

E-COMMERCE

Documento de Arquitectura de Software

Versión 1.0

- Aldo Ponce de la Cruz A01651119
- Ignacio Alvarado Reyes A01656149
- José Antonio Berrón Gutiérrez A01377269
- Arturo Efrén Jiménez Garibaldi A00824428
- Manuel Herrasti González A01657028

Historial de Revisiones

| Fecha | Versión | Descripción | Autor |
|------------|---------|-----------------|-----------|
| 30/03/2022 | 1 | Primera Versión | El equipo |
| | | | |
| | | | |
| | | | |

Contenido

| | |
|--|----------|
| 1. Introducción | 4 |
| 1.1 Propósito | 4 |
| 1.2 Alcance | 4 |
| 1.3 Definiciones, Siglas y Abreviaturas | 5 |
| 1.4 Referencias | 5 |
| 1.5 Resumen | 6 |
| 2. Representación Arquitectónica | 8 |
| 2.1.1 Objetivos Arquitectónicos | 8 |
| 2.1.2 Restricciones arquitectónicas | 8 |
| 2.2 Estilo Arquitectónico y rationale | 10 |
| 2.2 Vista de casos de uso | 11 |
| 2.3 Escenarios Arquitectónicos y Requerimientos No Funcionales Asociados | 28 |
| 2.3.1 Escenarios Arquitectónicos | 28 |
| 2.3.1 Requerimientos No Funcionales Asociados | 41 |
| 2.3 Vista lógica | 45 |
| 2.4 Vista de proceso | 46 |
| 2.5 Vista de implementación | 51 |
| 2.6 Vista de datos | 51 |
| 2.6.1 Base de datos | 52 |

| | |
|--|-----------|
| 2.7 Vista de layers | 55 |
| 2.8 Diagrama de microservicios | 56 |
| 3. Lineamientos Arquitectónicos | 57 |
| 3.1 Diseño | 58 |
| 3.1 Codificación | 59 |
| 3.2 Reutilización de Software | 59 |

Documento de Arquitectura de Software

1. Introducción

1.1 Propósito

La creación de una webapp que permita a un grupo de administradores utilizar un sistema crud de una tienda online, para la gestión de productos y usuarios. Y por otra parte, una webapp para los usuarios que permita realizar compras y pedidos de productos electrónicos.

El sistema permitirá a los usuarios comprar productos que se ofrezcan en la tienda online, y de la misma manera los usuarios administradores pueden añadir sus propios productos para ser vendidos dentro de la plataforma.

Los administradores de la misma manera deben ser capaces de administrar los usuarios para un funcionamiento adecuado del sistema. En caso de ser necesario los administradores tendrán la capacidad de borrar usuarios inactivos y de la misma manera actualizar información de productos que hayan cambiado en algún ámbito.

El sistema de ventas debe tener implementado un módulo de seguridad de pago y entrega de productos, con esto asegurando el rendimiento de la página y la escalabilidad de la misma. Todo esto será gracias al uso de servicios o módulos añadidos a nuestro sistema principal.

1.2 Alcance

Crear un sistema CRUD para la administración de una tienda online y sistema de compras para usuarios.

Realizar un sistema donde toda la funcionalidad sea correcta como la autenticación de los distintos usuarios en nuestro sistema como: cliente, admin y superadmin. Este sistema debe de operar de una manera correcta dependiendo del usuario que use la página web.

Implementar una arquitectura de software que nos guíe en el proceso y documentación de una tienda en línea. A través de esta arquitectura de microservicios añadiremos paquetes que

amplíen la funcionalidad del sistema como: ventas, administración bancaria, pagos seguros, etc.

1.3 Definiciones, Siglas y Abreviaturas

CRUD: de las siglas en inglés Create Read Update Delete, se refiere a la serie de operaciones estándar que se realizan en el uso de una base de datos donde se crean, consultan, cambian y borran datos.

API: de las siglas en inglés Application Programming Interface (Interfaz de Programación de Aplicaciones) consiste básicamente en un intermediario de software que permite la comunicación entre dos programas.

Software: Datos o programas utilizados para completar tareas de cómputo.

Hardware: Parte física de una computadora.

PC: de las siglas en inglés Personal Computer (Computadora Personal). Hardware computacional designado para uso personal.

Módulos: Paquetes destinados al incremento de la funcionalidad del sistema teniendo en cuenta el sistema principal.

Customer: Persona que navega la página, este puede buscar productos por diferentes categorías y agregar los productos a un carrito. Sus productos y ajustes serán guardados en el perfil del cliente.

Admin: Usuario que administra los productos y clientes del sistema de ventas. Este tiene la posibilidad de modificar tanto los productos como los clientes.

SuperAdmin: Usuario con el nivel de acceso más profundo que tiene las características de un Admin, además de tener la posibilidad de modificar los Admins existentes.

REST: Es una interfaz de programación de aplicaciones (API o API web) que se ajustan a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful.

1.4 Referencias

What is CRUD? Explaining CRUD Operations | Sumo Logic. (2022). 30 Marzo 2022, de Sumo Logic

Sitio Web: <https://www.sumologic.com/glossary/crud/>

What is an API? (Application Programming Interface) | MuleSoft. (2022). 30 Marzo 2022, de MuleSoft

Sitio web: <https://www.mulesoft.com/resources/api/what-is-an-api>

What is Software? Definition, Types and Examples. (2022). 30 Marzo 2022, de techtarget Sitio Web:

<https://www.techtarget.com/searchapparchitecture/definition/software>

IBM Cloud Education. (2021). API REST. 30 Marzo 2022, de IBM Sitio web:

<https://www.ibm.com/mx-es/cloud/learn/rest-apis>

1.5 Resumen

2. Representación Arquitectónica

2.1.1 Objetivos Arquitectónicos

El objetivo principal de esta arquitectura es crear un sistema completamente modular, donde cada servicio es independiente de los otros, y existen interfaces que comunican los componentes. Cada componente puede ser desplegado por separado, los componentes pueden combinarse entre sí para representar un sólo servicio o función y el acceso a estos es a través de protocolos de acceso remoto. Todas estas ventajas y características están alineadas a ofrecer un servicio web con altos estándares de calidad.

Respecto a los drivers del proyecto, describimos los puntos más importantes.

Atributos de calidad: En el sistema se hará uso de métricas de calidad del sistema, como lo son el desempeño del mismo, la seguridad para los usuarios, la facilidad de modificar o realizar cambios al sistema y finalmente la utilización de pruebas y hacer que estas se realicen de manera eficaz y sencilla. Con estos podemos asegurar la calidad de nuestro sistema para cualquier usuario que quiera usar la plataforma.

Funcionalidades del sistema: El sistema mostrará un servicio eficaz para el acceso a los productos y carrito de compras de cada uno de los usuarios. De la misma manera debe de poder tener un manejo de sesión integrado para que el usuario pueda administrar la sesión como él vea conveniente.

Business Drivers: Entre los puntos más importantes para ofrecer un buen servicio de negocio está la eficiencia de respuesta de la página web, porque esto permite que más clientes puedan acceder al sistema y realizar compras de forma simultánea. También se ofrece un dashboard de administración que permite categorizar los productos y registrar sus respectivos precios, así se tiene un orden en cuanto a las finanzas disponibles del negocio.

Constraints: Estos están especificados en el siguiente apartado de restricciones arquitectónicas.

2.1.2 Restricciones arquitectónicas

Algunas desventajas considerables de esta arquitectura son los siguientes:

Mantenimiento: Los sistemas complejos y distribuidos suelen tener algunos problemas de mantenimiento, porque hay que conocer a detalle toda la estructura para tener las relaciones bien definidas. Algunos ejemplos de mantenimiento son implementación de comunicación interna entre todos los servicios del sistema y solicitudes que se extienden a otros servidores.

Disponibilidad del sistema: El sistema web puede estar comprometido si existen picos de usuarios en consultas, ya que se supera la capacidad de procesamiento de la base de datos y el host.

Mayor consumo de recursos: Puesto que cada microservicio tiene su propio sistema, es más costoso realizar procesos dentro de estas. Es importante tener un conteo y un límite de servicios que el usuario puede hacer uso en el programa. Al no limitar esto los recursos que el sistema puede utilizar son más grandes que un sistema sin servicios externos.

| | |
|-------------------------|------|
| Agilidad | High |
| Facilidad de despliegue | High |
| Pruebas | High |
| Rendimiento | Low |
| Escalabilidad | High |
| Facilidad de desarrollo | High |

2.2 Estilo Arquitectónico y rationale

Microservicios es un estilo arquitectónico que divide a la aplicación en una colección de servicios independientes.

Es un modelo de arquitectura que usa conceptos de:

- Unidades de despliegue: Cada componente es desplegado como una unidad separada.
- Componente de servicio: Clases que representan una función del sistema.
- Arquitectura distribuida: Los componentes están separados y se requieren protocolos de acceso para comunicarse.

En particular, seleccionamos la topología de API-REST, donde tendremos componentes granulares de servicio para las diferentes funcionalidades de nuestra tienda que se comunicaran con APIs externas.

Los componentes que tenemos planeados para nuestra aplicación son:

- Carrito de Compras: Agregación de productos y estimación de precios.
- Orden: Contenidos de ordenes, precios y envíos.
- Productos: Descripción de productos en almacén, precios y cantidad en inventario.
- Usuarios
 - Cliente: Capaz de poder seleccionar productos y guardarlos en un carrito de compras. Posteriormente puede borrar y comprar los productos.
 - Administrador: Capaz de crear productos y usuarios.
 - Superadministrador: Mismas habiliades del Administrador, con la funcionalidad extra de poder crear otros Administradores.
- Autentificador: Validación de las credenciales de los usuarios para otorgar acceso a las diferentes funcionalidades del sistema.
- Landing Page
 - Para Clientes: Presentación de productos y direcciones solamente a componentes donde estos puedan hacer consultas y compras.
 - Para Administradores: Opción de poder crear usuarios o productos que se mostrarán en la landing page de los clientes.

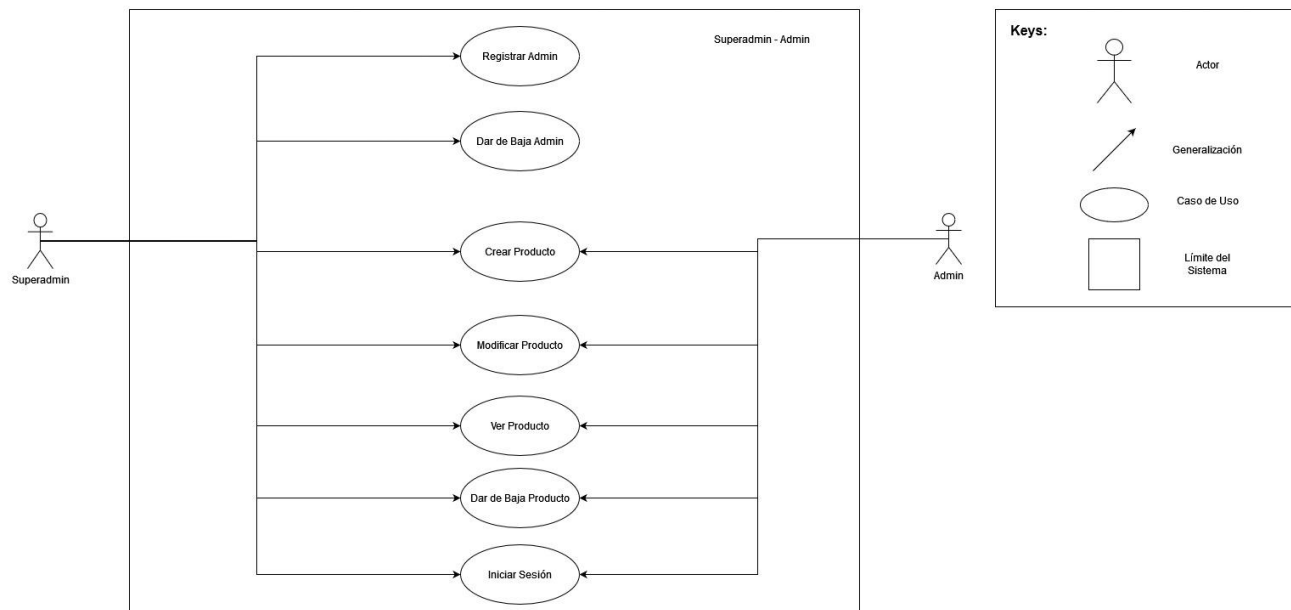
Respecto al motivo de nuestra selección, podemos decir que esta arquitectura nos da las siguientes ventajas:

- Altamente modificable y testable.
- Cada componente y view es independiente.

- Buena escalabilidad.
- Fácil implementación de servicios y APIs externas.

Por otra parte, es una arquitectura comúnmente utilizada en servicios de web apps porque es la misma estructura que siguen las metodologías sugeridas por frameworks de front-end como React, dónde creamos cada componente de forma aislada y las comunicaciones son a través de protocolos de prompts.

2.2 Vista de casos de uso



Casos de uso para los administradores

El superadmin puede realizar un registro de administrador y eliminar administradores, en un apartado del sistema donde tenga un listado para visualizarlos, esta sección solamente puede ser accedida por el superadmin al dar click en 'Users' en el dashboard. Por otra parte, tanto el admin o superadmin pueden crear, modificar, ver y cambiar estatus de los productos de la tienda a través de un listado, está lista aparece cuando el admin da clic en 'products' dentro de su dashboard.

Tanto el admin como el superadmin pueden hacer login usando sus respectivas credenciales registradas en el sistema.

Caso #1: Registro de Admin

Actores Primarios: Superadmin

Stakeholders e intereses:

- Gerente de tienda: Tener un control de las personas que tendrán el poder de administrar los datos del sistema, limitando quienes y cuantos disponen del control del sistema.
- Compañía: Quiere tener un registro de los supervisores que están dados de alta en el sistema para evitar que cualquier usuario tenga permisos administrativos.

Precondiciones: Debe de existir una cuenta de un superadministrador y que esté autenticado.

Garantía de éxito (postcondiciones): Un superadministrador verá disponible en la lista de usuarios al nuevo administrador, el cual fue guardado en el sistema.

Principal escenario de éxito:

- Un superadministrador se autentica en el sistema y tiene acceso al dashboard de administración
- El superadministrador se va a la página de “Users” dentro del dashboard.
- El superadministrador le da click al botón de “Create” en donde podrá crear a un nuevo administrador.
- El superadministrador indica el nombre de usuario y contraseña para el nuevo administrador.
- El nuevo administrador creado se ve en en la lista de usuarios dentro de la página de “Users”

Extensiones (o flujos alternativos):

1. En cualquier momento que no haya conexión a la base de datos:
 - a. Cancelar cualquier modificación que se quiera hacer en el sistema.
 - b. Mandar un mensaje que mencione la causa del error.
 - c. Redireccionar al usuario usando el sistema a la página principal.
2. En caso de haber puesto mal los datos del nuevo administrador:

Caso #1: Registro de Admin

- a. Se borran los datos de los cuadros de texto.
- b. Se manda un mensaje mediante una ventana flotante diciendo que los datos, ya sea el usuario o contraseña, no son válidos

Caso #2: Dar de baja admin

Actores Primarios: Superadmin


Stakeholders e intereses:

- Gerente de tienda: Tener control de las personas que tendrán el poder de administrar la tienda. Su interés principal es saber y tomar acciones sobre quién necesita ser dado de baja en el sistema como administrador.
- Supervisor: Mantener la seguridad del sistema, ya que si por ejemplo una persona administrador renunció, es necesario darlo de baja lo más pronto posible del sistema para evitar riesgos de seguridad. Ya sea que alguien tome la cuenta sin permiso y realice operaciones inadecuadas (cambiar precios de productos, eliminar usuarios, entre otros).

Precondiciones: Debe de existir una cuenta de un superadministrador y que esté autenticado.

Garantía de éxito (postcondiciones): Un superadministrador ya no verá disponible en la lista de usuarios al administrador, el cual fue deshabilitado en el sistema.

Principal escenario de éxito:

- Un superadministrador se autentica en el sistema y tiene acceso al dashboard de administración
- El superadministrador se va a la página de “Users” dentro del dashboard.
- El superadministrador le da click al botón de “Deshabilitar” ().
- El superadministrador ve la lista actualizada de administradores en donde el admin aparece con un check de deshabilitado.

Extensiones (o flujos alternativos):

3. En cualquier momento que no haya conexión a la base de datos:

- a. Cancelar cualquier intento de deshabilitar un usuario.
- b. Mandar un mensaje que menciona la causa del error.
- c. Redireccionar al usuario usando el sistema a la página principal.

Caso #3: Crear Producto

Actores Primarios: Superadmin y Admin

Stakeholders e intereses:

- Gerente de tienda: Dar de alta productos nuevos en el inventario y poder reflejar cualquier adición en el sistema.
- Supervisores: Dar de alta productos nuevos en el inventario y poder reflejar cualquier adición en el sistema.
- Compañía: Quiere poder visualizar cuando se agregan nuevos productos al sistema.

Precondiciones: Debe de existir una cuenta de un Superadministrador o Administrador y que este autenticado.

Garantía de éxito (postcondiciones):

En el caso de la creación de un producto, los nuevos datos que un administrador o super administrador pusieron sobre un producto se verán en el dashboard de administración y de igual manera en la página de la tienda se verán los datos del nuevo producto.

Principal escenario de éxito:

Creación de un producto

- a. Un superadministrador o administrador se autentica en el sistema y tiene acceso al dashboard de administración
- b. El usuario que esté autenticado en ese momento va a la página de “Products” dentro del dashboard para crear un producto.
- c. Para crear un producto nuevo se usa el botón de “Create”
- d. En el formulario que se presenta se le dan los datos de imagen, título, descripción, precio, tipo, color y existe en el inventario.

- e. Al confirmar los datos se crea el producto y se puede ver dado de alta en el sistema en la página de “Productos” del dashboard administrativo.

Extensiones (o flujos alternativos):

1. En cualquier momento que no haya conexión a la base de datos:
 - a. Cancelar cualquier alta que se quiera hacer en el sistema.
 - b. Mandar un mensaje que mencione la causa del error.
 - c. Redireccionar al usuario usando el sistema a la página principal.
2. Si el usuario requiere cancelar la operación de dar de alta el producto:
 - a. Dar click en el botón de “Cancelar” en el formulario de creación de producto.
 - b. Dar click en el botón “Confirmar” en la ventana emergente para completar la operación.

Caso 4#: Modificar producto

Actores Primarios: Admin y Superadmin

Stakeholders e intereses:

- Gerente de tienda: Tener control de los productos disponibles en el inventario y poder reflejar cualquier cambio en el sistema. Poder dar de alta a otros supervisores que puedan tener cierta administración de la tienda.
- Supervisores: Tener control de los productos disponibles en el inventario y poder reflejar cualquier cambio en el sistema.
- Compañía: Quiere tener un registro de los productos vendidos y quienes son los encargados de ellos en el sistema.

Precondiciones: Debe de existir una cuenta de un Superadministrador o Administrador y que este autenticado.

Garantía de éxito (postcondiciones): Los cambios o actualizaciones del producto generados por el administrador o superadministrador se verán en el dashboard de administración y de igual manera en la página de la tienda se verán los datos actualizados o el nuevo producto.

Principal escenario de éxito:

1. Modificación del estatus de un producto:
 - a. Para modificar un producto se usa el botón de “Edit”
 - Se muestra un formulario con los datos pre llenados que son los datos actuales de ese producto.
 - El usuario actual modifica los datos que requieran un cambio.
 - En la página de “Products” dentro del dashboard administrativo mostrará el producto con los nuevos datos.

Extensiones (o flujos alternativos):

4. En cualquier momento que no haya conexión a la base de datos para la modificación de algún producto:
 - a. Cancelar cualquier modificación o actualización que se quiera hacer en el sistema.
 - b. Mandar un mensaje que menciona la causa del error y lo reporta al sistema para que se pueda dar seguimiento del problema.
 - c. Redireccionar al usuario usando el sistema a la página principal y notificar que hubo un error en el sistema.

Caso #5: Ver Producto

Actores Primarios: Admin y Superadmin.

Stakeholders e intereses:

- Gerente de tienda: Tener control de los productos disponibles en el inventario y poder reflejar cualquier producto disponible en el momento. Como también poder dar de alta a otros supervisores que puedan tener cierta administración de la tienda.
- Supervisores: Tener control de los productos disponibles en el inventario en todo momento. Como también poder reflejar los cambios en los productos.

- Compañía: Quiere tener un registro de los productos disponibles a los clientes y quienes son los que los administran.

Precondiciones: Debe de existir una cuenta de un Superadministrador o Administrador y que esté autenticado.

Garantía de éxito (postcondiciones): Al darle click a ver productos se mostrará una lista con todos los productos registrados en el sistema.

Principal escenario de éxito:

1. Mostrar la lista de productos.
 - a. Para mostrar la lista se usa el botón "ver productos"
 - Se muestra una lista para los admins y superadmins de todos los productos registrados en el sistema.

Extensiones (o flujos alternativos):

1. En cualquier momento que no haya conexión a la base de datos para mostrar la lista de productos disponibles:
 - Mostrar un mensaje de error.
 - Redireccionar al usuario a la página principal.

Caso #6: Dar de Baja un producto

Actores Primarios: Admin y Superadmin.

Stakeholders e intereses:

- Gerente de tienda: Tener control de los productos disponibles en el inventario y poder reflejar cualquier baja de un producto.
- Supervisores: Tener control de las bajas de los productos en el sistema.
- Compañía: Quiere tener un registro de los productos que se dan de baja en el sistema

Precondiciones: Debe de existir una cuenta de un Superadministrador o Administrador y que esté autenticado.

Garantía de éxito (postcondiciones): Al darle click en el botón lista de productos y después a lado de cada artículo habrá un botón rojo, al darle click el producto desaparece de la lista. Todos los productos se verán actualizados en el dashboard de la tienda y en el de los admins y superadmins.

Principal escenario de éxito:

1. Dar de baja un producto
 - a. Para borrar un producto se da click en el botón de “borrar”
 - Cuando se le da click al botón de borrar en la lista de productos el admin dará de baja los productos que desea.
 - En la página de “Products” dentro del dashboard administrativo no estará el producto el cual se dio de baja.

Extensiones (o flujos alternativos):

1. En cualquier momento que no haya conexión a la base de datos para dar de baja un producto:
 - a. Cancelar cualquier baja o actualización que se quiera hacer en el sistema.
 - b. Mandar un mensaje que menciona la causa del error y lo reporta al sistema para que se pueda dar seguimiento del problema.
 - c. Redireccionar al usuario usando el sistema a la página principal y notificar que hubo un error en el sistema.

Caso #7: Iniciar sesión

Actores Primarios: Admin y Superadmin

Stakeholders e intereses:

- Gerente de tienda: Tener control de los admins y superadmins en el sistema.
- Supervisores: Tener control de las cuentas admin y superadmins en el sistema.

- Compañía: Quiere tener un registro de los usuarios que pueden administrar productos y usuarios en el sistema.

Precondiciones: Tener una cuenta con derechos de administrador previamente registrada.

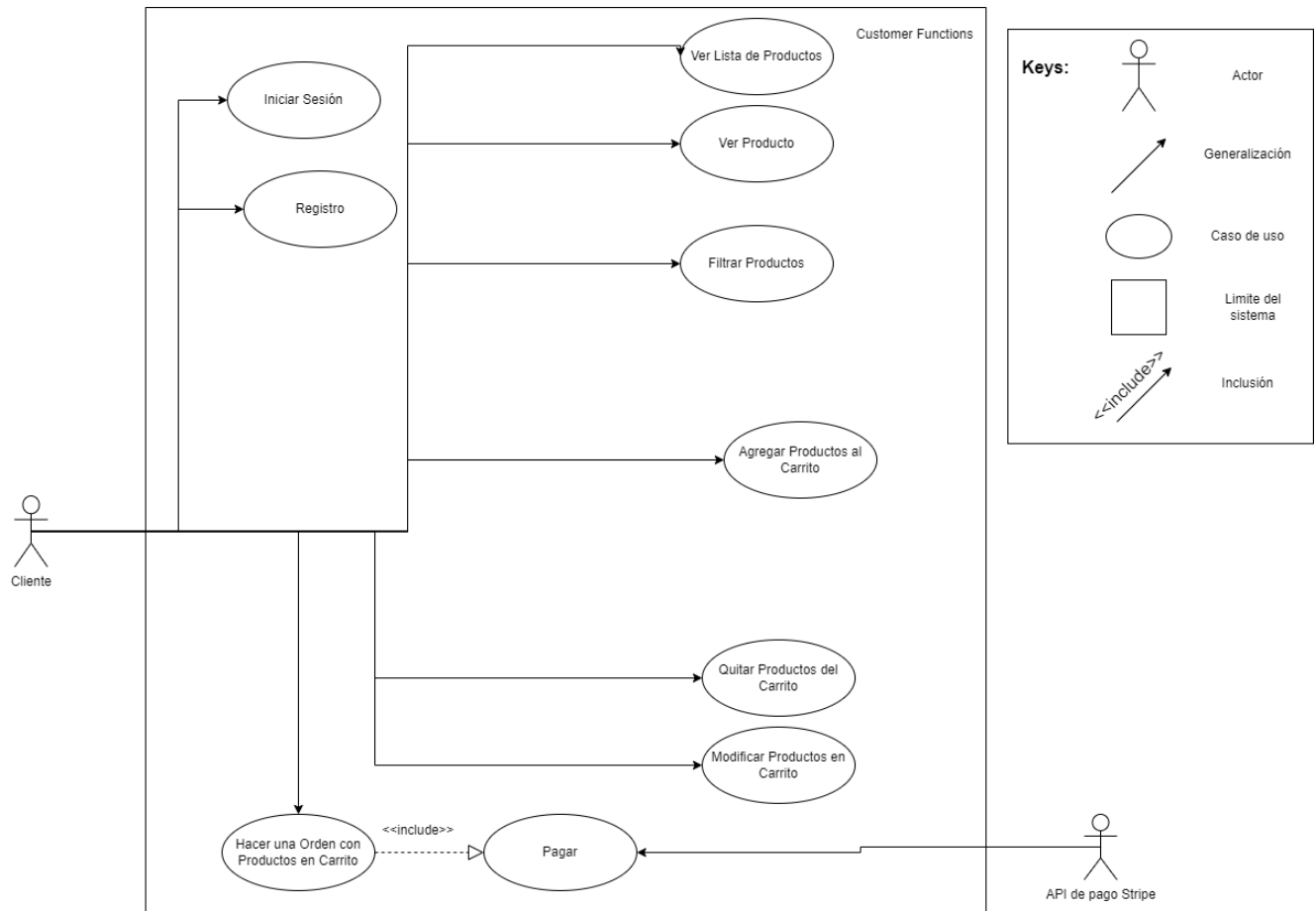
Garantía de éxito (postcondiciones): El usuario es direccionado a la dashboard de administrador una vez que ingrese sus datos de usuario y contraseña en la página de inicio de sesión.

Principal escenario de éxito:

1. Un admin o superadmin con una cuenta ya creada previamente da click al botón de “Login” para poder iniciar una sesión.
 - a. Introduce su usuario y contraseña en los campos y le da click al botón de “Login”
 - b. Se redirecciona al usuario al dashboard de administradores con su sesión iniciada.

Extensiones (o flujos alternativos):

1. En cualquier momento que no haya conexión a la base de datos para hacer el login:
 - a. Cancelar cualquier alta que se quiera hacer en el sistema.
 - b. Mandar un mensaje que menciona la causa del error.
 - c. Redireccionar al usuario usando el sistema a la página principal.
2. En caso de introducir mal los datos:
 - a. Se borran los datos de los cuadros de texto.
 - b. Se manda un mensaje mediante una ventana flotante diciendo que los datos, ya sea el usuario o contraseña, no son válidos



Caso #1: Iniciar Sesión

Actores Primarios: Cliente

Stakeholders e intereses:

- Cliente: Contar con una cuenta la cual pueda guardar información como dirección de envío o carrito de compras.
- Compañía: Saber cuántas personas están interesadas en comprar sus productos.

Precondiciones: Tener una cuenta previamente registrada.

Garantía de éxito (postcondiciones): El usuario es direccionado a la página principal una vez que ingrese sus datos de usuario y contraseña en la página de inicio de sesión.

Principal escenario de éxito:

- Un cliente con una cuenta ya creada previamente da click al botón de “Login” para poder iniciar una sesión.
- Introduce su usuario y contraseña en los campos y le da click al botón de “Login”
- Se redirecciona al cliente a la página principal de la tienda con su sesión iniciada

Extensiones (o flujos alternativos):

3. En cualquier momento que no haya conexión a la base de datos para mostrar la lista de productos disponibles:
 - a. Cancelar cualquier alta que se quiera hacer en el sistema.
 - b. Mandar un mensaje que mencione la causa del error.
 - c. Redireccionar al usuario usando el sistema a la página principal.
4. En caso de introducir mal los datos:
 - a. Se borran los datos de los cuadros de texto.
 - b. Se manda un mensaje mediante una ventana flotante diciendo que los datos, ya sea el usuario o contraseña, no son válidos

Caso #2: Registrarse

Actores Primarios:

Stakeholders e intereses:

- Cliente: Poder crear una cuenta la cual pueda guardar información como dirección de envío o carrito de compras.
- Compañía: Saber cuántas personas están interesadas en comprar sus productos.

Precondiciones: Tener en mente un usuario y contraseña para el registro, así como también una cuenta de correo.

Garantía de éxito (postcondiciones): El usuario es direccionado a la página principal una vez que ingrese sus datos de usuario, contraseña y correo electrónico en la página de registro.

Principal escenario de éxito:

- Un cliente que desea crear una cuenta da click al botón de “Register” para poder registrar una cuenta.
- Introduce su usuario, contraseña y cuenta de correo electrónico en los campos y le da click al botón de “Register”.
- Se redirecciona al cliente a la página principal de la tienda con su sesión iniciada

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos para mostrar la lista de productos disponibles:

1. Cancelar cualquier alta que se quiera hacer en el sistema.
 - a. Mandar un mensaje que menciona la causa del error.
 - b. Redireccionar al usuario usando el sistema a la página principal.
2. En caso de introducir mal los datos:
 - a. Se borran los datos de los cuadros de texto.
 - b. Se manda un mensaje mediante una ventana flotante diciendo que los datos, ya sea el usuario o contraseña, no son válidos

Caso #3: Filtrar productos

Actores Primarios: Cliente

Stakeholders e intereses:

- Gerente de tienda: Debe asegurar que los usuarios puedan filtrar los productos por categoría, color de producto y ordenarlos por precio. Ya que de esta forma brinda una mejor experiencia a los potenciales clientes y las ventas de la compañía podrían aumentar.

Precondiciones: El usuario puede haber hecho login o no, es una vista que se muestra al público en general.

Garantía de éxito (postcondiciones): El usuario ve los productos seleccionados por el filtro solamente.

Principal escenario de éxito:

- El usuario da click en 'ver productos' en el homepage.
- Selecciona algún filtro para los productos.
- La lista de productos se actualiza y solamente muestra aquellos que cumplen con el filtro aplicado.

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos:

Cancelar cualquier intento de filtro, se muestran todos los productos.

Caso #4: Agregar productos al carrito

Actores Primarios: Cliente

Stakeholders e intereses:

- Gerente de tienda: El encargado de la tienda está interesado en que los usuarios puedan agregar los productos de interés a su carrito, ya que solamente así pueden proceder a hacer una compra.

Precondiciones: El usuario está registrado y tiene su carrito asignado desde la base de datos.

Garantía de éxito (postcondiciones): El usuario ve que el icono de carritos se actualizó en cantidad y cuando da click en el icono del carrito pasa a otra vista donde se muestra el producto agregado.

Principal escenario de éxito:

- El usuario da click en "añadir al carrito" cuando está viendo un producto.
- El carrito se actualiza con el producto añadido.

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos:

No se agrega el producto al carrito y el usuario tiene que volver a intentarlo.

Caso #5: ver lista de productos

Actores Primarios: Cliente

Stakeholders e intereses:

- Gerente de tienda: El encargado de la tienda está interesado en que los usuarios puedan ver la lista de productos de tal manera que el cliente tenga una referencia de todos los productos que se venden en la página.

Precondiciones: El usuario está registrado y es redireccionado a la página principal.

Garantía de éxito (postcondiciones): El usuario le da click al botón de “ver todos los productos” y consecutivamente se mostrarán todos los productos registrados en el sistema.

Principal escenario de éxito:

- Mostrar la lista de productos.
 - a. Para mostrar la lista se usa el botón “ver productos”
 - Se muestra una lista para los clientes de todos los productos registrados en el sistema donde se podrá agregarlos al carrito y darles click para observar sus especificaciones .

Extensiones (o flujos alternativos):

2. En cualquier momento que no haya conexión a la base de datos para mostrar la lista de productos disponibles:
 - Mostrar un mensaje de error.
 - Redireccionar al usuario a la página principal.

Caso #6: Quitar Producto del Carrito

Actores Primarios: Cliente

Stakeholders e intereses:

- Cliente: Tener control sobre los productos que seleccionó en su carrito y remover uno si lo necesita y así tener una mejor experiencia de uso.
- Gerente de tienda: El encargado de la tienda está interesado en que los usuarios puedan controlar los productos de su carrito, ya que con esto les pueden garantizar que los clientes están ordenando lo que quieren.

Precondiciones: El usuario está registrado, tiene su carrito asignado desde la base de datos y tiene productos asignados a ese carrito.

Garantía de éxito (postcondiciones): Al acceder a su carrito el usuario, si lo desea, puede remover un producto de su carrito de compras. Este producto ya no aparecerá en la lista de productos en el carrito del usuario y el precio total del carrito disminuirá para reflejar de igual manera este cambio.

Principal escenario de éxito:

- El usuario le da clic al carrito en la pantalla.
- El usuario le da clic al botón “eliminar” que se encuentra al lado de cada producto listado en el carrito.
- El producto se quita de la lista de productos del carrito y se actualiza el precio total listado para este.

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos:

No se remueve el producto del carrito y el usuario tiene que volver a intentarlo.

Caso #7: Modificar Producto del Carrito

Actores Primarios: Cliente

Stakeholders e intereses:

- Cliente: Tener control sobre los productos que seleccionó en su carrito y modificar sus cantidades si lo necesita y así tener una mejor experiencia de uso.

- Gerente de tienda: El encargado de la tienda está interesado en que los usuarios puedan controlar los productos de su carrito, ya que con esto les pueden garantizar que los clientes están ordenando lo que quieren.

Precondiciones: El usuario está registrado, tiene su carrito asignado desde la base de datos y tiene productos asignados a ese carrito.

Garantía de éxito (postcondiciones): Al acceder a su carrito el usuario, si lo desea, puede cambiar la cantidad de un producto de su carrito de compras. Este producto aparecerá con la nueva cantidad en la lista de productos en el carrito del usuario y el precio total del carrito cambiará para reflejar de igual manera este cambio.

Principal escenario de éxito:

- El usuario le da clic al carrito en la pantalla.
- El usuario le da al botón desplegable que se encuentra al lado izquierdo de cada producto listado en el carrito y selecciona la cantidad que quiere dándole click a la opción deseada en el botón desplegable.
- Se le asignará la cantidad seleccionada al producto, se reflejará en la parte izquierda del producto y se actualizará el precio en el total listado del carrito.

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos:

No se cambia la cantidad del producto del carrito y el usuario tiene que volver a intentarlo.

Caso #8: Hacer una orden con productos del carrito

Actores Primarios: Cliente

Stakeholders e intereses:

- Cliente: Tener control sobre los productos que seleccionó en su carrito y poder utilizar esos productos para realizar una orden y posteriormente realizar el pago.

Precondiciones: El usuario está registrado, tiene su carrito asignado desde la base de datos y tiene productos asignados a ese carrito.

Garantía de éxito (postcondiciones): Al acceder al carrito el usuario es capaz de hacer una orden con los productos disponibles en ese momento en su carrito. Esto se logra si el botón de realizar orden está disponible al seleccionar los productos. De la misma manera se deberá de mandar a una pantalla donde se mostrará el checkout con los productos seleccionados.

Principal escenario de éxito:

2. El usuario da click a realizar orden
3. Se muestra el checkout de los productos de manera exitosa
 - a. Se muestra el total de los productos seleccionados
 - b. Se muestra los productos seleccionados
4. Se muestra el botón de pago de los productos
5. Se asigna una orden al cliente

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos o que la orden no se pueda realizar:

No se modifica el carrito de compras del cliente
Se regresa a la página de carrito del cliente

Caso #9: Pago

Actores Primarios: Cliente

Stakeholders e intereses:

- Cliente: Tener control sobre su método de pago a elegir y conforme a eso verificar que el cliente cuenta con una cuenta válida y proceder a pagar los productos que requiere.

Precondiciones: El usuario está registrado, tiene una orden asignada al cliente y se tiene un registro de la cantidad de productos a comprar y a qué precio.

Garantía de éxito (postcondiciones): Al acceder a la orden se va a poder observar por medio de la API Stripe la condiciones de pago y seleccionar el método de pago válido. Posteriormente la API se encargará de conectar con el banco de la tarjeta especificada y realizar el pago a la brevedad.

Principal escenario de éxito:

6. Seleccionar y especificar el pago de manera exitosa por medio de la API Stripe
7. Comprobar los datos especificados con el cliente con el banco por medio de la API Stripe
8. Realizar el pago con los datos especificados utilizando la API Stripe
9. Comprobar la transacción con la API Stripe

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos o que la orden no se pueda pagar

No se modifica el carrito de compras del cliente

Se regresa a la página de carrito del cliente

No se realiza ningún cobro al cliente

No se guarda ningún dato del cliente sobre su método de pago por confidencialidad

2.3 Escenarios Arquitectónicos y Requerimientos No Funcionales Asociados

2.3.1 Escenarios Arquitectónicos

Autenticación

Escenario #1 :

Usuario registra una nueva cuenta usando su correo electrónico o mediante los servicios

disponibles de Google y Facebook.

CRUD System

Atributo: Security

High - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | El usuario |
| Stimulus | El usuario ingresa datos como el nombre de la cuenta, contraseña y correo electrónico en caso de hacer un registro de usuario sin el uso de una red social externa, y posteriormente dar click en el botón 'Registrar'. En caso de usar Google o Facebook se sigue el método de registro implementado por el servicio. |
| Artifact | Módulo RegisterView.tsx |
| Environment | Operación normal |
| Response | RegisterView hace un POST request al endpoint "api/users/register" con los parámetros recibidos del usuario, ya sea por registro normal o por los APIs de Google o Facebook. Una vez que se valida, se redirecciona al usuario al homepage con su sesión iniciada y su respectivo token. |
| Measure response | Se recibe un status response 201 si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema al iniciar registro. |

Escenario #2 :

El usuario puede hacer login con su cuenta asociada y es redireccionado al Homepage.

CRUD System

Atributo: Security

High - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | El usuario |
| Stimulus | El usuario ingresa datos como el nombre de la cuenta y contraseña en caso de hacer un registro de usuario sin el uso de una red social externa, y posteriormente dar click en el botón 'Login'. En caso de usar Google o Facebook se sigue el método de registro implementado por el servicio. |
| Artifact | Módulo LoginView.tsx |
| Environment | Operación normal |
| Response | LoginView hace un POST request al endpoint "api/users/login" con los parámetros recibidos del usuario, ya sea por registro normal o por los APIs de Google o Facebook. Una vez que se valida, se redirecciona al usuario al homepage con su sesión iniciada y su respectivo token. |
| Measure response | Se recibe un status response 200 si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema al iniciar login. |

Carrito

Escenario #3:

Usuario agrega un producto al carrito de compras.

CRUD Cart

Atributo: Modifiability

High-High

| Portion of Scenario | Possible Values |
|---------------------|---|
| Source | El usuario |
| Stimulus | Al darle click al botón de "Add to cart" dentro de ProductView.tsx |
| Artifact | ProductView.tsx |
| Environment | Operación normal |
| Response | Al darle clic al botón, se guarda el producto y la cantidad de este hacia el carrito. En el símbolo del carrito que se encuentra en el componente de la Navbar.tsx, se verá actualizado con la nueva cantidad de productos agregados. |
| Measure response | Número actualizado de productos segundos después de darle clic al botón de "Add to cart" |

Escenario #4:

Usuario elimina un producto de su carrito de compras.

CRUD Cart

Atributo: Modifiability

High - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | Usuario |
| Stimulus | El usuario selecciona la opción de eliminar un artículo de su carrito |
| Artifact | CartView.tsx |
| Environment | Operación Normal |
| Response | Cuando se registra el click del botón se elimina el componente visual del producto en el carrito, se quita del registro de la orden y se reduce el total del precio del carrito en la cantidad del producto eliminado. |
| Measure response | Se actualiza la vista de los productos y el precio de la orden. |

Seguridad - Administración de tokens

Escenario #5:

User realiza una acción (ya sea user, admin o superadmin) y se verifica el token para validar el nivel de privilegio de la sesión.

Verify Token

Atributo: Security

Medium - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | Sistema |
| Stimulus | Al hacer una operación la cual solo cierto tipo de usuario tenga permiso a realizarla |
| Artifact | En el view actual |
| Environment | Operación normal |
| Response | Se valida o invalida la operación del usuario dependiendo del rol que tenga y de la operación que realice. |
| Measure response | Si es una operación inválida se envía un mensaje con la razón por la cual no pueda hacerla. Si es válida se envía un status con un código 200. |

Escenario #6:

El usuario puede visualizar los productos en su carrito de compras durante el checkout, y antes de confirmar puede remover productos del carrito.

Método Pago

Atributo: Usability

High - Medium

| Portion of Scenario | Possible Values |
|---------------------|-----------------|
|---------------------|-----------------|

| | |
|------------------|---|
| Source | El usuario |
| Stimulus | Al darle click al icono carrito para tener una vista detallada del mismo. |
| Artifact | En la view que se encuentre el usuario |
| Environment | Operación normal |
| Response | Se redirecciona a la view de CartView en donde se mostrarán los productos y el desglose del precio total. |
| Measure response | Se hace la redirección máximo 1 segundo. |

Usuario - navegación

Escenario #7:

El usuario puede navegar y filtrar productos por categoría (color, tipo).

User Views

Atributo: Usability

Medium - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | El usuario |
| Stimulus | El usuario selecciona múltiples filtros en los filtros 'color' y 'type' ProductListView.tsx |
| Artifact | ProductListView.tsx |
| Environment | Operación normal |

| | |
|------------------|---|
| Response | ProductListView pasa los params de filters al component Products.tsx, Products.tsx hace un GET request a “api/products?\$params” y este se encarga de actualizar la lista de productos. |
| Measure response | Se recibe un status response 200 si el proceso terminó satisfactoriamente, o un error status 500. |

Escenario #8:

El usuario podrá ver más detalles de un producto al darle click en su imagen.

User Views

Atributo: Usability

Medium - High

| | |
|---------------------|---|
| Portion of Scenario | Possible Values |
| Source | El usuario |
| Stimulus | Al darle click a la imagen del producto en ProductListView.tsx |
| Artifact | ProductListView.tsx |
| Environment | Operación normal |
| Response | El component Products.tsx se encarga de hacer un redirect al user hacia el path /product/:id y se carga el view ProductView.tsx |
| Measure response | El usuario es redireccionado sin errores. |

Escenario #9:

El usuario puede reordenar la lista de productos por precio y fecha de publicación de producto.

User Views

Atributo: Usability

Medium - High

| Portion of Scenario | Possible Values |
|---------------------|---|
| Source | El usuario |
| Stimulus | El usuario selecciona una opción de filtro en "Sort Products" en ProductListView.tsx |
| Artifact | ProductListView.tsx |
| Environment | Operación normal |
| Response | ProductListView pasa los params de filters al component Products.tsx, Products.tsx hace un GET request a "api/products?\$params" y este se encarga de actualizar la lista de productos. |
| Measure response | Se recibe un status response 200 si el proceso terminó satisfactoriamente, o un error status 500. |

Usuario - admin

Escenario #10:

El superadmin y admin puede crear productos que se van a almacenar en el sistema.

CRUD Admin

Atributo: Modifiability

High - High

| Portion of Scenario | Possible Values |
|---------------------|---|
| Source | El admin o superadmin |
| Stimulus | Ingresa la información válida para dar de alta a un nuevo producto y da click en 'create'. |
| Artifact | NewProduct.tsx |
| Environment | Operación normal |
| Response | NewProduct.tsx hace un POST request al endpoint "api/products/" con los parámetros recibidos. Una vez que se validan, se redirecciona al view ProductList.tsx donde puede ver el producto creado. |
| Measure response | Se recibe un status response 200 si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema al crear el producto. |

Escenario #11:

El superadmin y admin puede modificar productos, y las modificaciones son actualizadas en el sistema.

CRUD Admin

Atributo: Modifiability

High - High

| | |
|---------------------|--|
| Portion of Scenario | Possible Values |
| Source | El admin o superadmin |
| Stimulus | Ingresa la información válida para modificar un producto y da click en 'Update'. |
| Artifact | Product.tsx |
| Environment | Operación normal |
| Response | Product.tsx hace un PUT request al endpoint "api/products/:id" con los parámetros recibidos. Una vez que se valida, se redirecciona al view ProductList.tsx donde puede ver el producto actualizado. |
| Measure response | Se recibe un status response 200 si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema al modificar el producto. |

Escenario #12:

El superadmin y admin puede ver todos los productos disponibles en en el sistema.

CRUD Admin

Atributo: Usability

High - High

| | |
|---------------------|--------------------------------------|
| Portion of Scenario | Possible Values |
| Source | El admin o superadmin |
| Stimulus | El superadmin da click en 'Products' |

| | |
|------------------|---|
| | desde el Homepage. |
| Artifact | ProductList.tsx |
| Environment | Operación normal |
| Response | ProductList.tsx hace un GET request al endpoint "api/products/". Una vez que se valida, se cargan todos los productos en el componente Product. |
| Measure response | Se recibe un status response 200 si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema al modificar el producto. |

Escenario #13:

El superadmin y admin pueden deshabilitar (sin borrarlos del sistema) productos.

CRUD Admin

Atributo: Modifiability

High - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | El admin o superadmin |
| Stimulus | Da click en el icono de deshabilitar a un producto seleccionado |
| Artifact | ProductList.tsx |
| Environment | Operación normal |
| Response | ProductList hace un UPDATE request al endpoint "api/products/:id" y cambia el parámetro "available" a 'False'. |

| | |
|------------------|--|
| Measure response | Se recibe un status response 200 un mensaje que dice "Product unsubscribed" si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema. |
|------------------|--|

Escenario #14:

El superadmin es capaz de crear cuentas de administrador.

CRUD Admin

Atributo: Security

High - High

| Portion of Scenario | Possible Values |
|---------------------|--|
| Source | El superadmin |
| Stimulus | Ingresa la información válida para dar de alta a un nuevo administrador y da click en registrar. |
| Artifact | NewUser.tsx |
| Environment | Operación normal |
| Response | UserView.tsx hace un POST request al endpoint "api/auth/register" con los parámetros recibidos del superadmin. Una vez que se validan, se redirecciona al view UserList.tsx donde puede ver el admin creado. |
| Measure response | Se recibe un status response 201 si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema al crear el usuario. |

Escenario #15:

El superadmin es capaz de deshabilitar (sin borrar de base de datos) cuentas de administrador.

CRUD Admin

Atributo: Security

High - High

| Portion of Scenario | Possible Values |
|---------------------|---|
| Source | El superadmin |
| Stimulus | Da click en el icono de deshabilitar a un usuario seleccionado |
| Artifact | UserList.tsx |
| Environment | Operación normal |
| Response | UserList hace un UPDATE request al endpoint "api/users/:id" y cambia el parámetro "available" a 'False'. |
| Measure response | Se recibe un status response 200 un mensaje que dice "User unsubscribed" si el proceso terminó satisfactoriamente, o un error status 500 si hubo un problema. |

2.3.1 Requerimientos No Funcionales Asociados**Tamaño y Rendimiento**

| | |
|----------|--|
| RNF 0001 | Las solicitudes por el usuario se deben de completar en máximo 10 segundos |
|----------|--|

| | |
|---|---|
| Tiempo de solución de solicitudes | |
| La aplicación debe de resolver todas las solicitudes realizadas por el usuario en un máximo de 10 segundos, en caso de no ser así se debe de advertir al usuario que se tiene un problema con el rendimiento del backend de la aplicación | |
| RNF 0002 | El sistema deberá soportar al menos 100 usuarios concurrentes en la página |
| Número de usuarios simultáneos. | |
| La interfaz permite que sea ejecutada por más de 100 usuarios concurrentes en la página. Con esto se debe de alcanzar una continuidad en el servicio otorgado por el sistema para los usuarios. En caso de no ser posible soportar la cantidad de usuarios especificada, se debe de notificar al usuario que se están teniendo problemas de conexión. | |

| | |
|--|---|
| RNF 0003 | El sistema debe ser capaz de correr de forma eficiente en dispositivos móviles fabricados después del 2019. |
| Eficiencia del sistema en dispositivos móviles | |
| La interfaz permite que sea ejecutada por dispositivos móviles sin problema alguno, debe ser ágil en la transición de pantallas para un funcionamiento fluido. De la misma manera esta debe ser adaptada para que el usuario se le haga posible navegar de manera natural la plataforma. | |

| | |
|--|---|
| RNF 0004 | El sistema debe ser capaz de manejar 100 transacciones por segundo. |
| Manejo de transacciones concurrentes | |
| La interfaz debe ser capaz de manejar de manera ágil y eficiente 100 transacciones por segundo y actualizar en tiempo real estas transacciones en la base de datos en máximo 10 seg por cada una. Esto para otorgar una respuesta al usuario que su transacción fue correctamente realizada. | |

Usabilidad

| | |
|---|--|
| RNF 0005 | El tiempo de aprendizaje del sistema por un usuario deberá ser menor a 4 horas |
| Aprendizaje del sistema por usuario | |
| El sistema debe ser capaz de enseñar al usuario a utilizar el sistema en menos de 4 horas. De la misma manera el sistema debe ser intuitivo para que el usuario se adapte a las funcionalidades del mismo de manera ágil. | |

| | |
|--|---|
| RNF 0006 | El sistema debe de poseer interfaces gráficas intuitivas que permitan al usuario realizar operaciones en pocos segundos (menos de 10 segundos). |
| Estructuralización e implementación correcta | |
| El usuario podrá navegar sin tener muchos problemas entre las diferentes páginas al contar con una interfaz intuitiva, esto quiere decir que no confunda o sea difícil de entender. Si el usuario quiere hacer alguna operación dentro del sistema, se tienen que realizar en menos de 10 segundos para que el usuario sienta fluidez en el sistema. | |

| | |
|--|--|
| RNF 0007 | La página debe ser responsiva para múltiples dispositivos con distintas resoluciones (Escritorio, tableta, celular), esto se mide al observar que los elementos en la página se acomoden a la resolución del dispositivo actual. |
| Responsividad en todos los dispositivos | |
| El sistema debe ser capaz de adaptarse a cualquier resolución de dispositivo, ya sea algún dispositivo móvil o una PC. Esta responsividad debe de mostrarse de manera adecuada y adaptada a las necesidades del usuario. | |

Fiabilidad

| | |
|---|--|
| RNF 0008 | La interfaz debe estar disponible todos los días a través de un servicio de alojamiento de sitios web confiable. |
| Alta disponibilidad | |
| La dependencia de conectividad a través del proveedor de servicios de Internet es inminente para que los usuarios puedan acceder al sistema, además para alojar la página web para mantener acceso al sistema será un proveedor confiable conocido como Heroku. | |

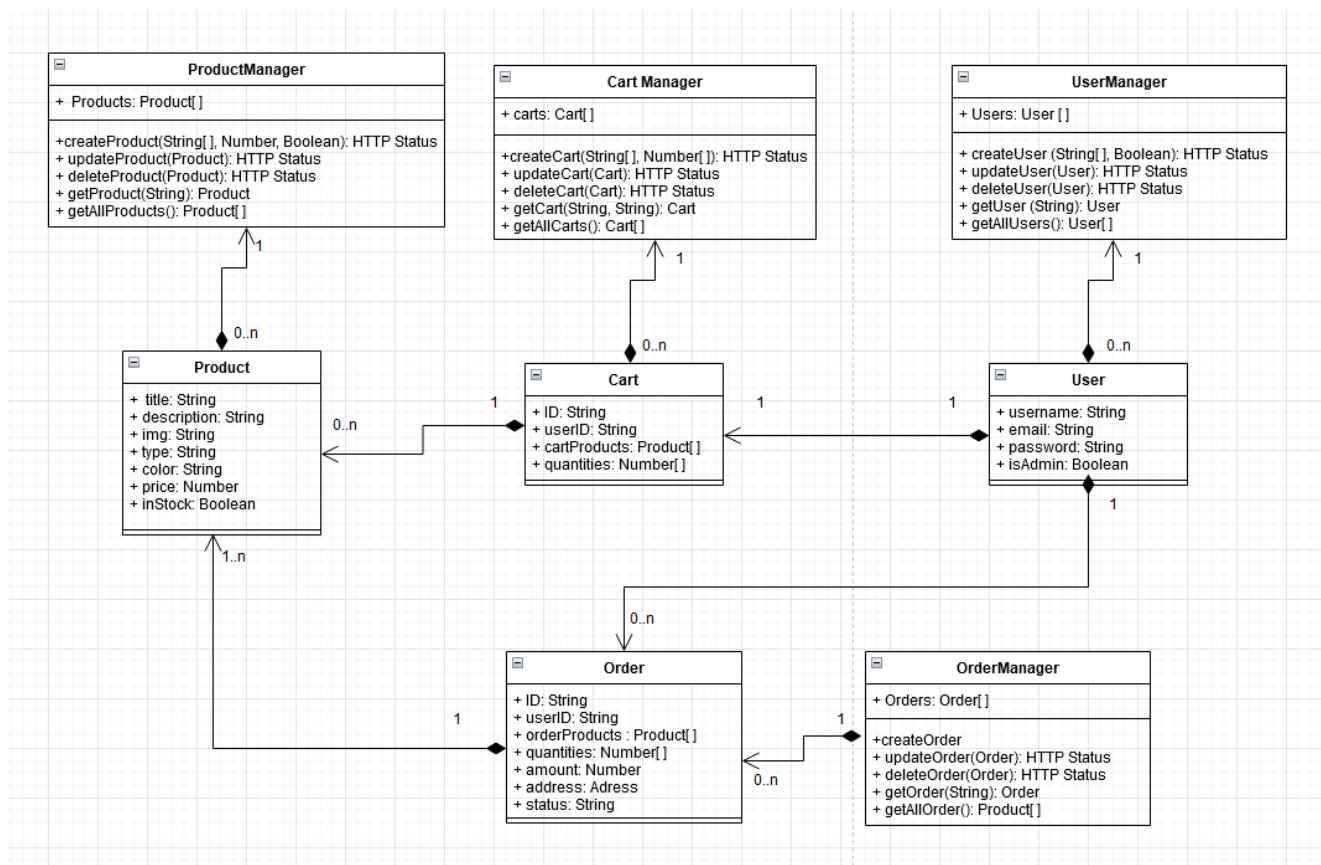
Seguridad

| | |
|--|---|
| RNF 0009 | Los pagos y su información deben ser encriptados mediante la API de Stripe. |
| Encriptación de información mediante Stripe | |
| Al realizar algún pago o movimiento que tenga que usar datos del usuario, esta debe ser asegurada con el API de Stripe para un funcionamiento adecuado y seguro de estos pagos. Esta herramienta nos otorga formas de encriptar la información del usuario y debe ser adaptada a las necesidades de la página. | |

| | |
|--|---|
| RNF 0010 | Las contraseñas para la creación y autenticación de usuarios deberá ser encriptada. |
| Encriptación de contraseñas al iniciar sesión | |
| Al realizar alguna creación o modificación de contraseña del usuario, esta debe ser encriptada para que el sistema tenga la seguridad de evitar filtraciones de información. De la misma manera al autenticar un token debe ser acoplado al usuario para el uso seguro de la plataforma. | |

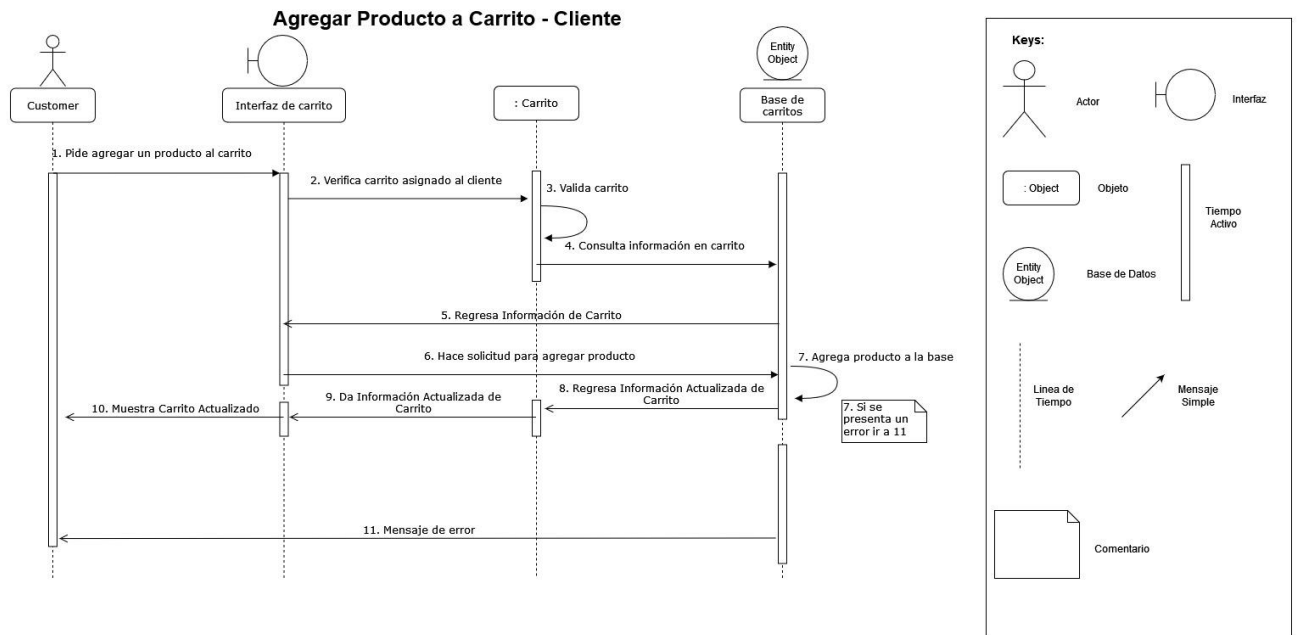
| | |
|--|--|
| RNF 0011 | Al iniciar sesión se genera un token para el usuario. |
| Creación de token en inicio de sesión | |
| <p>Quando un usuario inicia sesión con el usuario y contraseña correctos, se crea un token el cual le permitirá navegar en la aplicación. Este mismo token es el que le da los permisos a las funcionalidades del sistema. Si se trata de un administrador o superadministrador, tendrán permiso a la administración de la página. Mientras que un usuario cliente solo podrá comprar productos.</p> | |

2.3 Vista lógica



2.4 Vista de proceso

Utilizar un diagrama de secuencia o de actividad para mostrar las interacciones



Secuencia #1: Agregar producto a carrito

Actores Primarios: Cliente


Stakeholders e intereses:

- Cliente: Agregar productos de su interés al carrito de compras y verificar que todos los productos se muestren correctamente
- Compañía: Vender sus productos a clientes que los encuentren útiles o interesantes..

Precondiciones: El usuario debe de estar registrado y contar con la lista de productos que considera necesarios

Garantía de éxito (postcondiciones): El usuario debe ser capaz de añadir los productos que requiera en el momento que los requiera a su carrito para tener la lista de los mismos

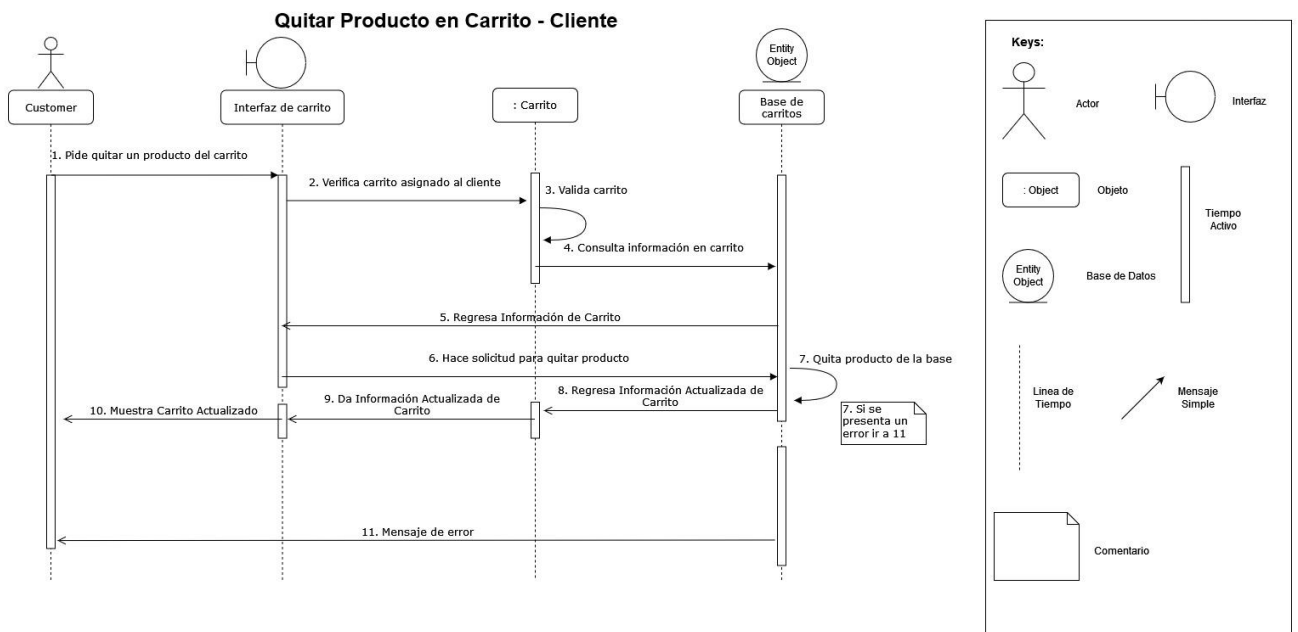
Principal escenario de éxito:

- El producto que visualiza el cliente es añadido por medio del botón “Añadir a carrito de compras”
- Dar click al botón del carrito de compras () para ir a la página de checkout.
- Puede visualizar en cualquier momento los productos que tiene agregado el cliente

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos o se agotan existencias del producto

El cliente regresa a la página del producto
No se realiza ningún cambio a la base de datos.
Se muestra un mensaje de error del sistema hacia el usuario
Se notifica al cliente del cambio del producto



Secuencia #2: Quitar Producto en Carrito

Actores Primarios: Cliente


Stakeholders e intereses:

- Cliente: Modificar su carrito de compras en caso de que vea un error.
- Compañía: Dar una experiencia de compra más amena a sus cliente

Precondiciones: El usuario debe de estar registrado y tener productos en el carrito de compras.

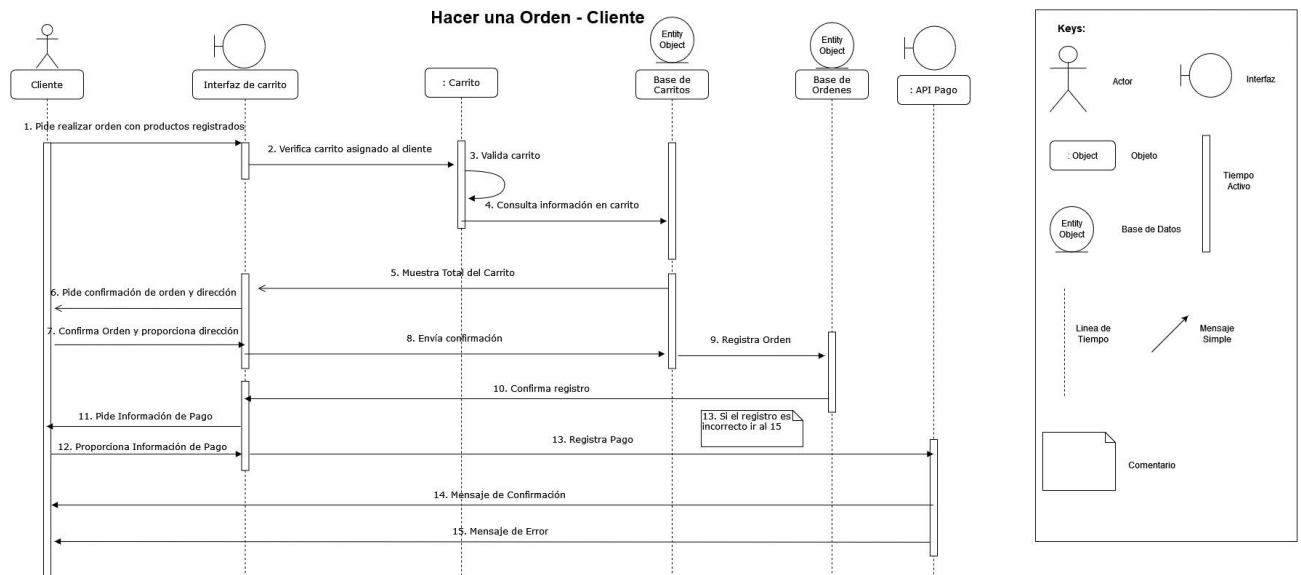
Garantía de éxito (postcondiciones): El usuario ve que al dar click al botón de “Remove” se quitan los productos que desea remover de su carrito de compras.

Principal escenario de éxito:

- Dar click al botón del carrito de compras () para ir a la página de checkout.
- Corroborar que los productos en el carrito sean los que se desean comprar
- En caso de querer quitar un producto se puede dar click al botón de “Remove” para quitarlo del carrito de compras
- La lista de productos del carrito se actualiza y se observa que el producto que se quiso remover ya no aparece.

Extensiones (o flujos alternativos):

1. En cualquier momento que no haya conexión a la base de datos:
 - d. Cancelar cualquier intento de deshabilitar un usuario.
 - e. Mandar un mensaje que menciona la causa del error.
 - f. Redireccionar al usuario usando el sistema a la página principal.



Secuencia #3: Hacer una orden

Actores Primarios: Cliente


Stakeholders e intereses:

- Cliente: Comprar productos que le fueron de su interés.
- Compañía: Vender sus productos a clientes que los encuentren útiles o interesantes.

Precondiciones: El usuario debe de estar registrado, tener productos en el carrito de compras y contar con una forma de pago para realizar el pedido con su orden.

Garantía de éxito (postcondiciones): El usuario debe ser capaz de realizar el pago de una orden de productos de su carrito, de la misma manera poder ingresar los datos bancarios y finalmente comprobar que su orden es correcta y proceder con el pago de sus productos.

Principal escenario de éxito:

- Dar click al botón del carrito de compras () para ir a la página de checkout.
- Corroborar que los productos en el carrito sean los que se desean comprar
- Si todo está en orden, dar click al botón de “Pagar”

- Introducir el método de pago para que la API de Stripe se encargue del proceso.

Extensiones (o flujos alternativos):

En cualquier momento que no haya conexión a la base de datos o que no haya conexión con la API de Stripe:

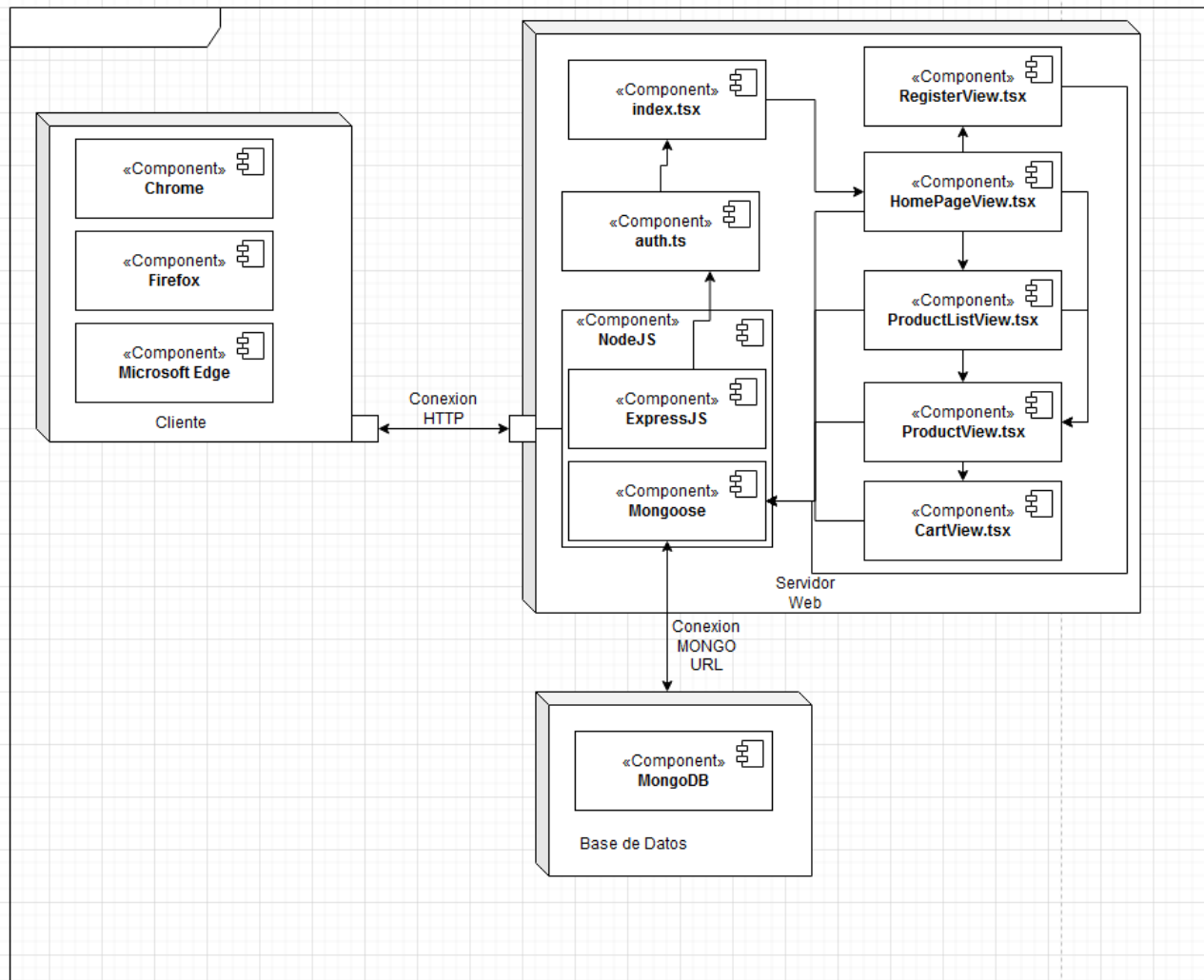
El cliente regresa a la página de su carrito con los productos especificados antes del pago

No se registra ningún cargo al cliente hasta que vuelva a realizar el proceso.

No se registra ningún dato bancario del cliente

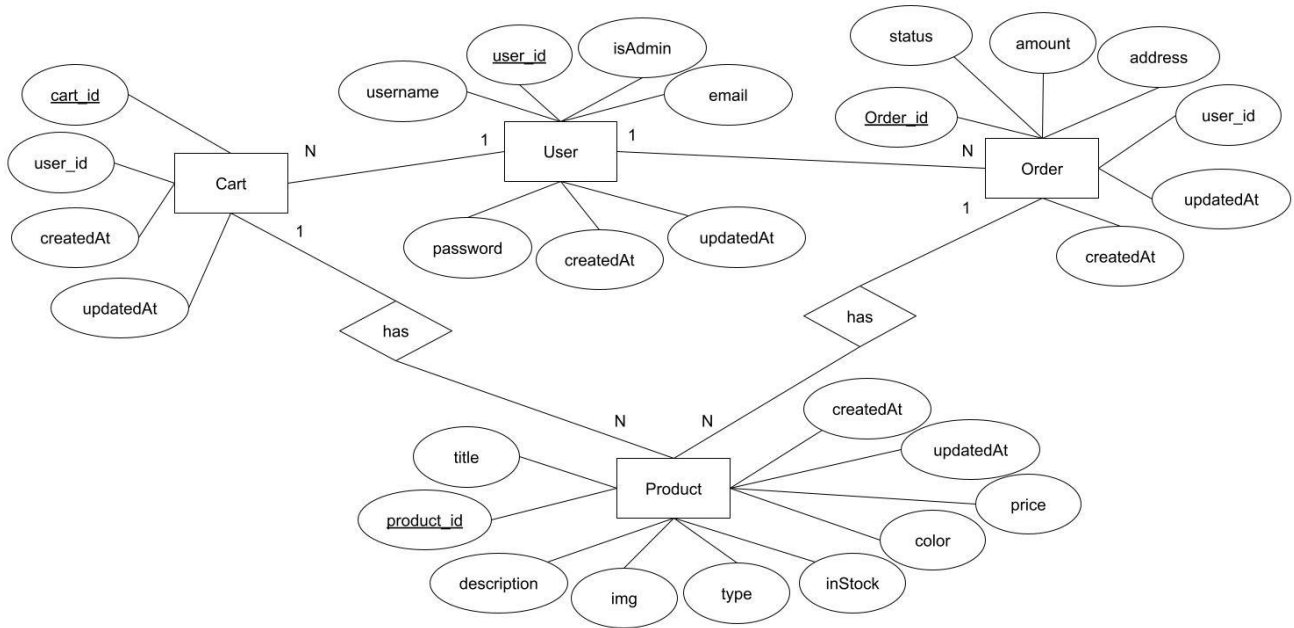
2.5 Vista de implementación

Descripción del diagrama de despliegue



2.6 Vista de datos

Incluir el diagrama entidad-relación



2.6.1 Base de datos

El back-end utiliza una base de datos NoSQL - MongoDB.

User

| Nombre de Columna | Tipo de Dato | Permite Null? | Descripción |
|-------------------|--------------|---------------|--|
| username | String | No | Nombre del usuario registrado en el sistema. |
| email | String | No | Correo Electrónico del usuario registrado en el sistema. |
| password | String | No | Contraseña del usuario registrado en el sistema. |
| isAdmin | Boolean | Sí | Indica si el usuario |

| | | | |
|--|--|--|---|
| | | | está registrado como administrador si es verdadero, y si es falso indica que es un usuario regular. |
|--|--|--|---|

Product

| Nombre de Columna | Tipo de Dato | Permite Null? | Descripción |
|-------------------|--------------|---------------|---|
| title | String | no | Título o nombre del producto. |
| description | String | no | Descripción del producto. |
| img | String | no | URL al path de la imagen. |
| type | String | yes | Tipo de producto, puede ser "pc-parts", "pc-cases", "pc-add-ons" |
| color | String | yes | Color del producto, puede ser "rgb", "neutrals", "pastel-colours" |
| price | Number | no | Precio del producto en dls. |
| inStock | Boolean | no | Booleano para declarar si el producto está disponible o no. |

Order

| Nombre de Columna | Tipo de Dato | Permite Null? | Descripción |
|-------------------|---------------------------|---------------|---|
| userID | String | No | ID en base de datos del usuario registrado en el sistema al que corresponde esta orden. |
| products | Array de <String, Number> | Sí | Lista de <product_id, amount>, donde 'amount' es la cantidad de productos con cierto ID agregados a la orden. |
| amount | Number | No | Precio total de la orden a pagar por el usuario. |
| address | Objeto | No | Dirección a la que se enviará la orden. |
| status | String | Sí | Estado de la orden en el sistema, puede ser: "Aceptada", "En proceso" o "Cancelada" |

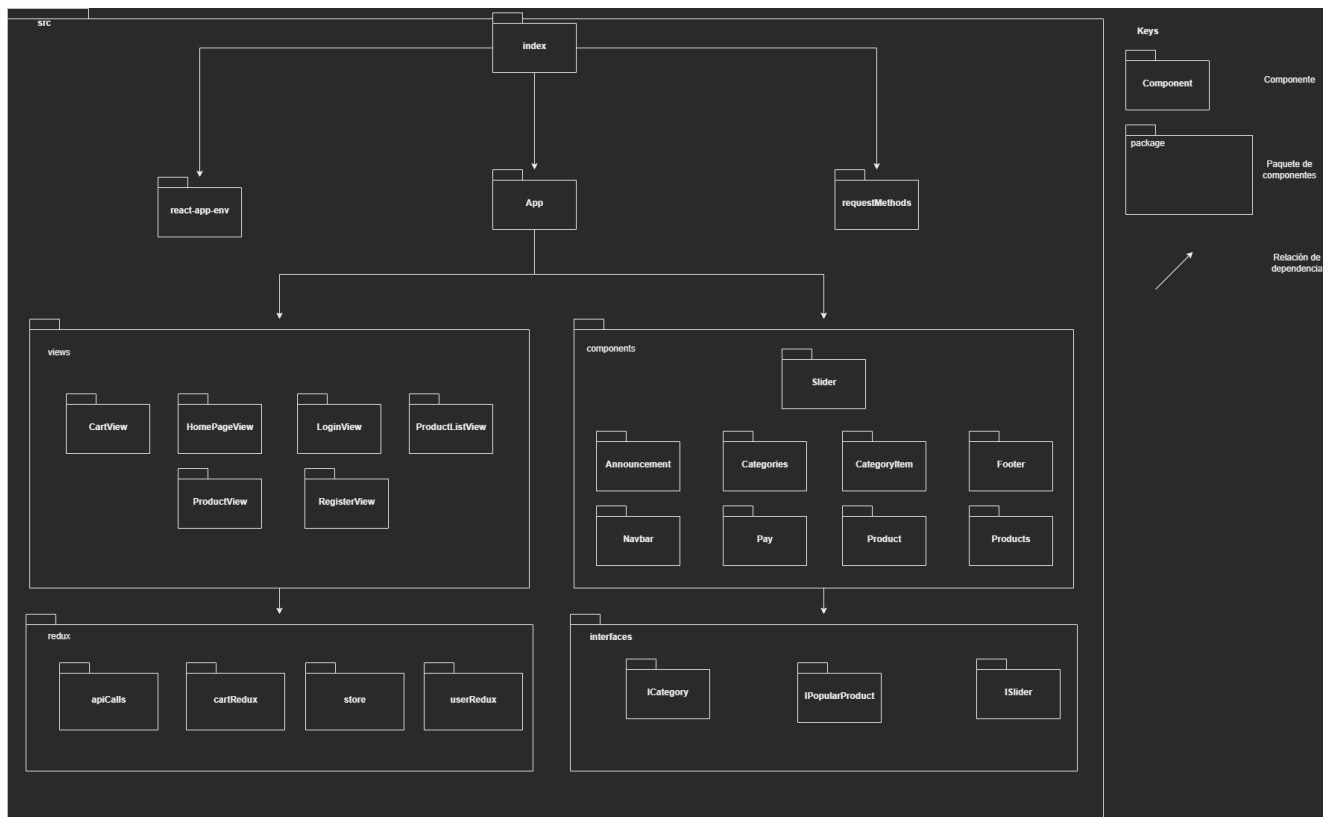
Cart

| Nombre de Columna | Tipo de Dato | Permite Null? | Descripción |
|-------------------|---------------------------|---------------|---|
| userID | String | No | ID del usuario asignado al carrito. |
| products | Array de <String, Number> | Sí | Lista de <product_id, amount>, donde 'amount' es la |

| | | | |
|--|--|--|---|
| | | | cantidad de productos con cierto ID agregados al carrito. |
|--|--|--|---|

2.7 Vista de layers

Descripción del diagrama de layers



En la capa de layers para el front-end de la webapp identificamos que index es el componente principal que llama a App, así como hay otros elementos esenciales del proyecto, como react-app-env que son las variables de entorno globales para la página web y requestMethods que contiene dos estructuras donde se declara la ruta principal de requests permitidos para un visitante sin registrarse y un usuario registrado.

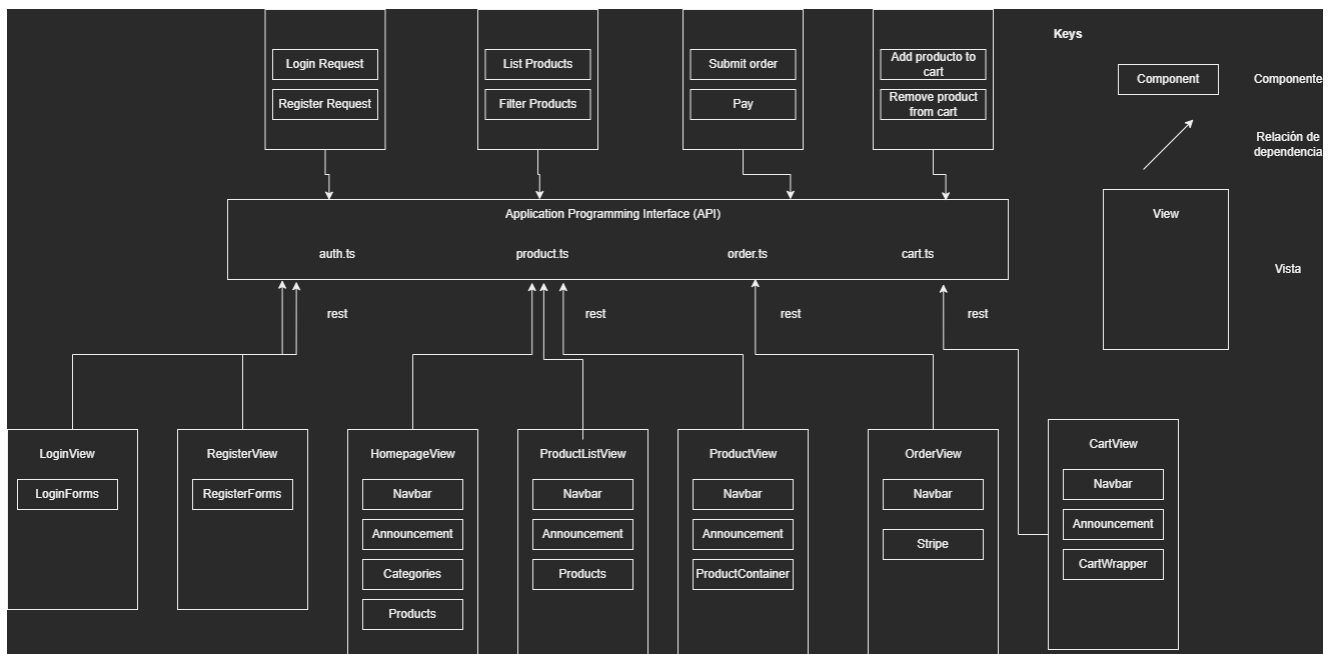
App.tsx es el componente que tiene las rutas hacia los otros views, así como mantener una condición que identifica si un usuario está login o no. Dentro de views están cada vista que el usuario podría ver, podemos identificar que está el Homepage, LoginView, RegisterView entre otros. Por otra parte, están los components de React, que son usados en su mayor parte por los views, ya que estos son dinámicos y cambian de acuerdo a la interacción con el usuario.

Redux es un administrador de estados, principalmente se encarga de mantener los estados del usuario, de los productos y el carrito de compras, además, hay un componente que se llama ApiCalls que es un wrapper para las funciones que llaman a los endpoints del back-end.

Finalmente interfaces tiene las estructuras esenciales del proyecto, en ellas se definen los tipos de objetos que utilizan algunas views, así como sus variables.

2.8 Diagrama de microservicios

Descripción del diagrama de microservicios



Básicamente las diferentes funcionalidades de negocio son recibidas por el API de nuestro sistema que interactúa con los diferentes módulos de funcionalidad que se encuentran dentro de las views del sitio web.

3. Lineamientos Arquitectónicos

3.1 Diseño

El diseño arquitectónico por el que optamos usar en nuestro proyecto fue el de Modelo-Vista-Controlador (MVC). Se caracteriza por contar con 3 capas o componentes distintos. Esto se hace con el objetivo de poder reutilizar código, además de poder separar las tareas. Esto hace que el desarrollo de un proyecto sea ordenado y también de fácil mantenimiento.

La estructura y fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores (Models, Views & Controllers).

En la capa de **Modelos** es donde se encuentra la lógica de negocios, la persistencia en la base de datos y en donde se manipulan los datos que se estarán utilizando en la aplicación.

En esta capa de nuestra aplicación se pueden encontrar métodos o funciones permite agregar productos al carrito de compras, así como también poder calcular el precio total de un carrito tomando en cuenta los precios individuales de los productos. Igualmente contamos con las queries que se realizan a la base de datos para recuperar los productos que se encuentren en inventario y mostrarlos en la aplicación, de la misma manera tenemos las funcionalidades de poder cambiar el estatus de un producto para indicar que no se encuentra en el inventario. En la parte de los usuarios contamos con los métodos de creación, modificación y suspensión de cualquier usuario.

En la capa de **Vistas** es donde se presenta la interfaz la cual el usuario podrá interactuar y proveer de datos los cuales podrán ser usados en el modelo de la aplicación. Estas vistas pueden llegar a ser desde líneas de comando en una consola, hasta interfaces gráficas más complejas y completas en donde es más intuitivo para el usuario el manejo del sistema.

En nuestra aplicación la capa de **Vistas** fue realizada con Typescript en conjunto de React.js para poder mostrar los productos que se encuentran dados de alta en la base de datos

La capa de **Controladores** sirve como un intermediario entre la capa de **Vistas y Modelos**. Se encarga de regular el flujo de la información y adapta la información para que tanto el Modelo como la Vista puedan funcionar correctamente.

En nuestro proyecto esto se puede ver cómo React hace un enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra

aplicación. Estos controladores sirven solamente de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

Al utilizar un diseño arquitectónico como lo es MVC obtenemos varias ventajas como lo es el orden en nuestro código, así como también facilidad de mantenimiento cuando se requiere modificar o agregar alguna nueva funcionalidad.

3.1 Codificación

Para nuestra codificación estamos usando la metodología de notación Camell de código fuente, que se basa en que el primer carácter de todas las palabras, excepto de la primera palabra se escribe en mayúsculas y los otros caracteres en minúsculas. Intentamos usar nombres de variables entendibles y descriptivas, sin importar que tan largo sea el nombre de la variable. Esto nos ayuda a saber el uso de la variable que asignamos o el método que llamamos.

Al momento de utilizar prefijos como boolean debemos de escribir “es” en inglés “is” para entender que esta variable o método regresa un booleano. Intentamos que cada uno de nuestros métodos o clases que utilizamos deban coincidir con el nombre del archivo que asignamos y no añadir métodos de más que no tengan que ver con la clase o archivo. Esto nos ayuda a tener un control para cuando se vayan a realizar las pruebas de la funcionalidad del sistema.

En caso de los inputs de los usuarios intentamos siempre validar si el campo no está vacío y tiene los campos necesarios para continuar la operación. Debemos de asumir que cualquier cosa puede salir mal por lo que en nuestro código tenemos condiciones para asegurar el funcionamiento correcto del sistema.

En nuestra implementación de los métodos y componentes de nuestra aplicación evitamos completamente la utilización de palabras reservadas de javascript y typescript para evitar la confusión al momento de buscar y obtener los componentes que nosotros escribimos, estos están estrictamente definidos con una relación a su representación en la vida real.

3.2 Reutilización de Software

- **React**

React nos permite definir diferentes componentes gráficos con funcionalidad asignada que podemos codificar en archivos propios e importarlos a las diferentes páginas de nuestra aplicación que los requieran y así agilizar nuestra creación del front-end.

- **Yarn**

Esta aplicación nos permite descargar e instalar diferentes paquetes programáticos que se utilizarán para añadir funcionalidad externa a nuestra aplicación.

- **Axios**

Este paquete nos permite hacer peticiones de http desde node js de forma sencilla para poder conectarnos a nuestro Back-end con mayor facilidad y fluidez.

- **Mongoose**

Este paquete nos ayuda a traducir la funcionalidad de MongoDB a un entorno web. Esto nos permitirá crear modelos para las diferentes colecciones de este sistema de bases de datos que representarán a los datos que guardamos y también nos facilitará realizar las diferentes operaciones CRUD para manipular los datos guardados como lo necesitemos en la aplicación.

- **Express**

Este paquete nos permite manejar las peticiones dentro de nuestra aplicación y el ruteo del mismo como middleware. Es algo esencial para poder realizar nuestra aplicación y funcione de manera correcta para el cliente con las rutas establecidas.

- **Jsonwebtoken**

Este paquete nos sirve para manejar nuestras sesiones con más seguridad y encriptar nuestros datos del usuario cuando inicia sesión o realiza alguna modificación para mandar a nuestra base de datos.