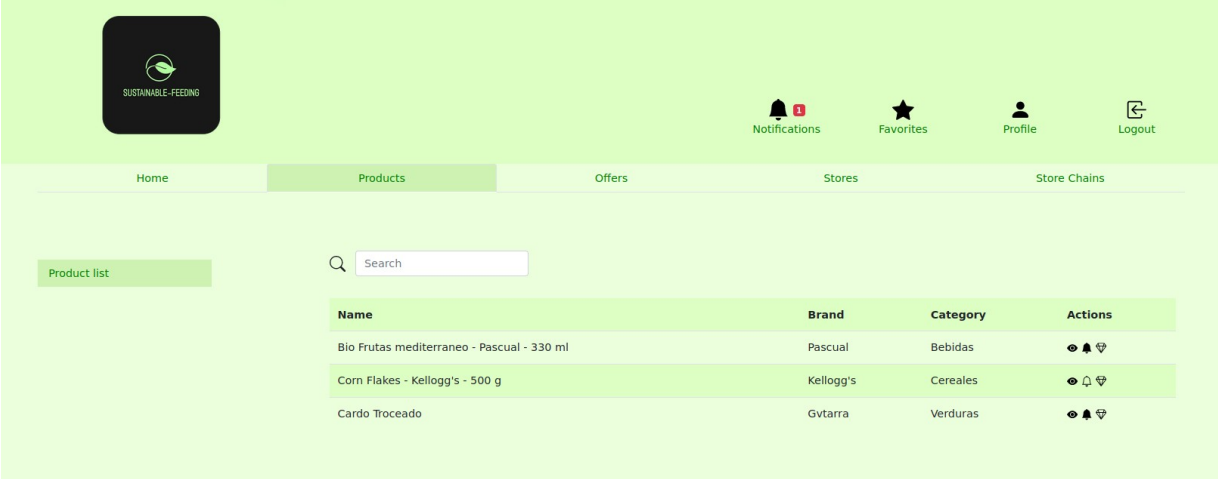


Figura 91: Ventana lista de productos para usuario

Para ver información ampliada de cada producto, se debe pulsar sobre el icono del ojo, llegandose a la siguiente ventana:

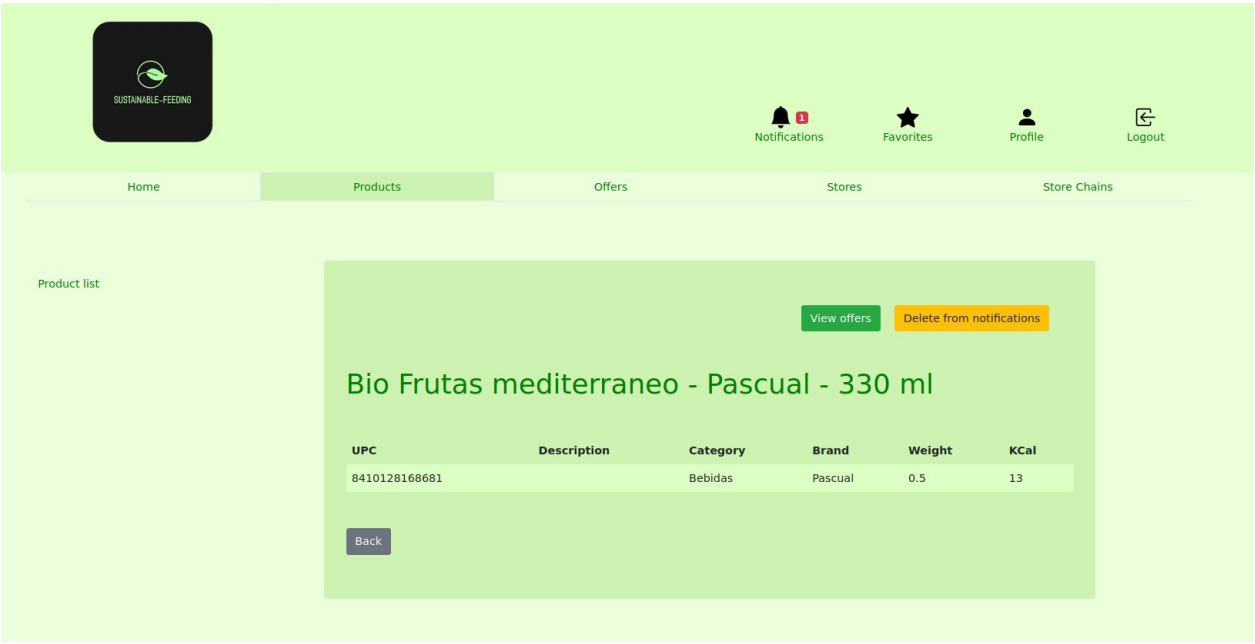


Figura 92: Ventana detalle de producto para usuario

Desde esta ventana pueden verse las ofertas de cada producto a través del botón "*View Offers*" y eliminar o añadir (según si ya se encuentra añadido o no) el producto para añadir notificaciones del mismo.

Tanto desde esta ventana pulsando “View Offers” como desde la ventana anterior pulsando el icono del diamante se puede ver la lista de las ofertas que existen actualmente sobre el producto:

Figura 93: Ventana ofertas de producto para usuario

Sobre la lista de ofertas del producto puede, tanto ampliarse la información de la oferta pulsando el icono del ojo, como añadir la oferta a favoritos en el caso de que no se encuentre actualmente o eliminarla en caso de que ya se encuentre.

Menú central: Ofertas

Si se desea acceder a la información de las ofertas que actualmente existen en la aplicación, se deberá seleccionar “*Offers*” en el menú central, accediendo a la siguiente ventana:

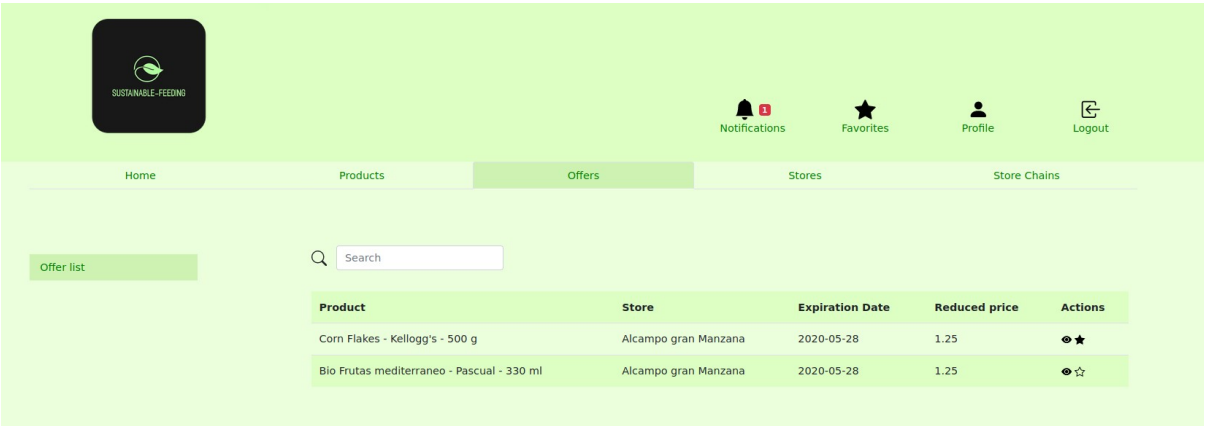


Figura 94: Ventana lista de ofertas para usuario

En cada oferta de la lista puede accederse a su información detallada pulsando el icono del ojo, como añadir la oferta a favoritos en el caso de que no se encuentre actualmente o eliminarla en caso de que ya se encuentre.

Si se seleccionar el icono del ojo, se accede a la ventana de detalle de la oferta:

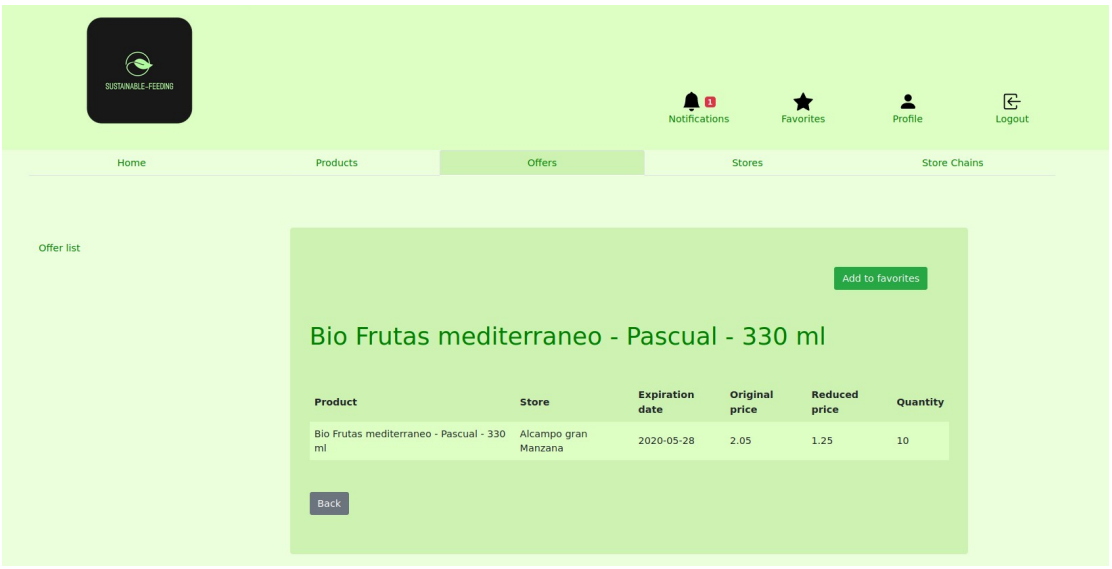


Figura 95: Ventana detalle de oferta para usuario

Desde esta pantalla es posible también añadir la oferta a favoritos en el caso de que no se encuentre actualmente o eliminarla en el caso de que ya se encontrase.

Menú central: Tiendas

Si se desea acceder a la información de las tiendas que actualmente existen en la aplicación, se deberá seleccionar "*Stores*" en el menú central, accediendo a la siguiente ventana:

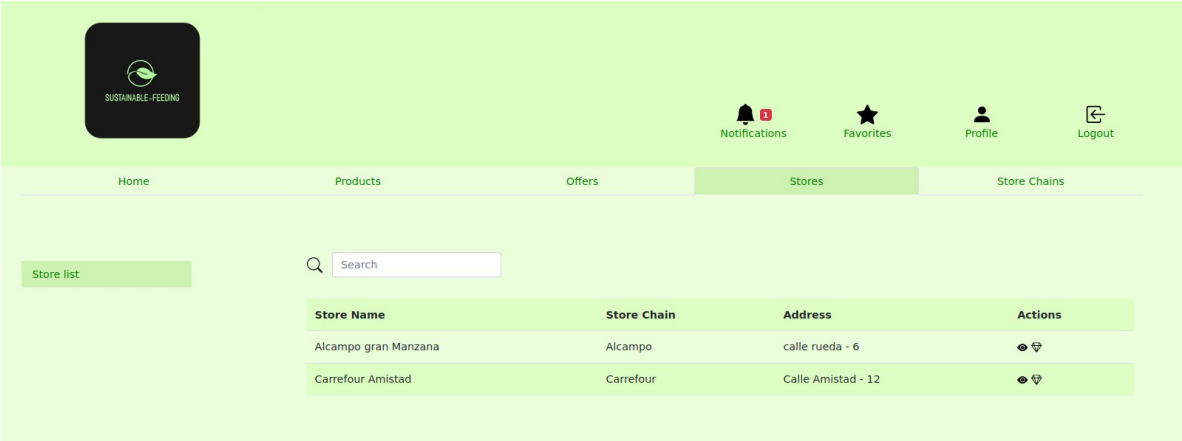


Figura 96: Ventana lista de tiendas para usuario

Desde cada una de las tiendas mostradas en la lista puede accederse a su información detallada seleccionando el icono del ojo:

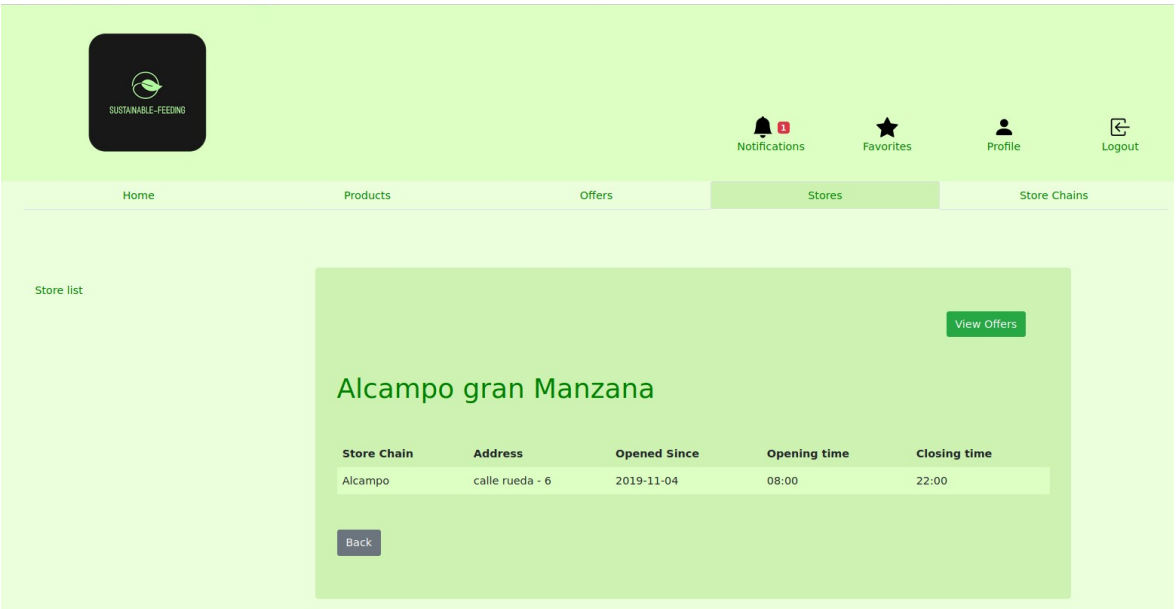


Figura 97: Ventana detalle de tienda para usuario

Desde esta ventana y desde el icono del diamante de la ventana anterior pueden consultarse las ofertas que esta tienda tiene publicadas actualmente en la aplicación:

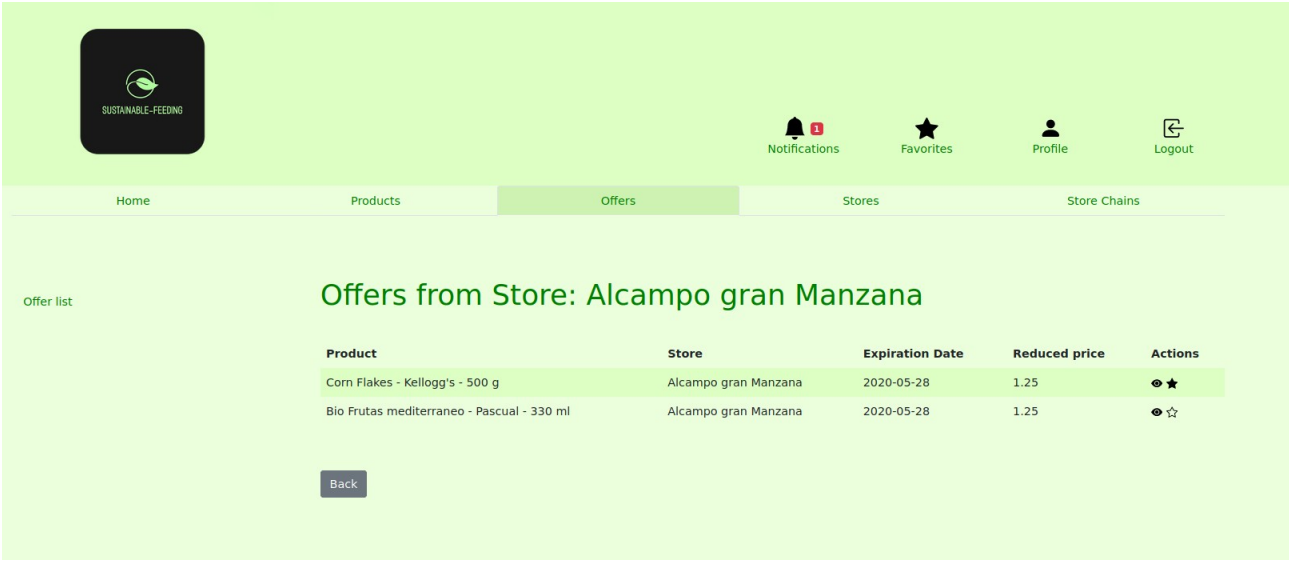


Figura 98: Ventana ofertas de una tienda para usuario

Desde cada una de las ofertas de la lista, podrá verse su detalle pulsando el icono del ojo, así como añadirla o eliminarla de favoritos.

Menú central: Cadena tiendas

Si se desea acceder a la información de las cadenas de tiendas que actualmente existen en la aplicación, se deberá seleccionar "*Store Chains*" en el menú central, accediendo a la siguiente ventana:

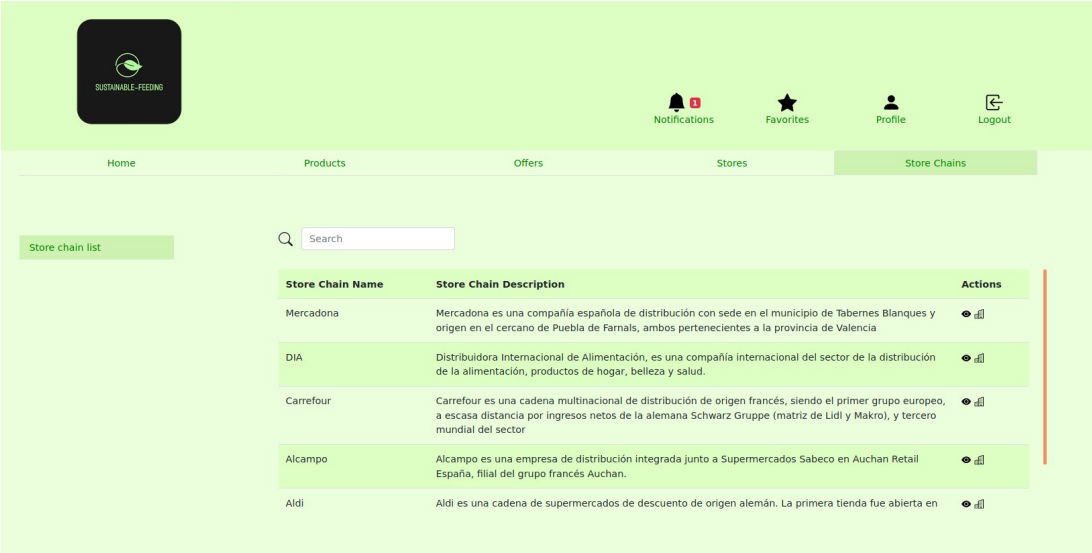


Figura 99: Ventana lista de cadenas de tiendas para usuario

Donde desde cada una de las cadenas de tiendas de la lista puede verse su detalle pulsando en el icono del ojo:

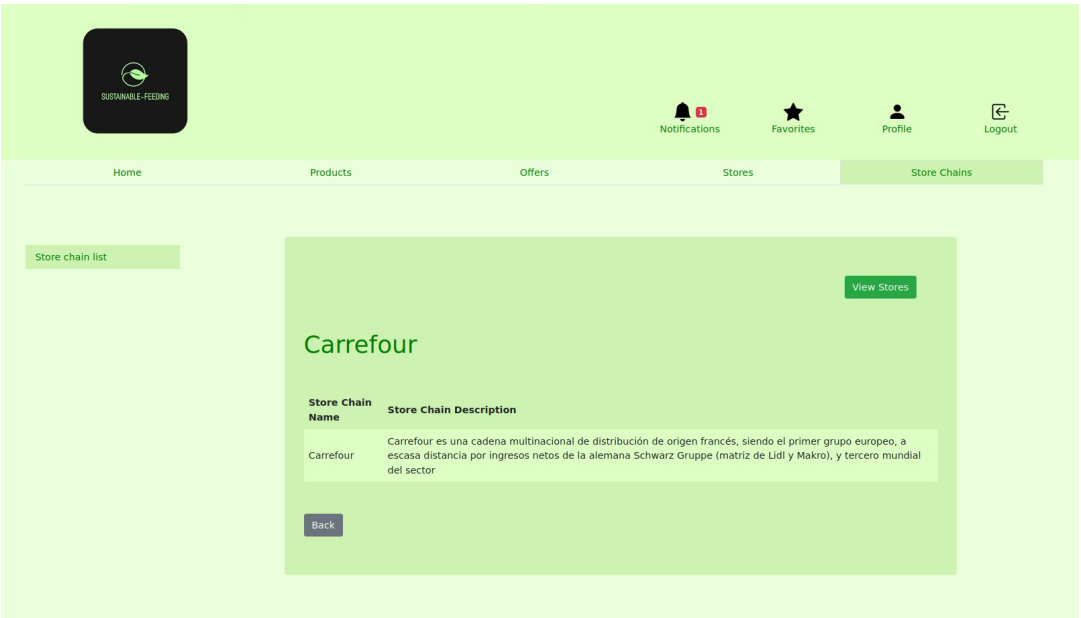


Figura 100: Ventana detalle de cadena de tiendas para usuario

Tanto desde el botón de esta ventana “View Stores” como desde el icono del edificio de la ventana anterior pueden consultarse las tiendas que pertenecen a tal cadena de tiendas que se encuentran dadas de alta en la aplicación:

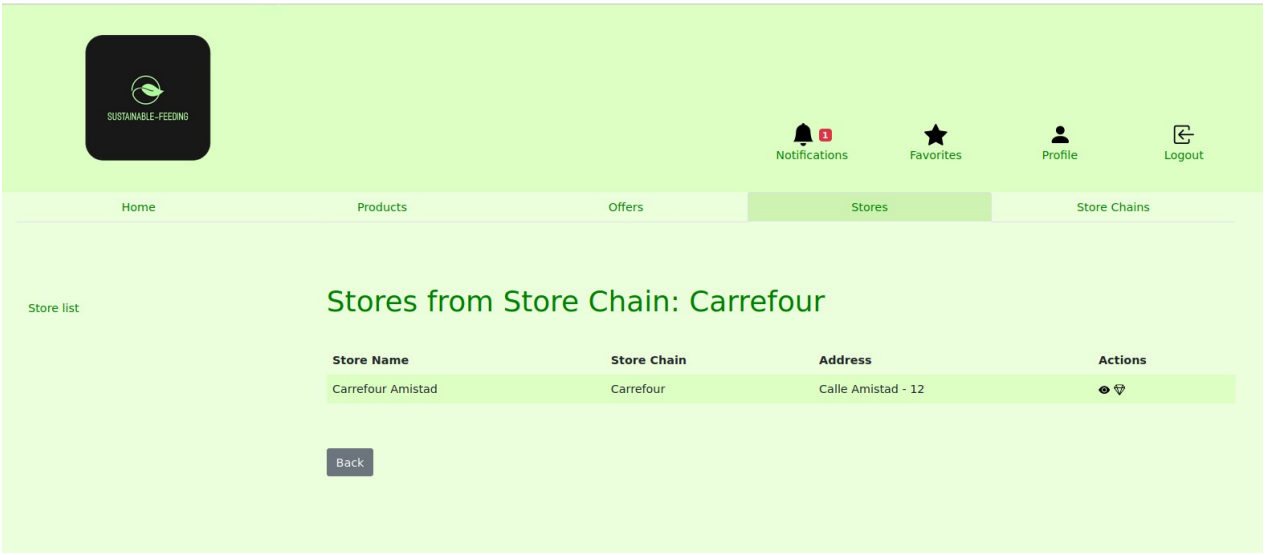


Figura 101: Ventana tiendas de una cadena de tiendas para usuario

Desde cada elemento de esta lista podrá verse el detalle de cada tienda pulsando el icono del ojo así como la lista de las ofertas que tiene publicadas en la aplicación, pulsando el icono del diamante.

Anexo 5.2.2 Opciones disponibles para comercios

Opciones de la cabecera superior

A continuación se muestra la página home cuando se accede a la aplicación con una cuenta de comercio.

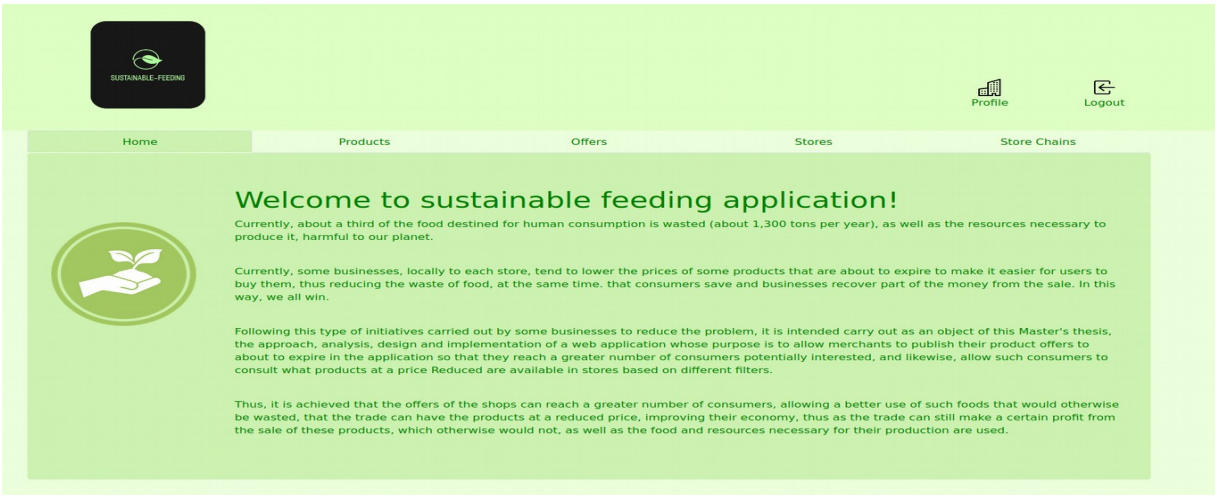


Figura 102: Ventana home para comercios

En la cabecera superior pueden observarse las operaciones de gestión del perfil de comercio y la de salida de la aplicación.

Si se selecciona el icono del edificio, se accede a la ventana de visualización del perfil del comercio:

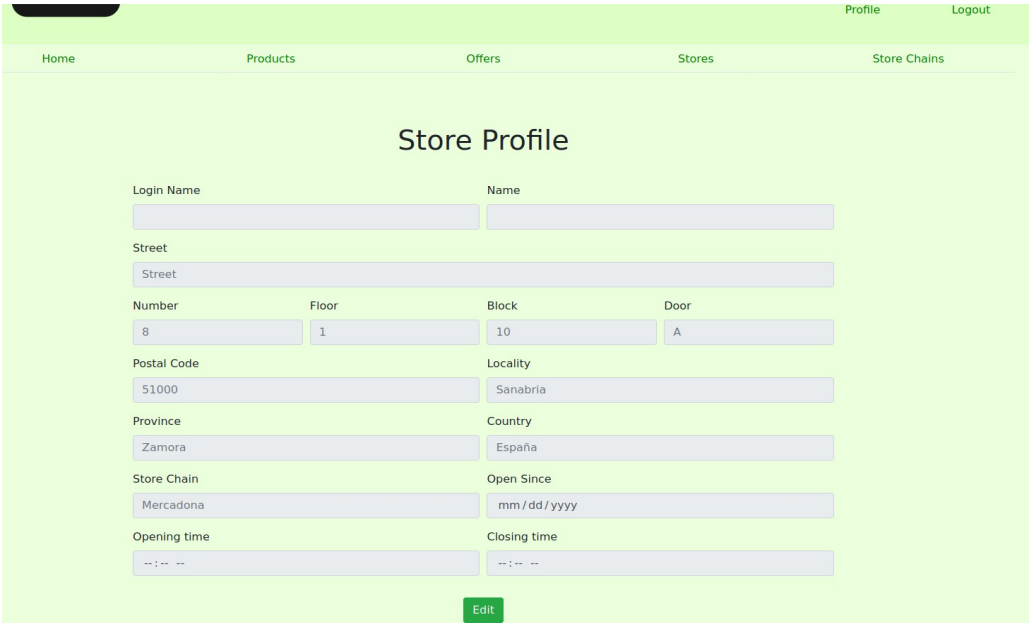


Figura 103: Ventana ver perfil para comercios

Donde pueden consultarse todos los datos de perfil del comercio.

Si se desea editar alguno de dichos datos, se debe seleccionar el botón "*Edit*", mediante el cual se accederá a la siguiente ventana, en la cual podrán modificarse los datos del perfil que se deseen.

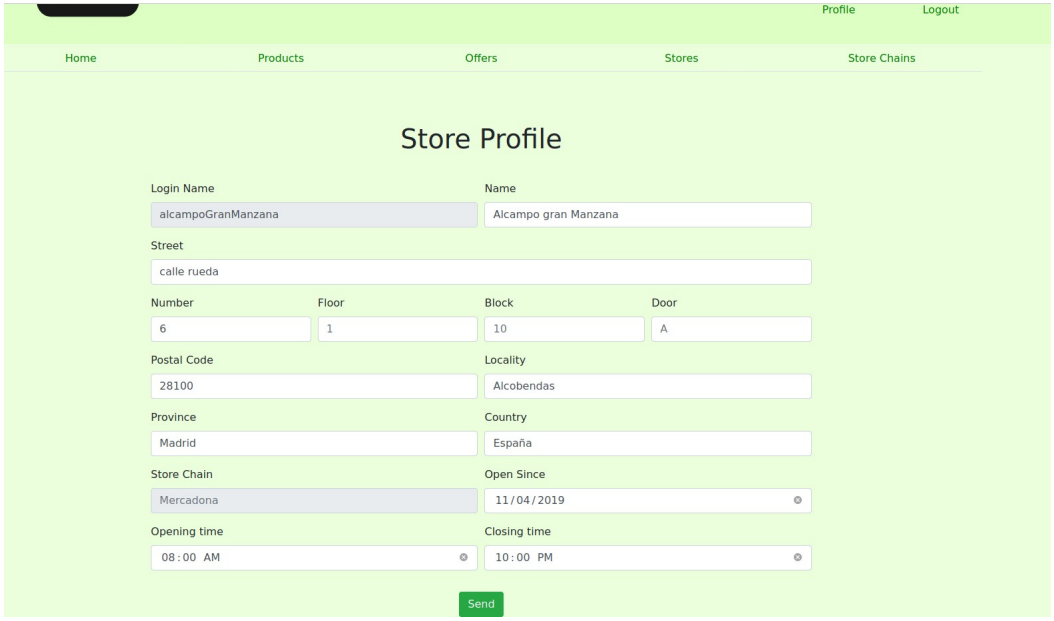


Figura 104: Ventana edición de perfil para comercios

Menú central: Productos

Si se desea acceder a la información de los productos dados de alta en la aplicación, se deberá seleccionar "*Products*" en el menú central, accediendo a la siguiente ventana:

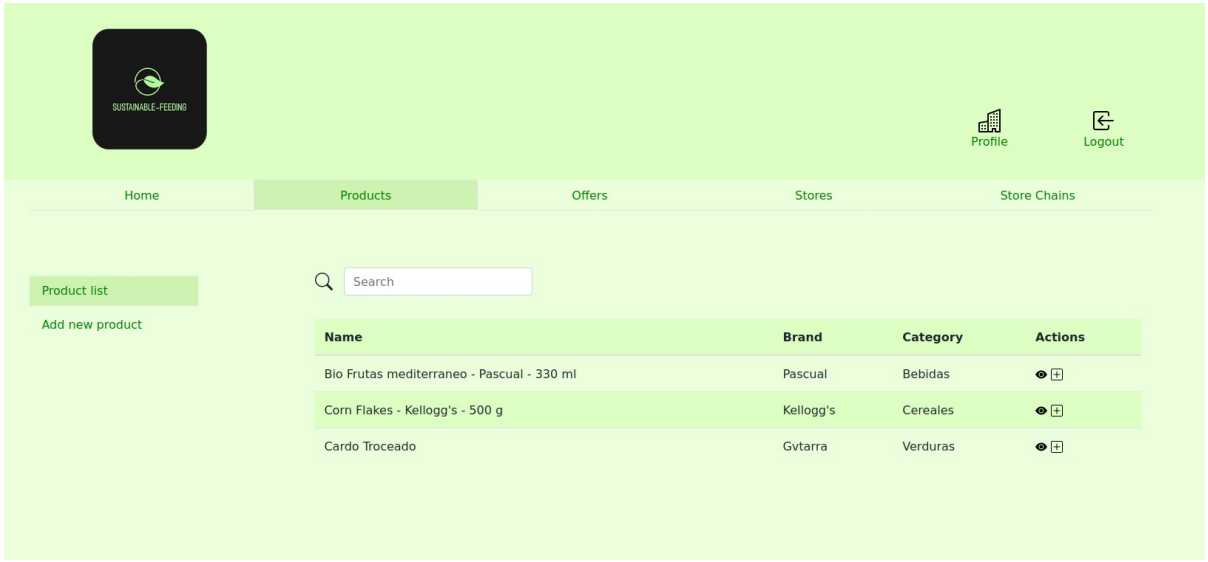


Figura 105: Ventana lista de productos para comercios

Para ver información ampliada de cada producto, se debe pulsar sobre el icono del ojo, llegándose a la siguiente ventana:

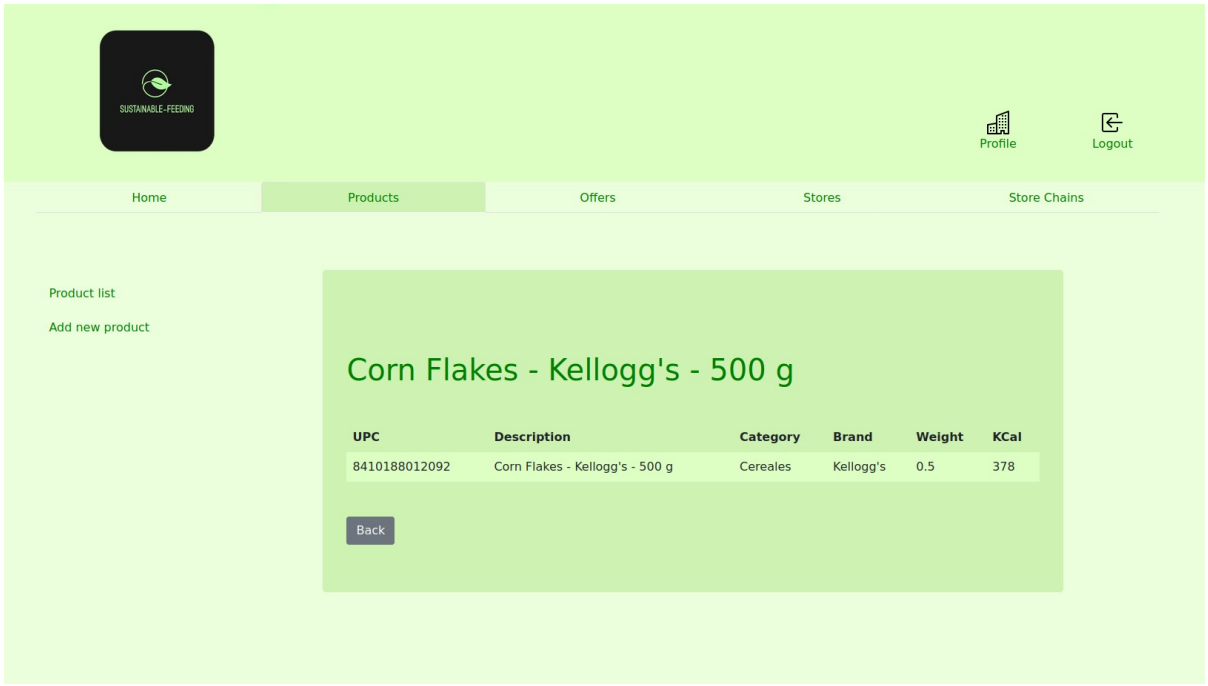


Figura 106: Ventana detalle de producto para comercios

Mientras que si se pulsa el icono del más, se abrirá la siguiente ventana modal:

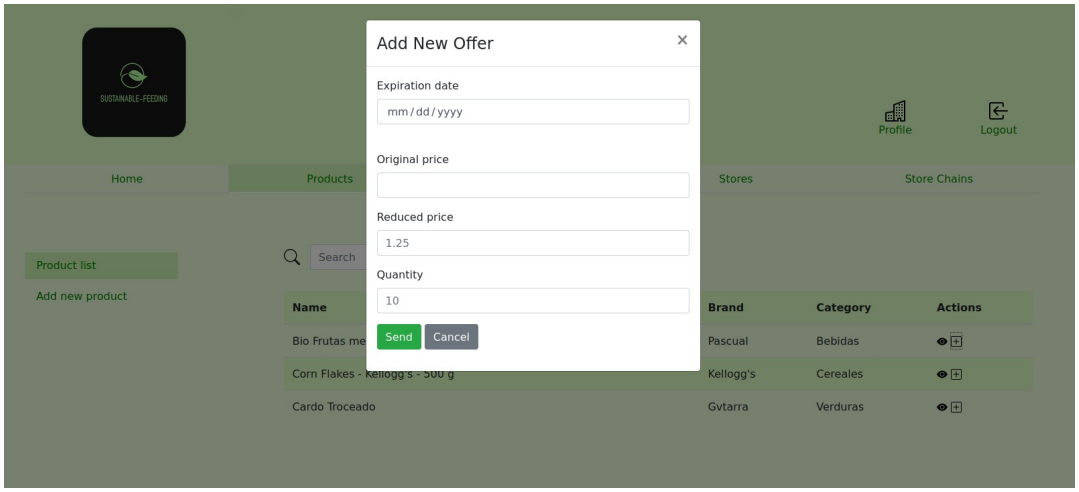


Figura 107: Ventana alta de oferta para comercios

Que permitirá añadir una oferta con los datos introducidos sobre el producto seleccionado.

Desde una cuenta de comercio es posible añadir nuevos productos a la aplicación, para ello, basta con seleccionar la opción “*Add new product*” y rellenar el siguiente formulario con los datos del producto:

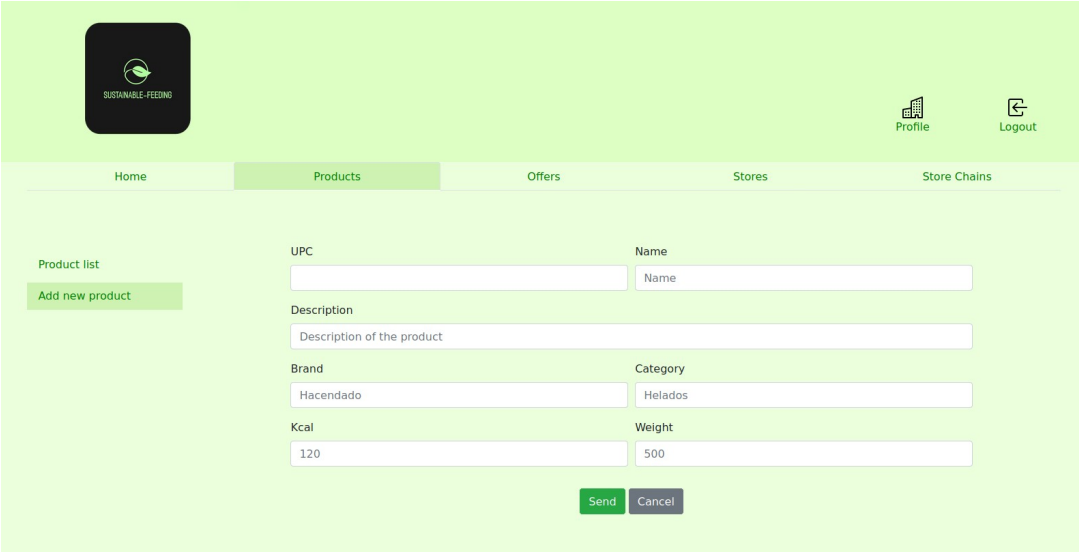


Figura 108: Ventana alta de producto para comercios

Menú central: Ofertas

Si se desea acceder a la información de las ofertas que actualmente existen en la aplicación, se deberá seleccionar "*Offers*" en el menú central, accediendo a la siguiente ventana:

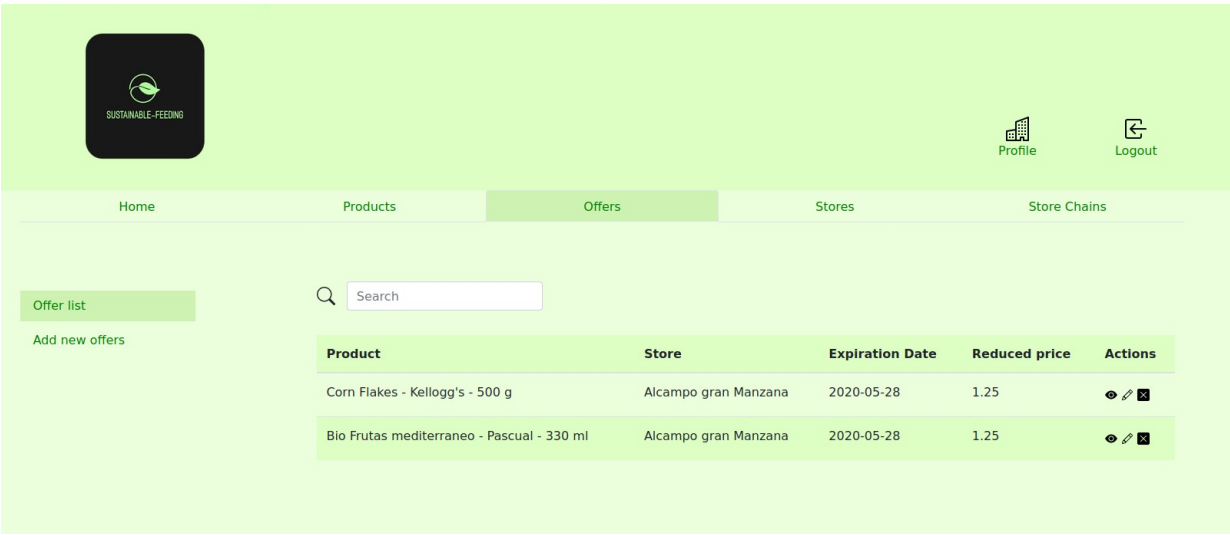


Figura 109: Ventana lista de ofertas para comercios

El listado obtenido será el de las ofertas dadas de alta por el comercio con el que se ha accedido a la aplicación. En cada oferta de la lista puede accederse a su información detallada pulsando el icono del ojo, así como editar los datos de tal oferta, seleccionando el icono del lapiz o eliminarla seleccionando el icono de la cruz.

Si se seleccionar el icono del ojo, se accede a la ventana de detalle de la oferta:

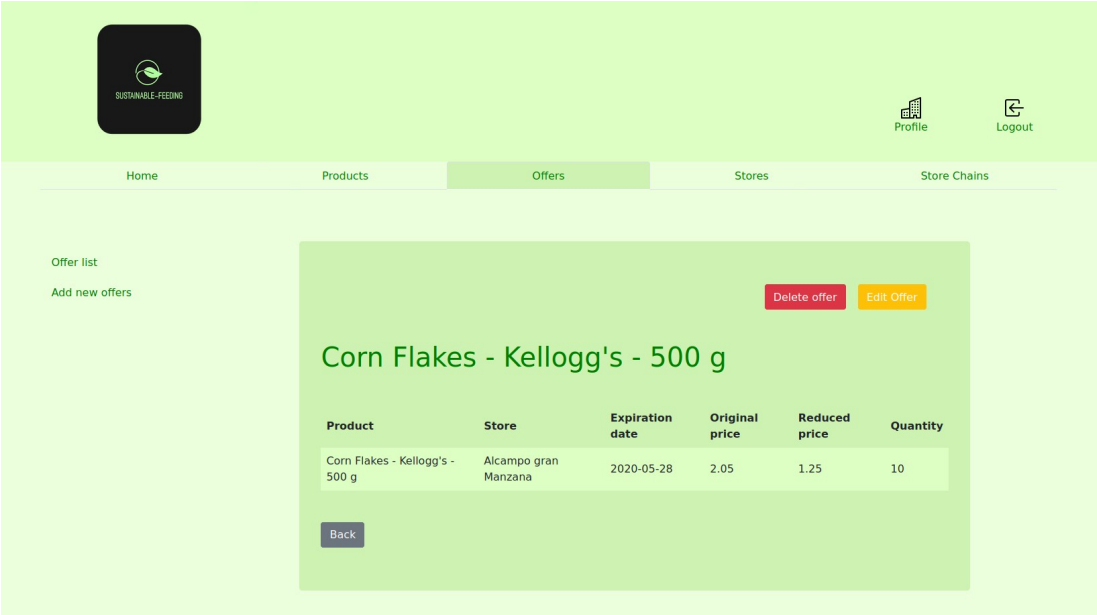


Figura 110: Ventana detalle de oferta para comercios

Desde esta pantalla es posible también eliminar la oferta o editarla.

Si se pulsa el botón "*Edit offer*" de esta ventana o el icono del lápiz en la ventana anterior, se accede al formulario de edición de la información de la oferta que se muestra a continuación:

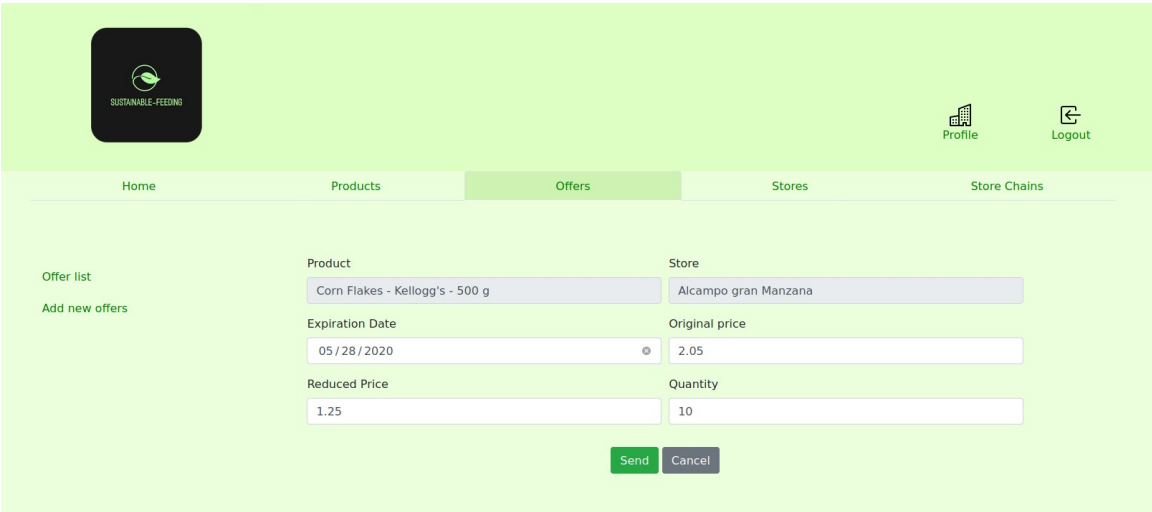


Figura 111: Ventana de edición de oferta para comercios

Si se desea añadir de una vez varias ofertas en la aplicación, deberá seleccionarse la opción en la lista de la derecha "*Add new offers*" llegándose a la siguiente ventana:

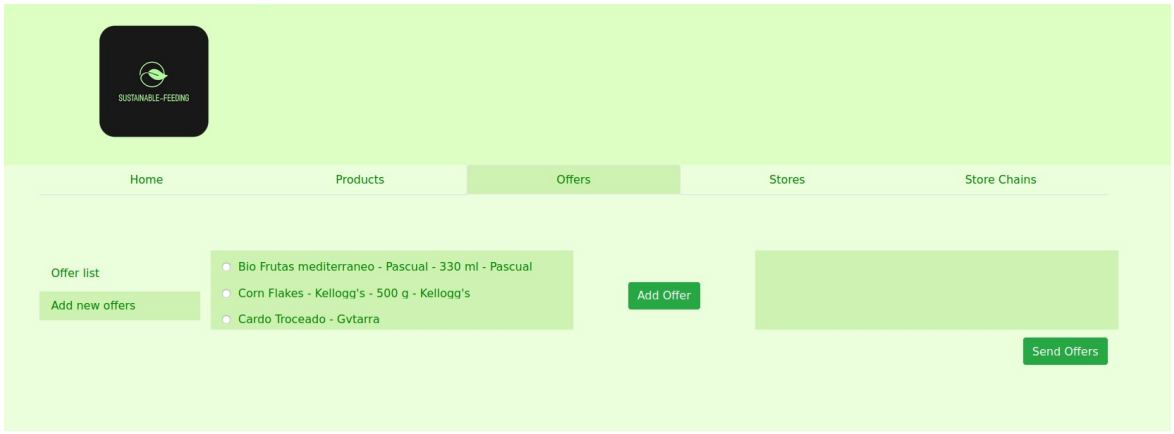


Figura 112: Ventana de alta de lista de ofertas para comercios inicial

En esta ventana se muestra la lista de productos en la lista de la izquierda. Si se quiere añadir una oferta sobre uno de los productos, deberá seleccionarse y pulsar el botón "*Add offer*", momento en el cual se moestrará una ventana modal en la que se solicitan los datos de la nueva oferta sobre el producto:

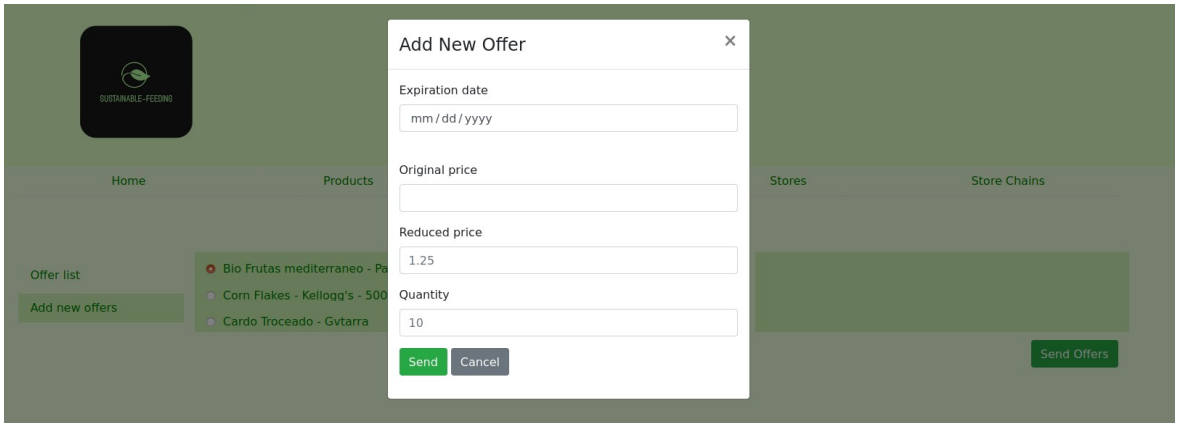


Figura 113: Ventana modal para alta de oferta para comercios

Y una vez se envíen tales datos, se observará la nueva oferta en la lista de la derecha:

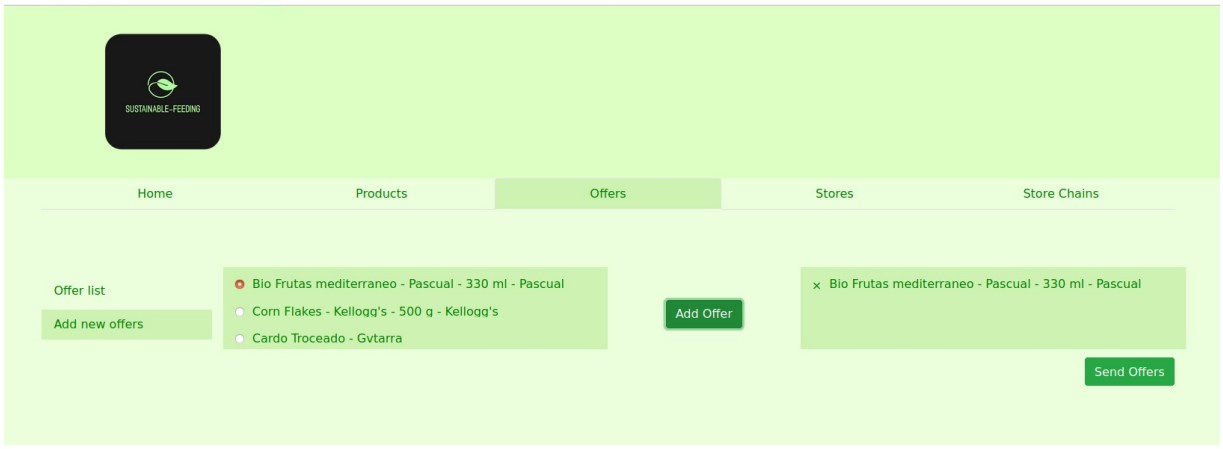


Figura 114: Ventana de alta de lista de ofertas para comercios final

Donde podrá ser eliminada pulsando sobre la cruz a la izquierda de su nombre si finalmente no se desea enviarla.
El mismo procedimiento puede repetirse tantas veces como ofertas sobre los productos deseen darse de alta. Y una vez se encuentren todas las nuevas ofertas en la lista de la derecha, se deberá seleccionar "*Send Offers"* para darlas todas de alta en la aplicación.

Menú central: Tiendas

Si se desea acceder a la información de los comercios de la misma cadena de tiendas existentes en la aplicación, se deberá seleccionar "*Stores*" en el menú central, accediendo a la siguiente ventana:

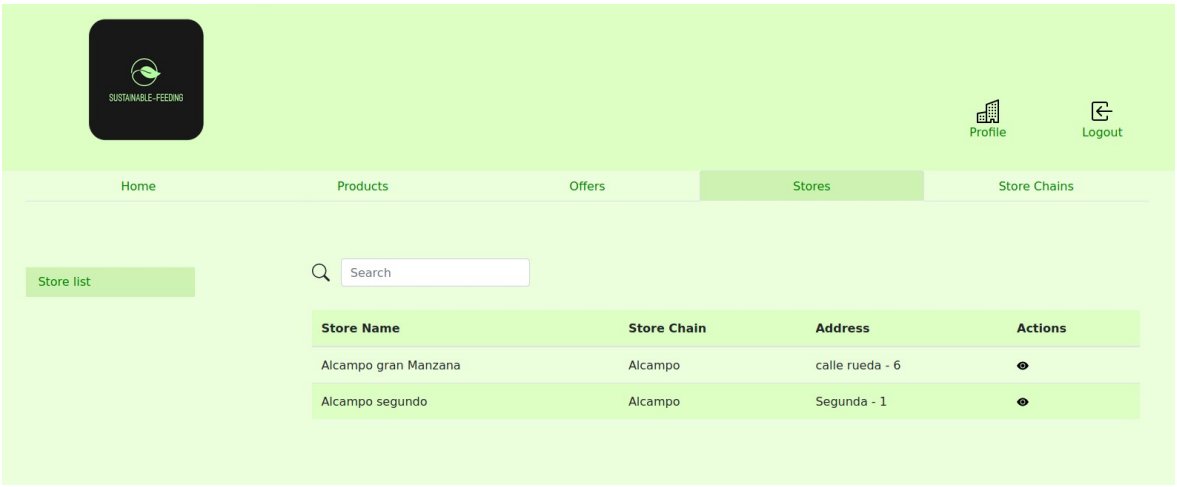


Figura 115: Ventana de lista tiendas para comercios

Desde cada una de las tiendas mostradas en la lista puede accederse a su información detallada seleccionando el icono del ojo:

Figura 116: Ventana detalle de tienda para comercios

Menú central: Cadena tiendas

Si se desea acceder a la información de cadena de tiendas a la que pertenece el comercio con el que se ha accedido, se deberá seleccionar "*Store Chains*" en el menú central, accediendo a la siguiente ventana:

Figura 117: Ventana lista de cadena de tiendas para comercios

Donde la cadena de tiendas de la lista puede verse su detalle pulsando en el icono del ojo:

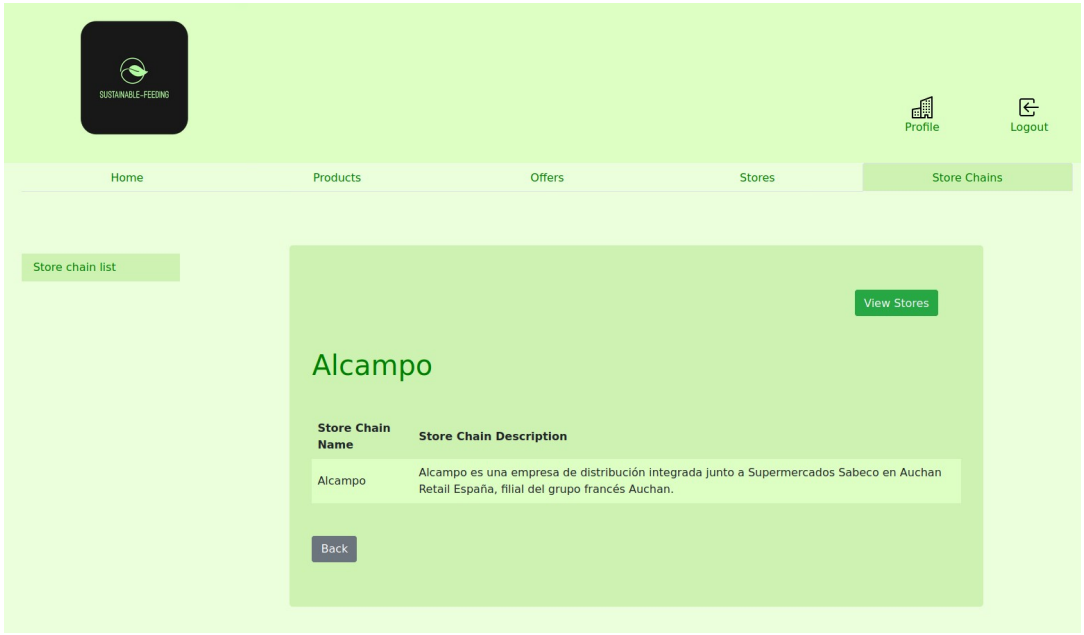


Figura 118: Ventana detalle de cadena de tiendas para comercios

Tanto desde esta ventana en el botón "*View Store*", como en la ventana anterior en el icono del edificio se puede consultar la lista de tiendas que pertenecen a la cadena de tiendas:

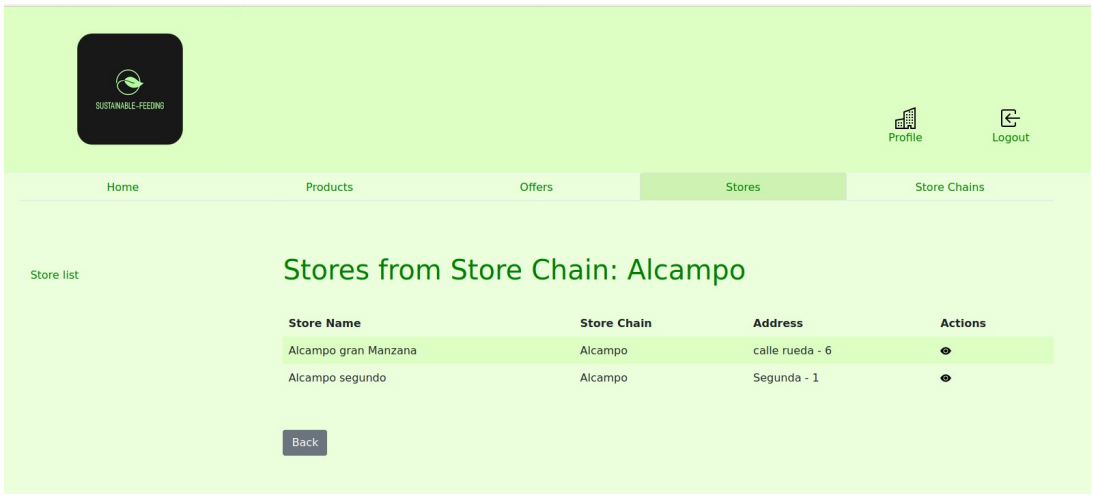


Figura 119: Ventana tiendas de una cadena de tiendas para comercios

Anexo 5.2.13 Opciones disponibles para administradores

Menú central: Productos

Si se desea acceder a la información de los productos dados de alta en la aplicación, se deberá seleccionar "*Products*" en el menú central, accediendo a la siguiente ventana:

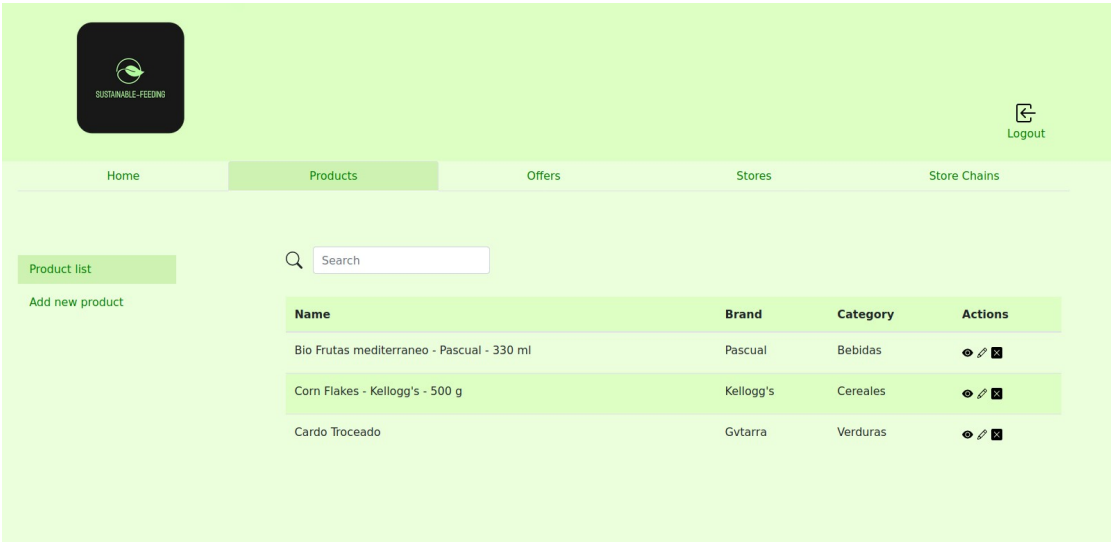


Figura 120: Ventana lista de productos para administradores

Para ver información ampliada de cada producto, se debe pulsar sobre el icono del ojo, llegándose a la siguiente ventana:

Figura 121: Ventana detalle de producto para administradores

Es posible borrar el producto seleccionando el botón "*Delete product*" o pulsando el icono de la equis de la ventana anterior, así como editar el producto seleccionando el botón "*Edit product*" de esta ventana o pulsando el icono del lápiz de la ventana anterior, llegándose a la siguiente ventana de edición de producto:

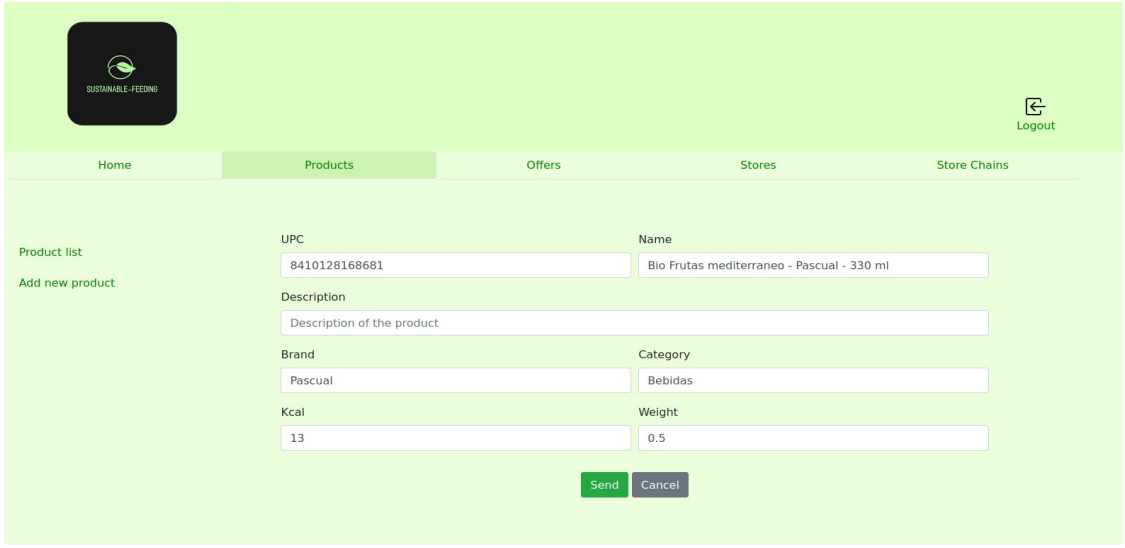


Figura 122: Ventana editar nuevo producto para administradores

Desde una cuenta de administrador es posible añadir nuevos productos a la aplicación, para ello, basta con seleccionar la opción "*Add new product*" y rellenar el siguiente formulario con los datos del producto:

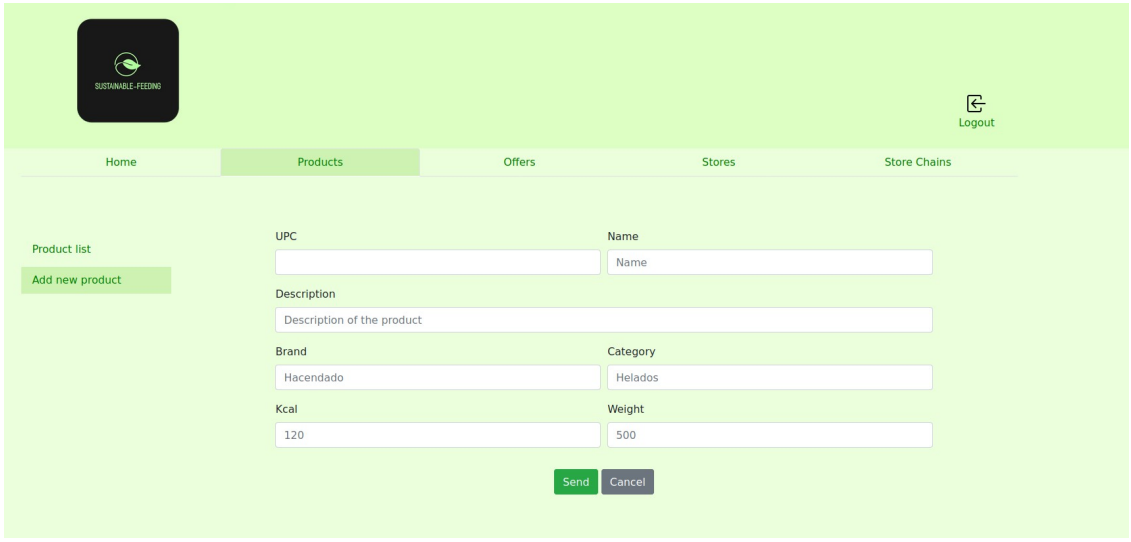


Figura 123: Ventana añadir nuevo producto para administradores

Menú central: Ofertas

Si se desea acceder a la información de las ofertas que actualmente existen en la aplicación, se deberá seleccionar "*Offers*" en el menú central, accediendo a la siguiente ventana:

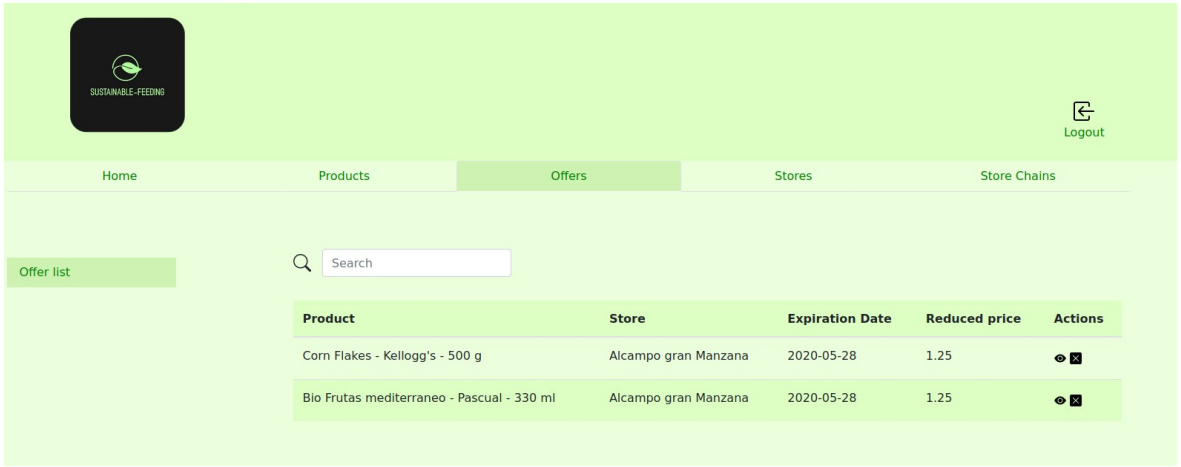


Figura 124: Ventana lista de ofertas para administradores

En cada oferta de la lista puede accederse a su información detallada pulsando el icono del ojo, así como eliminarla seleccionando el icono de la cruz.

Si se seleccionar el icono del ojo, se accede a la ventana de detalle de la oferta:

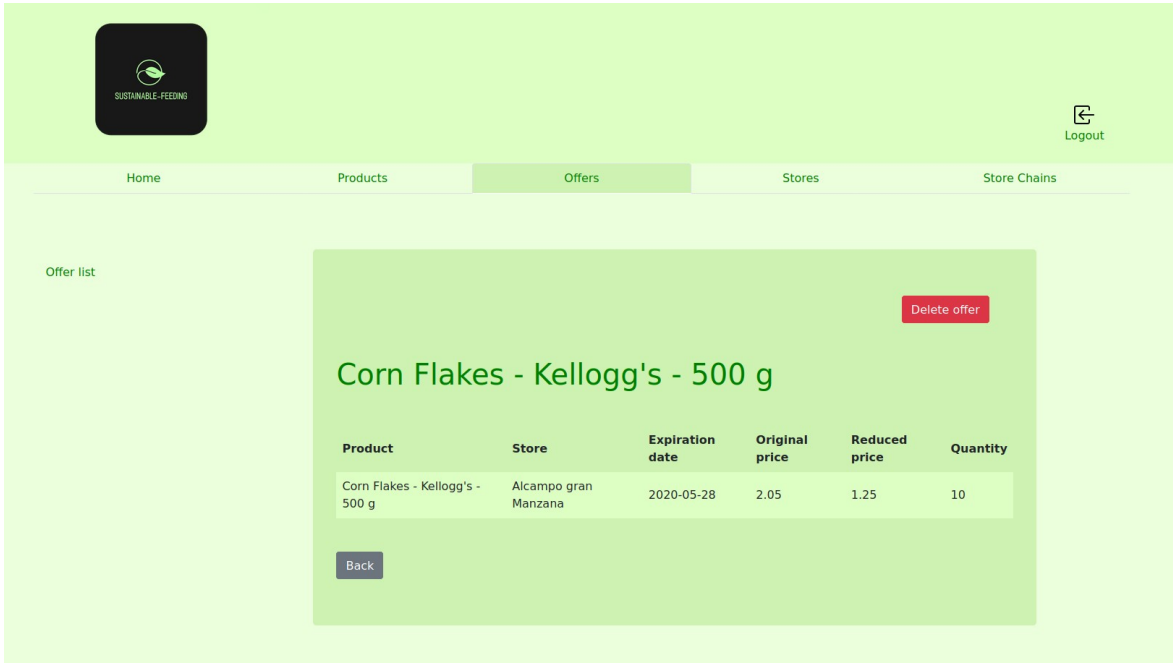


Figura 125: Ventana detalle de oferta para administradores

Menú central: Tiendas

Si se desea acceder a la información de las cadenas de tiendas existentes en la aplicación, se deberá seleccionar "*Stores*" en el menú central, accediendo a la siguiente ventana:

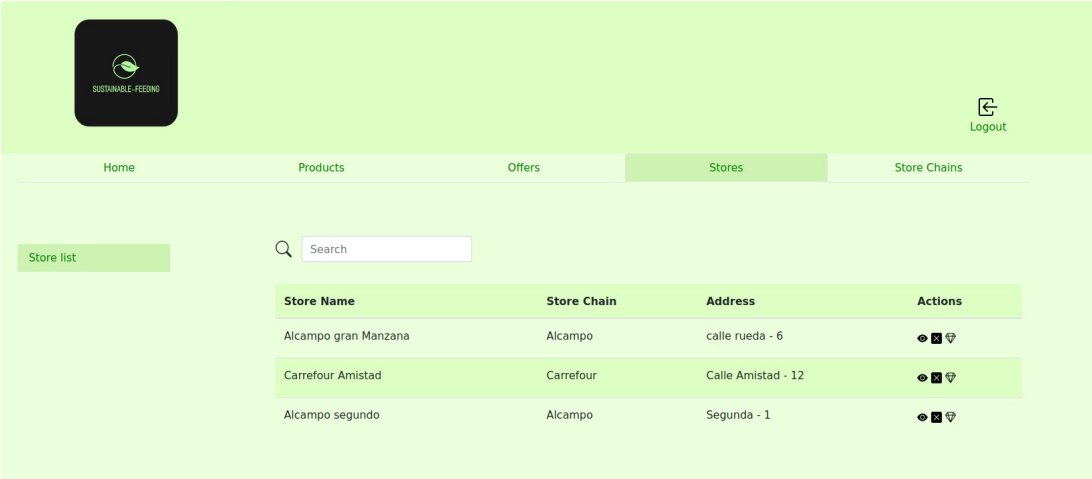


Figura 126: Ventana lista de tiendas para administradores

Desde cada una de las tiendas mostradas en la lista puede accederse a su información detallada seleccionando el icono del ojo, o eliminarla seleccionando el icono de la equis o ver las ofertas de la tienda seleccionando el icono del diamante.

Menú central: Cadena tiendas

Si se desea acceder a la información de cadena de tiendas dadas de alta en la aplicación, se deberá seleccionar "*Store Chains*" en el menú central, accediendo a la siguiente ventana:

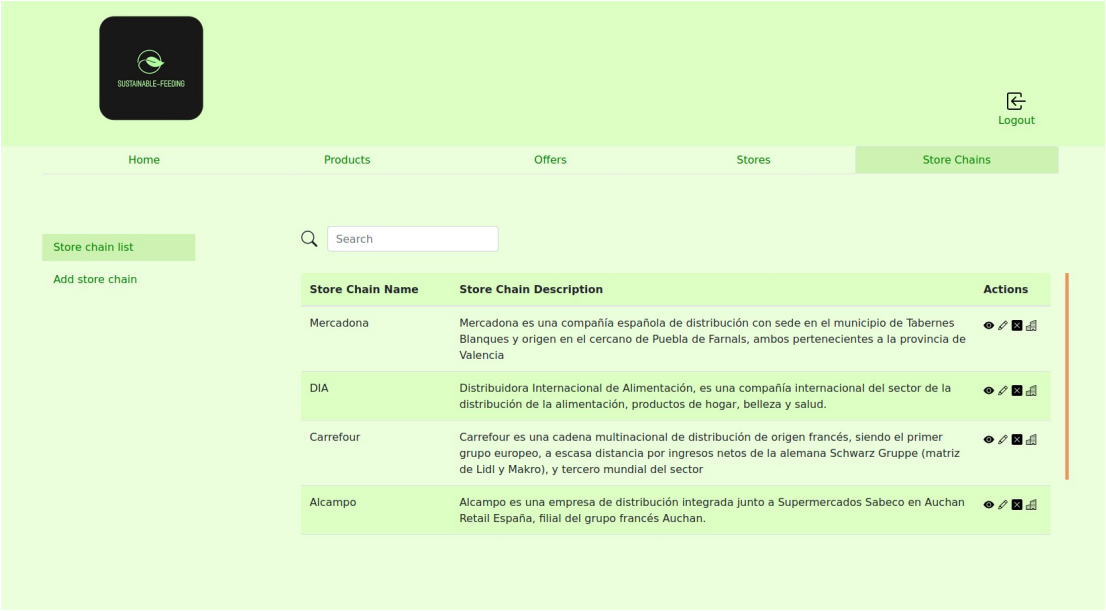


Figura 127: Ventana lista de cadenas de tiendas para administradores

Donde desde cada elemento de la lista podrá accederse al detalle de cada cadena de tiendas seleccionando el icono del ojo, editar la información de una cadena de tiendas seleccionando el icono del lápiz, eliminar una cadena de tiendas seleccionando el icono de la equis o ver las tiendas de la cadena seleccionando el icono del edificio.

El detalle de una cadena de tiendas es como se muestra a continuación:

Figura 128: Ventana detalle de cadena de tiendas para administradores

Desde esta ventana también es posible eliminar la cadena de tiendas, editar su información o ver las tiendas que pertenecen a tal cadena de tiendas mediante los botones "*Delete store chain", "Edit Store Chain"* y *"view Stores"* respectivamente.

La ventana de edición de información de una cadena de tiendas es como la que se muestra a continuación:

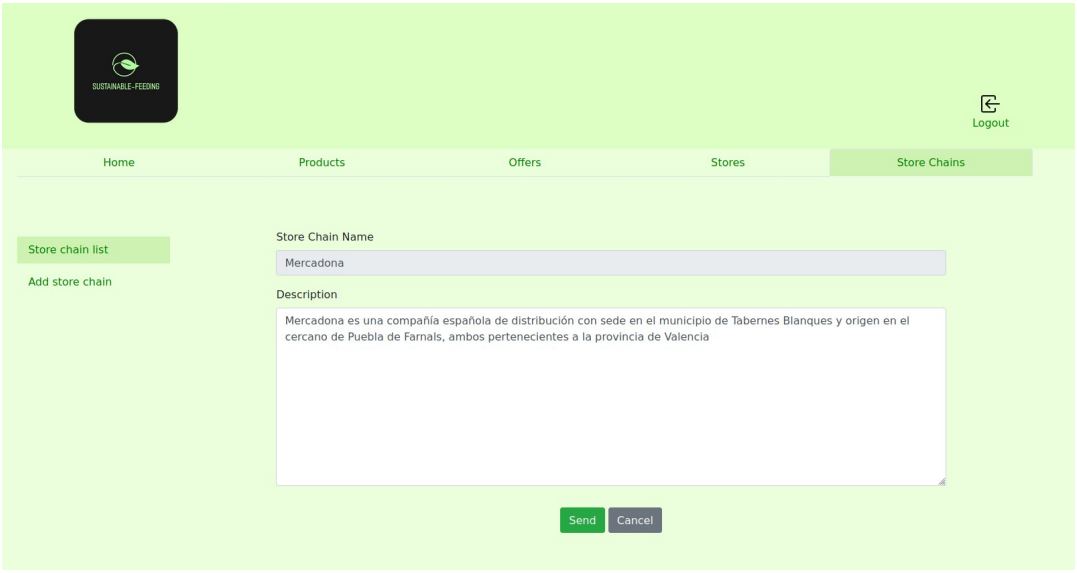


Figura 129: Ventana edición de cadena de tiendas para administradores

Además, con el la cuenta de administrador pueden darse de alta nuevas cadenas de tiendas en la aplicación, para ello, se debe acceder a la opción *"Add store chain"* del menú de la izquierda, visualizandose una ventana como la siguiente:

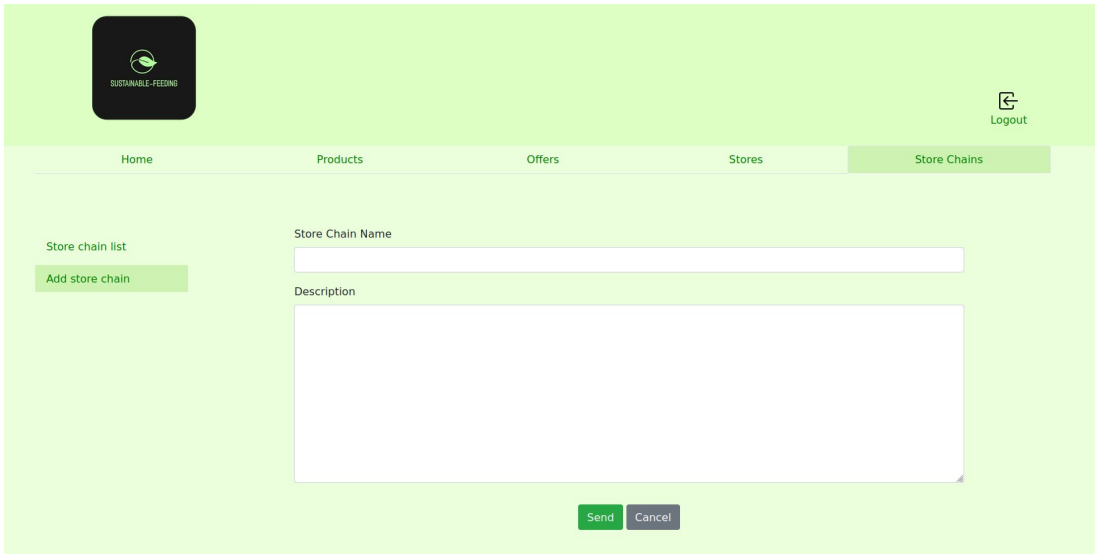


Figura 130: Ventana adición de cadena de tiendas para administradores

Anexo 6. Libro de estilo

Libro de estilo que define la línea gráfica del trabajo. Es recomendable incluir, entre otros:

- Logotipos y anagramas.
- Paleta de colores.
- Paleta tipográfica y tamaño de fuentes.
- Márgenes.
- Fondos, iconos y otros elementos gráficos.
- Banners, botones y otros elementos de promoción.
- Reglas de uso.
- Ejemplos de uso correcto e incorrecto.

Anexo 7. One-page business plan/Resumen ejecutivo

Micro-plan de empresa de una página de extensión, sin contar la titulación de capítulo y con el formato de texto que el alumno considere. En su realización ha de tenerse en cuenta que debe poder presentarse como documento individual. Debe resumir, como mínimo, los siguientes aspectos, propios de un plan de empresa completo:

- Nombre comercial
- Resumen comercial
- Modelo de negocio
- Expertise
- Productos y servicios
- Mercado
- Competencia
- Plan de marketing
- Inversión inicial y costes a corto y medio plazos
- Proyección económica corto y medio plazos y ROI
- DAFO

Anexo 8. Glosario/Índice analítico

Glosario de términos y acrónimos utilizados en el trabajo (sólo aquellos mencionados en el presente documento) con breves definiciones de cada uno de ellos, o un índice analítico con la lista de términos, nombres y palabras clave en el texto y las páginas donde se pueden encontrar.

Anexo 9. Bibliografía

[1] Save Food (2011), Pérdidas y desperdicio de alimentos en el mundo, [Online]. Recuperado de http://www.fao.org/3/a-i2697s.pdf

[2] Castillo, S (2019). Autenticación de APIs basadas en tokens con Spring y JWT [Figura]. Recuperado de blog.softtek.com.

[3] aspgems.com (2019). Metodología de desarrollo de software (II) – Modelo de diseño iterativo [Figura]. Recuperado de https://aspgems.com/metodologia-de-desarrollo-de-software-ii-modelo-de-diseno-iterativo

Anexo 10. Vita

Sección opcional aunque recomendable.

Breve nota biográfica del autor del TF. Máximo 700 caracteres.