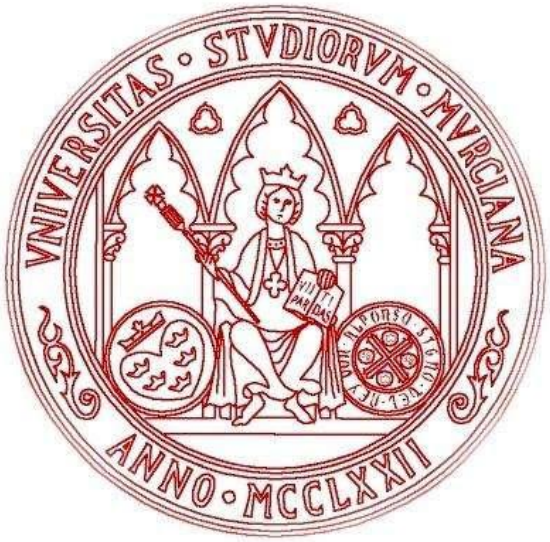

UNIVERSIDAD DE MURCIA
Facultad de Informática

App para pedidos de impresión

Trabajo de fin de grado

Rubén Sánchez Fernández
21067018F

Tutor
Francisco García Sánchez

Convocatoria de Junio 2024

Índice general

Índice de figuras

1. Resumen

Este es el resumen... ESTILO DE REDACCIÓN APLICABLE A TODO EL DOCUMENTO: evitaremos el uso de la primera persona; en su lugar se suele emplear el reflexivo ("hemos desarrollado" "se ha desarrollado").

2. Extended abstract

Resumen extendido en inglés (2000 palabras). En el abstract conviene replicar de algún modo la estructura de la memoria en sí: (i) introducción, (ii) estado del arte, (iii) objetivos y metodología, (iv) diseño y resolución del trabajo, (v) conclusiones y trabajo futuro. El contenido del abstract tiene que abarcar todos esos conceptos en ese orden.

3. Introducción

Esto se escribe lo último, junto con el resumen y el abstract. Se establece el contexto en el que se sitúa el proyecto introduciendo claramente la problemática e indicando el objetivo general del proyecto (como consecuencia de esos problemas que se quieren resolver). Suele venir acompañado de numerosas referencias bibliográficas relevantes sobre los distintos conceptos tratados. En el último párrafo de la introducción hay que indicar la estructura/organización del resto del documento (un párrafo indicando brevemente en qué secciones se ha dividido el documento y el contenido de cada sección).

4. Estado del arte

En este apartado se expondrá por un lado el contexto inicial y la situación de partida junto con el problema a solucionar que se nos plantea. Por otro lado se realizará un análisis de las herramientas y tecnologías que hemos decidido emplear para el desarrollo de la solución al problema.

4.1. Contexto y análisis de la situación de partida

Este trabajo de Fin de Grado, ha sido desarrollado en colaboración con Upango, empresa especializada en transformaciones digitales B2B. Durante años, Upango, ha trabajado con uno de los proveedores de soluciones de tiendas online más populares del mundo como es ePages, desarrollando aplicaciones y tiendas online a medida para los clientes en el ámbito del comercio electrónico. A lo largo del tiempo, se ha observado en esta empresa, una creciente demanda de incorporar al desarrollo de tiendas personalizadas, funcionalidades que permitan a los clientes personalizar los productos para su posterior compra, es decir en el mercado actual hay una gran cantidad de clientes potenciales que podrían solicitar desarrollos de comercio electrónico a medida con funcionalidades de personalización de productos.

Recientemente se ha decidido migrar de ePages a otra plataforma de comercio electrónico como es Shopify que ofrece otras tecnologías y herramientas para el desarrollo personalizado de tiendas online. Debido a esta migración, los desarrollos a medida y aplicaciones estandarizadas de la otra plataforma quedan obsoletas para nuevos desarrollos y surge la necesidad de adaptarse a estas nuevas herramientas y funcionalidades que nos proporciona Shopify para conseguir crear estos desarrollos de comercio a medida. Por lo que si bien en las tiendas de ePages de los actuales clientes existen ya desarrollos y funcionalidades para la personalización de artículos, la transición a Shopify requiere la creación de una nueva solución y adaptación para esta plataforma.

Debido a la necesidad que se crea propulsada por este gran cambio, nuestro objetivo con este proyecto es crear una aplicación para la plataforma Shopify que sea versátil y escalable y permita integrarse en las tiendas de nuestros nuevos clientes para ofrecerles la capacidad de proporcionar experiencias de compra de productos personalizados, manteniendo nuestro compromiso con la excelencia en el desarrollo de soluciones digitales para comercio electrónico.

4.2. Herramientas y tecnologías empleadas

En el desarrollo de aplicaciones web modernas, concretamente en la creación de tiendas personalizadas, la selección de los proveedores de soluciones y las tecnologías apropiadas desempeña un papel fundamental para garantizar el desarrollo de soluciones efectivas y personalizadas para los clientes. En este contexto, la decisión más trascendental ha sido la de seleccionar el mejor proveedor de soluciones, en nuestro caso **Shopify**.

Es importante destacar que la elección de parte de las tecnologías ha estado fuertemente influenciada por nuestra decisión de emplear Shopify como base para nuestros desarrollos, pues debido a esto la búsqueda de las tecnologías necesarias para desarrollar el proyecto se han limitado a tecnologías que se integran y están aceptadas sin problemas con la plataforma para conseguir aprovechar al máximo sus capacidades, si bien es cierto que Shopify es una plataforma que está en continua mejora y crecimiento y no se queda atrás en cuanto a las tecnologías.

La plataforma **Shopify**, con su amplia gama de herramientas y funcionalidades para el comercio electrónico, proporciona una base sólida para el desarrollo de tiendas en línea personalizadas. Originaria de Canadá y fundada en 2006, esta se ha convertido en uno de los proveedores líderes en el mercado de comercio electrónico, atendiendo a millones de comerciantes en todo el mundo.

Cuenta con multitud de ventajas frente a otras plataformas de comercio electrónico muy populares en el mercado. Una de las ventajas mas trascendentales y que ha influenciado fuertemente la decisión de Upango de migrar de **ePages** a esta tecnología, es el hecho de que Shopify elimina las barreras iniciales al ofrecer un servicio completo que incluye registro de dominio y hosting web ilimitado. Esto nos simplifica el proceso de lanzamiento y mantenimiento de una tienda al eliminar la necesidad de buscar proveedores de hosting externos, además de dotarnos de tecnologías avanzadas como servidores de alta calidad, redes optimizadas y un CDN global para garantizar una velocidad de carga rápida. Esto no solo mejora la experiencia del usuario, sino que tam-

bién tiene un impacto positivo en el posicionamiento SEO y en la tasa de conversión. Cabe destacar que Shopify cuenta con una tasa de conversión de las más elevadas del mercado. [1]

Una de las características con las que cuenta Shopify, es la habilidad de poder crear aplicaciones que puedan ampliar las capacidades existentes de esta plataforma. Esto nos permite agregar funcionalidades a las tiendas, ampliar la experiencia de la parte de administración o crear experiencias de compra únicas para nuestros clientes. Gracias a esto los usuarios pueden instalar estas aplicaciones para ayudar a desarrollar su negocio, integrarse con servicios externos y agregar funciones a su panel de control de Shopify. [2]

4.2.1. Frontend

En cuanto al frontend de nuestra aplicación podemos observar dos partes. Por un lado tenemos el frontend asociado a la parte de Administración de Shopify, esta parte hace referencia a la interfaz del usuario que los administradores de la tienda emplean para gestionar y controlar los diversos aspectos de la misma, como la gestión de productos, pedidos, clientes, configuración de temas, traducciones y otros muchos aspectos de configuración de la tienda. Al instalar nuestra aplicación en las tiendas que necesiten de sus funcionalidades esta parte del front se integrará con la página de administración de cada tienda, proporcionando la interfaz y las funcionalidades que configuremos en esta parte del desarrollo. En la Figura 1 podemos observar la interfaz de administración de una tienda Shopify en desarrollo.

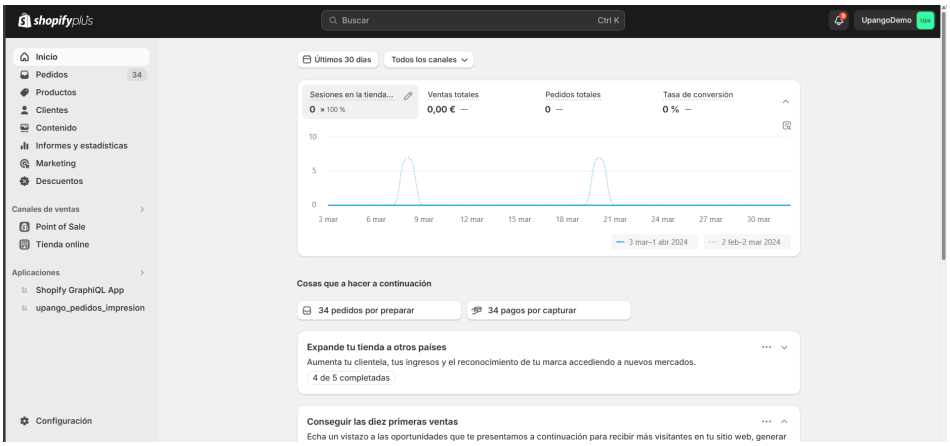

Figura 1: Interfaz de administración de tienda Shopify

Esta parte del front ha sido desarrollada empleando la biblioteca de JavaScript **React**. Esta es una libería de código abierto creada por el equipo de la compañia Facebook (Meta) junto a una gran comunidad de desarrolladores independientes, que desde su lanzamiento en 2013 se ha convertido en una de las tecnologías de frontend más usadas, ya que permite construir interfaces de usuario dinámicas y escalables. Esta se usa en el desarrollo de aplicaciones web y móviles. [3] Toda aplicación web React, se compone de componentes reutilizables que conforman partes de la interfaz de

usuario, podemos tener un componente distinto para nuestra barra de navegación, otro para el pie de página, otro para el contenido principal, etc. Tener estos componentes reutilizables facilita el desarrollo ya que no tenemos que repetir el código reiterativo, solo tendríamos que crear su lógica e importar el componente en cualquier parte del código donde se necesite. Estos componentes representan una fusión de la estructura **HTML** con la funcionalidad de **JavaScript**. React también conforma una aplicación de una sola página, por tanto, en lugar de enviar una petición al servidor cada vez que hay que renderizar una nueva página, el contenido de la misma se carga directamente desde los componentes de React conduciendo de este modo a una renderización más rápida sin recargas de página. [4]

Para la integración de nuestra aplicación con el panel de administración de las tiendas Shopify se ha hecho uso de **App Bridge**, una biblioteca de JavaScript desarrollada por Shopify que permite a los desarrolladores integrar las aplicaciones personalizadas para las tiendas. Es una interfaz de programcaión diseñada para facilitar la comunicación entre la tienda y las aplicaciones externas que proporciona herramientas y funcionalidades que permiten a los desarrolladores que sus aplicaciones se integren de forma nativa en el panel de administración y en la experiencia de compra del cliente, permitiendo la capacidad de acceder a datos de la tienda como productos, pedidos y clientes así como la capacidad de interactuar con la interfaz de usuario de la tienda agregando componentes y paneles personalizados. [2]

Además se ha empleado el sistema de diseño **Polaris**, una biblioteca de React que define los componentes necesarios para el panel de administración. El panel de control de Shopify proporciona una superficie para que las aplicaciones integradas rendericen su interfaz de usuario, y el empleo de Polaris es fundamental para crear experiencias de usuario familiares y coherentes. Polaris incluye una guía de diseño, con instrucciones sobre accesibilidad, colores, tipografía, espaciado, nomenclatura y lenguaje práctico para ayudarnos a crear experiencias que se parezcan al resto del panel de administración. Nos proporciona componentes, es decir bloques de construcción reutilizables compuestos de elementos y estilos de interfaz empaquetados a través de código. Además nos proporciona tokens, es decir valores CSS con nombre que representan decisiones de diseño para elementos como el color, el espaciado y la tipografía que podemos aplicar a los diseños para ayudar a unificar las experiencias de los usuarios. También nos proporciona iconos cuidadosamente diseñados que se centran en el comercio y el emprendimiento, que podemos emplear como ayudas visuales para ayudar a los usuarios a completar tareas. Y por último nos proporciona patrones, soluciones repetibles a problemas comunes de UX en una situación expecífica del usuario El buen uso de estos patrones hace que el panel de administración sea familiar y facil de usar. [2]

En cuanto a la otra parte del frontend de nuestra aplicación, tenemos la parte relacionada con el tema de la tienda. Un tema en Shopify es un conjunto de recursos y archivos que determinan la apariencia, funcionalidad y sensación de una tienda en línea en esta plataforma, tanto para los comerciantes como para sus clientes. Estos temas se construyen empleando un lenguaje de plantillas llamado **Liquid** que junto con **HTML**, **CSS** y **JavaScript** conforman la apariencia y funcionalidades del frontend de la tienda online. Los desarrolladores pueden crear temas personalidados para las tiendas y ajustar los temas existentes para adaptarse a las necesidades de los comerciantes. [2]

Los administradores tienen la posibilidad de elegir entre varios temas para aplicar a su tienda desde la propia interfaz de administración, ya sean los temas que proporciona Shopify o temas desarrollados y personalizados por otros usuarios como es el caso de Upango. Estos temas pueden ser configurados desde la interfaz de administración por los comerciantes. En la Figura 2 podemos observar la pantalla de la interfaz de administración en la que aparece el tema seleccionado, además se puede observar una biblioteca de temas con otros temas sin publicar. También podemos destacar el botón de personalizar que aparece sobre el tema que está activo, este nos abriría un editor que nos permitiría ajustar ciertos aspectos visuales y detalles funcionales del tema establecido.

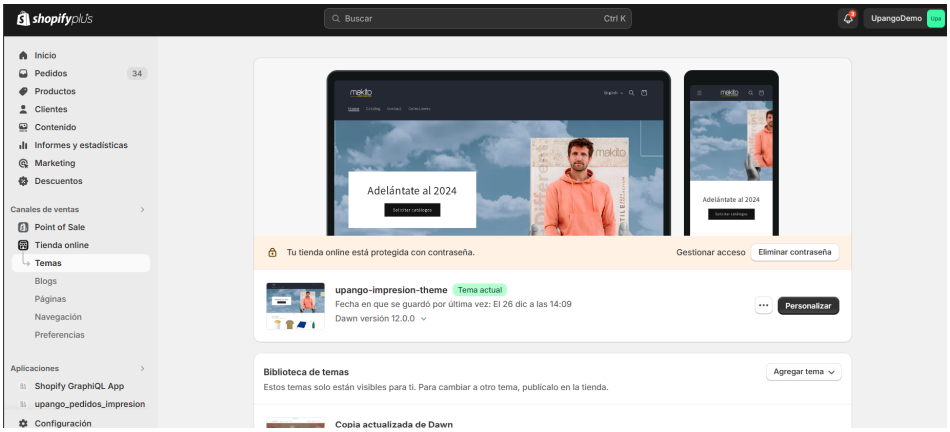

Figura 2: Selección de tema en la interfaz de administración

Para ampliar el tema de las tiendas en las que se instale la aplicación que hemos desarrollado y así poder dotarla de las funcionalidades de personalización de artículos, hemos creado en la aplicación lo que se conoce como **App Theme Extension**. Las extensiones de aplicaciones de temas permiten a los comerciantes agregar fácilmente elementos dinámicos a sus temas sin tener que interactuar directamente con las plantillas o el código **Liquid** del tema. Estas extensiones se integran con el tema de la tienda exponiendo automáticamente en el editor de temas las secciones y elementos que haya programados en la aplicación. [2] Una de las ventajas de desarrollar estas partes del tema en la aplicación y no directamente en el tema de la tienda, es que las secciones de código y funcionalidades que programemos en la aplicación pasarán a estar disponibles en el tema de la tienda en la que se instale directamente sin necesidad de tener que ir modificando el tema de cada tienda en la que queramos instalar nuestra aplicación, es decir, puede implementarse la aplicación al mismo tiempo en todas las tiendas que la usan, sin necesidad de que los comerciantes tengan que editar manualmente el código del tema.

Los **App Theme Extension** tienen una estructura similar a los temas, ya que son como una prolongación de ellos. Estos contienen los siguientes recursos. Por un lado contienen los Bloques, que son archivos en Liquid que actúan como punto de entrada para lo que se desea inyectar en un tema. Por otro lado tienen los recursos, que contienen los ficheros CSS, JavaScript y otros contenidos estáticos de la aplicación que se insertan en los temas. Además tienen los snippets o fragmentos, que son fragmentos

de código Liquid reutilizables que se pueden emplear en varios bloques. Y por último tienen una estructura de archivos locales para implementar un sistema de traducciones en los idiomas que se configuren. [2]

Como se ha comentado anteriormente, para el desarrollo de estas interfaces del tema se emplea **Liquid**, un lenguaje de plantillas creado por Shopify que permite crear una sola plantilla para alojar el contenido estático e insertar información dinámicamente en función de dónde se representa la plantilla. Por ejemplo, podemos crear plantillas de producto que alojen todos los atributos estándar del mismo, como la imagen, el título, el precio y demás, y esta plantilla puede representar dinámicamente esos atributos con el contenido adecuado en función del valor actual que se esté viendo. [2] Además en las plantillas junto con Liquid hemos empleado **HTML**, **CSS** y **JavaScript** para completar el desarrollo de la extensión del tema. Hacemos uso de HTML para estructurar el contenido y crear elementos de las páginas, CSS para definir y dar estilos visuales a los elementos y JavaScript para agregar la interactividad y la funcionalidad dinámica a la página.

En el desarrollo frontend de la parte del tema, se ha hecho uso de una de las herramientas de las que nos proporciona Shopify, la **Shopify Ajax API**. La API de Ajax proporciona un conjunto de puntos finales ligeros para el desarrollo de temas, esta nos permite agregar productos al carrito y actualizar el contador de artículos del mismo, mostrar recomendaciones de productos relaccionados, sugerir productos y colecciones a los visitantes a medida que escriben en un campo de búsqueda y una larga lista de funcionalidades que nos serán útiles para interactuar con la tienda.[2]

4.2.2. Backend

En cuanto al Backend de nuestra aplicación vamos a usar **Node.js**. Node es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas del lado del servidor y aplicaciones en **JavaScript**. Este nos proporciona un gran numero de ventajas. Ofrece un gran rendimiento, pues ha sido diseñado específicamente para optimizar el rendimiento y la escalabilidad en aplicaciones web, lo que lo convierte en un excelente complemento para resolver problemas comunes de desarrollo web como aplicaciones en tiempo real. Todo el código se escribe en JavaScript, lo que nos simplifica trabajo y ahorra tiempo al no tener que preocuparse por las conmutaciones de contexto entre diferentes lenguajes al escribir tanto el código del navegador como del servidor. Además JavaScript es un lenguaje relativamente nuevo y se beneficia de los avances en diseño de lenguajes en comparación con otros lenguajes tradicionales. **Node** cuenta con **NPM** (Node Packet Manager), un gestor de paquetes que nos proporciona acceso a una amplia gama de paquetes reutilizables, ofrece una excelente resolución de dependencias y nos permite automatizar gran parte de la cadena de herramientas de compilación. También Node es portable y compatible con multitud de sistemas operativos y está soportado por muchos proveedores de alojamiento web, contando con un gran ecosistema y una comunidad de desarrolladores muy activa. Por último este entorno nos brinda muchas facilidades para crear servidores web básicos capaces de responder a cualquier solicitud empleando el paquete HTTP de Node y simplificando el proceso de desarrollo y

despliegue de aplicaciones Web.[5]

Además hemos empleado **Express**, el framework web más popular de Node, este simplifica enormemente el desarrollo de aplicaciones web al proporcionar una serie de características esenciales. Con Express, los desarrolladores pueden escribir fácilmente manejadores de peticiones para diferentes verbos HTTP en diversas rutas URL, integrar motores de renderización de vistas para generar respuestas dinámicas y establecer configuraciones de aplicaciones web de manera intuitiva. Además, permite la adición de procesamiento de middleware adicional en cualquier punto de la tubería de manejo de la petición, brindando flexibilidad y modularidad al desarrollo. [5]

En esta parte de la aplicación, hemos hecho uso de la API de administración **GraphQL** que nos proporciona Shopify. Esta consituye la principal forma en la que las aplicaciones pueden interactuar con la tienda, permitiendonos leer y escribir la información de la tienda incluidos los productos, inventario, pedidos, envíos y mucho más, permitiendonos ampliar la funcionalidad existente de Shopify. Gracias a esta API podemos conectar el inventario de la tienda con otros marketplaces y nos ofrece infinidad de posibilidades, pudiendo añadir nuevas funcionalidades en el panel de control. Esta API es compatible tanto con GraphQL como con REST aunque para nuestro desarrollo emplearemos GraphQL. Para interactuar con esta API las aplicaciones deben autenticarse además de contar con los ámbitos de acceso pertinentes para acceder a los datos que se necesiten o se quieran modificar. [2]

4.2.3. Otras tecnologías

Para la elaboración de este TFG hemos empleado **Visual Studio Code** como entorno de desarrollo tanto para la creación de la aplicación como para la redación del propio TFG en **LaTeX**, empleando una serie de extensiones que nos han facilitado el trabajo. Hemos aprovechado extensiones como Shopify Liquid Templates, Shopify Liquid, Polaris for VS Code, JavaScript (ES6), GraphQL, Latex Language support, Latex Workshop y ESLint para garantizar un desarrollo y documentación fluida y eficaz. Estas extensiones han proporcionado una serie de funcionalidades como resaltado de sintaxis, autocompletado y validación de código, acceso rápido a la documentación, formateo de código y una serie de ventajas que nos han facilitado significativamente la creación y documentación de la aplicación.

Hemos empleado **Git** para llevar un control de versiones de la aplicación junto con **Azure DevOps**, una plataforma de Microsoft que nos ofrece una amplia gama de características y funcionalidades que nos permiten a los equipos de desarrolo colaborar de manera efectiva en equipo y optimizar el proceso de desarrollo de software. Esta herramienta nos ofrece gestión del código fuente, permitiendonos almacenar y administrar el código fuente de nuestro proyecto en repositorios de código. También nos facilita la planificación, asignación y seguimiento de las tareas, problemas y requisitos del proyecto, haciendo un seguimiento del trabajo, y muchas otras fucionalidades útiles para el desarrollo como seguimiento, pruebas y gestión del ciclo de vida de los proyectos. [6]

5. Análisis de objetivos y metodología

En esta sección se describen los objetivos que se persiguen con el desarrollo de este proyecto, así como la metodología empleada (tareas y temporalización) para alcanzar estos objetivos.

5.1. Objetivos

El objetivo principal de este TFG consiste en el análisis, diseño y desarrollo de una aplicación para la plataforma Shopify que nos permita la personalización de productos para su posterior compra en las tiendas en las que se acople dicha aplicación. Para conseguir lograr este objetivo principal, se han propuesto los siguientes objetivos específicos.

1. Realizar el análisis y estudio de las tecnologías y herramientas necesarias para el desarrollo de esta aplicación
2. Analizar las funcionalidades y requisitos necesarios que debe tener la aplicación.
3. Habiendo hecho el estudio de tecnologías y análisis de requisitos, realizar una división de tareas para obtener una serie de pasos a seguir para lograr el desarrollo. (Diseño)
4. Implementar la aplicación siguiendo la división de tareas establecida.
5. Desplegar la aplicación en una tienda de pruebas para comprobar su funcionamiento, realizar pruebas manuales y compras ficticias.
6. Documentar todo este procedimiento para plasmarlo en el TFG

5.2. Metodología

Etapas en las que se ha divido la realización del TFG, con indicación de fechas de inicio/fin.

5.2.1. Temporalización

En este apartado vamos a enumerar las etapas en las que se ha dividido el desarrollo de este proyecto.

- Etapa 1.
- Etapa 2.
- Etapa 3.

6. Diseño y resolución del trabajo realizado

Parte más extensa del trabajo donde se describe lo que se ha hecho paso por paso y el resultado final.

6.1. Análisis

Requisitos textuales o la definición de casos de uso o las historias de usuario

6.1.1. Historias de usuario

Historia de Usuario 1: Botón de Personalizar en la Ficha de Producto

Como administrador de la tienda, quiero poder observar en las páginas de producto un botón que posteriormente servirá para activar una funcionalidad de personalización del producto.

Criterios de Aceptación:

1. Cuando visualizo la ficha de un producto en la tienda,
   - Debe haber un botón claramente identificable como "Personalizar".
2. El botón "Personalizar"solo debe mostrarse si el producto es de impresión,
   - Un producto se considerará de impresión si tiene información específica en su metafield denominado *upng.areas_impresion*.
3. El título del botón debe ser recuperado de los archivos de locales,
   - Se deben preparar traducciones en inglés y español para el texto del botón y cualquier otro texto relevante en la aplicación.

Historia de Usuario 2: Configuración de Opciones de Envío Internacional

Como administrador de la tienda, quiero poder configurar opciones de envío internacional para poder ofrecer servicios de envío a clientes de todo el mundo y gestionar eficazmente los envíos internacionales.

Criterios de Aceptación:

1. Debe existir una sección en el panel de administración para configurar opciones de envío internacional.
   - Los administradores deben poder acceder fácilmente a esta sección desde el panel de control de la tienda.
2. Se deben ofrecer diferentes métodos de envío internacional,

   - Los métodos de envío deben incluir opciones como correo prioritario, envío estándar, envío exprés, entre otros.

3. Los administradores deben poder establecer tarifas de envío específicas para cada región internacional,

   - Se debe proporcionar un formulario donde los administradores puedan ingresar tarifas de envío para diferentes regiones del mundo.

Historia de Usuario 3: Configurar Impresión al Añadir al Pedido

Como administrador de la tienda, deseo poder configurar la impresión de productos antes de añadirlos al pedido, para poder personalizarlos según las necesidades de los clientes.

Criterios de Aceptación:

1. Al pulsar el botón de .ªñadir al Pedido de Impresión", se debe abrir un modal que permita al usuario configurar la impresión del producto.

2. El modal debe mostrar toda la información relevante sobre la impresión y permitir al usuario ajustar la configuración según sea necesario.

3. Se debe crear una estructura JavaScript utilizando Liquid para tener preparados todos los datos de áreas, trabajos, clichés, etc., de manera que no sea necesario realizar llamadas adicionales al Admin API para obtener la información.

4. En el modal, se debe mostrar un resumen claro y detallado de los elementos seleccionados por el usuario, incluyendo el nombre del producto, la cantidad, las áreas seleccionadas, los trabajos en cada área, los clichés, los precios individuales y el importe total.

5. El usuario debe poder ingresar la cantidad de productos a imprimir, empleando el selector de cantidad por defecto.

6. Se deben mostrar las áreas de impresión que tenga el producto y de cada área de impresión debe mostrarse una imagen, su nombre y medidas ajustables (ancho y largo) en inputs. Estos inputs deben validar que el valor máximo sea el de la medida del área y el mínimo sea cero.

7. Se debe mostrar un checkbox junto a cada área para que el usuario pueda seleccionar en cuáles desea imprimir.

8. Las imágenes de las áreas que no tengan una definida deben mostrar una imagen por defecto, configurable a través del Theme App Extension.

9. Cada área debe mostrar los trabajos disponibles en un desplegable para que el usuario pueda seleccionar el que desee.

10. Cada trabajo tiene un número máximo de colores, por lo que al lado del desplegable de trabajos debe aparecer un desplegable de colores para que el usuario elija el número de colores que necesita. Y cada vez que se actualice el trabajo seleccionado deberá refrescarse este desplegable.

11. Los selectores de trabajo y colores de las áreas que el usuario no haya seleccionado deberán aparecer como desactivadas.

12. Se debe habilitar un campo de observaciones en cada área donde el usuario pueda agregar las notas pertinentes.

13. Para cada trabajo, se debe permitir al usuario seleccionar el número de colores y especificar los colores deseados. Para ello según la selección del usuario en el desplegable de colores se deberán añadir tantos selectores de colores como colores se hayan seleccionado en el mismo. Además del selector de colores se añadirán campos de texto libre para que el usuario pueda especificar cada color mediante esta forma.

14. Se debe incluir la opción de marcar si un cliché es de repetición, afectando al precio final.

15. Se debe actualizar dinámicamente la información del resumen al seleccionar o deseleccionar áreas, trabajos y colores.

16. El diseño del modal debe ser responsive para una experiencia óptima en dispositivos móviles.

17. Los trabajos deben tener precios diferentes para el color principal y el resto de colores, con posibilidad de seleccionar variantes según sea necesario.

Historia de Usuario 4: Mejora en la Funcionalidad de Añadir al Carrito

Como administrador de la tienda, quiero que al añadir productos relacionados (productos normales, trabajos y clichés) al carrito, estos estén agrupados y vinculados de manera que no se puedan borrar o editar por separado, para mejorar la experiencia de compra y facilitar la gestión de los productos de impresión.

Criterios de Aceptación:

1. Se debe investigar si el uso de customized bundles es adecuado para agrupar y vincular los productos relacionados en el carrito.

2. Estos productos tienen que estar agrupados y de forma que no se puedan eliminar o editar por separado.

3. Toda la información de impresión configurada (medidas, colores, imagen adjunta y observaciones de cada área) debe guardarse al añadir al carrito, para que pueda transferirse al pedido durante el proceso de checkout.

4. Al añadir productos al carrito, se deben incluir:
    - El producto "normal".
    - Los productos "trabajos"seleccionados en diferentes áreas, con la variante adecuada según el número de colores seleccionados.
    - Los productos çlichés.ªsociados al trabajo, con la variante adecuada según si es de repetición o no.
5. La cantidad de cada producto añadido al carrito debe ser considerada y reflejada correctamente.

Historia de Usuario 5: Visualización Agrupada de Productos de Impresión en el Carrito

Como comerciante en la plataforma, quiero que los productos de impresión, junto con sus trabajos y clichés asociados, se visualicen de forma agrupada en el carrito de la tienda, para proporcionar una experiencia de compra clara y comprensible para mis clientes.

Criterios de Aceptación:

1. Los productos de impresión deben aparecer agrupados en el carrito, mostrando claramente sus trabajos y clichés asociados.
2. El detalle de estos paquetes de productos de impresión en el carrito debe ser similar a la visualización que se presenta durante el proceso de checkout.
3. La visualización agrupada en el carrito debe ser coherente con el diseño y la funcionalidad general de la tienda.

Historia de Usuario 6: Personalización Dinámica de Precios para Trabajos Específicos

Como administrador de la tienda, quiero que algunos trabajos puedan cambiar su precio de forma dinámica en función de ciertas condiciones, para ofrecer precios precisos y competitivos a mis clientes según las características específicas de cada trabajo.

Criterios de Aceptación:

1. Se debe poder personalizar dinámicamente el precio de ciertos trabajos en función de condiciones específicas.
2. Los precios personalizados deben ser precisos y reflejar adecuadamente cualquier cambio en las condiciones que afecten al precio del trabajo.

Historia de Usuario 7: Visualización Atractiva del Logo del Cliente en el Producto

Como administrador de la tienda, quiero poder mostrar al cliente una representación visual atractiva de cómo quedaría su logo en el producto, para ofrecer una experiencia de compra más personalizada y satisfactoria.

Criterios de Aceptación:

1. Se debe poder mostrar una imagen del logo del cliente en el producto de forma atractiva y realista.
2. La visualización del logo debe reflejar fielmente su posición y tamaño en relación con las áreas designadas del producto.
3. La implementación de la visualización del logo debe ser viable y práctica dentro del contexto de la tienda en línea.

Historia de Usuario 8: Automatización de la Creación de Metafields Necesarios

Como administrador de la tienda, quiero una opción para crear automáticamente los metafields necesarios para la app, para simplificar y agilizar el proceso de configuración y evitar posibles errores manuales.

Criterios de Aceptación:

1. Se debe proporcionar una opción en el panel de configuración de la app para crear automáticamente los metafields necesarios.
2. La creación de los metafields debe realizarse de manera controlada y asegurarse de no duplicar campos ya existentes.
3. Después de ejecutar la creación de metafields, se debe mostrar un mensaje de feedback al usuario indicando el resultado de la operación.
4. La configuración de qué metafields se crearán debe ser configurable para facilitar la expansión y reutilización de la funcionalidad en otras aplicaciones.

Historia de Usuario 9: Gestión de Casuística "Doble Pasada.^{en} Productos y Trabajos

Como administrador de la tienda, quiero poder gestionar la casuística de "doble pasada.^{en} productos y trabajos, para aplicar recargos adicionales en caso de que tanto el producto como el trabajo seleccionados tengan la opción de "doble pasada.^{a}ctivada.

Criterios de Aceptación:

1. Debe existir un nuevo metafield booleano llamado "DOBLE PASADA"para los productos y los trabajos.
2. Si se selecciona un producto con la opción "DOBLE PASADA.ªctivada y se elige un trabajo que también tenga esta opción activada, se debe mostrar en el presupuestador un check para permitir al usuario seleccionar la opción de "doble pasada".
3. Si se cambia el trabajo seleccionado a uno que no tenga la opción de "doble pasada", el check debe ocultarse automáticamente.
4. Este recargo adicional debe mostrarse en el resumen del pedido del presupuestador.

Historia de Usuario 10: Gestión del Canon Digital en Productos

Como administrador de la tienda, quiero poder gestionar el canon digital en ciertos productos, para aplicar un importe fijo adicional al precio de algunos productos específicos.

Criterios de Aceptación:

1. Debe existir un metafield llamado Ïmporte Canon"para los productos, que permita indicar si un producto tiene canon digital y especificar el importe correspondiente.
2. Se debe crear un producto especial con un precio de un céntimo, que se utilizará para añadir el importe del canon digital al carrito de compra.

Historia de Usuario 11: Adición Automática de Línea de Canon al Añadir un Producto al Carrito

Como administrador de la tienda, quiero que al añadir un producto al carrito, se agregue automáticamente una línea de canon digital correspondiente al producto seleccionado, para reflejar correctamente el importe del canon asociado a los productos en mi carrito de compra.

Criterios de Aceptación:

1. Cuando se añada un producto al carrito, se debe agregar automáticamente una línea de canon digital asociada al producto.
2. La línea de canon digital debe estar vinculada al producto correspondiente y reflejar el importe del canon establecido para ese producto.
3. Se debe utilizar bundles para asociar cada línea de producto con su línea de canon digital correspondiente en el carrito.

6.2. Diseño

6.2.1. Tareas

Tareas relacionadas con la Historia de usuario 1:

- **Tarea 1:** Crear una Aplicación y una Extensión de Tema:
  - Descripción: Crear una aplicación e instalarla en la tienda y crear un theme app extensión para poder añadir el botón a las plantillas de ficha de producto a través del personalizador del theme.
- **Tarea 2:** Crear el botón y mostrarlo si el Producto es de Impresión:
  - Descripción: Utilizar liquid para verificar si el producto es de impresión. Un producto se considerará de impresión si tiene información específica en su metafield denominado *upng.areas_impresion*.
  - Mostrar el botón "Personalizar"si el producto cumple con los criterios establecidos; de lo contrario, ocultar el botón.
- **Tarea 3:** Recuperar el Texto del Botón de los Archivos de Locales:
  - Descripción: Configurar la aplicación para que el texto del botón y cualquier otro texto relacionado se recupere de los archivos de locales. Preparar traducciones en inglés y español para garantizar una experiencia multilingüe completa.

Tareas relacionadas con la Historia de usuario 2:

- **Tarea 1:** Crear Sección de Configuración de Envío Internacional:
  - Descripción: Implementar una sección en el panel de administración de la tienda para que los administradores puedan configurar opciones de envío internacional de manera intuitiva.
- **Tarea 2:** Definir Métodos de Envío Internacional:
  - Descripción: Establecer diferentes métodos de envío internacional disponibles para los clientes, incluyendo opciones de envío prioritario, estándar y exprés, entre otros.
- **Tarea 3:** Establecer Tarifas de Envío por Región:
  - Descripción: Crear un formulario en la sección de configuración de envío internacional donde los administradores puedan ingresar tarifas de envío específicas para cada región del mundo.

Tareas relacionadas con la Historia de usuario 3:

- **Tarea 1:** Abrir un modal:

  - Descripción: Al pulsar el botón .Añadir al Pedido de Impresión", se debe abrir un modal que contenga toda la información necesaria para configurar la impresión del producto.

- **Tarea 2:** Crear estructura JavaScript para datos de impresión:

  - Descripción: Utilizar Liquid para crear una estructura JavaScript que contenga todos los datos relevantes de áreas, trabajos, clichés, etc., necesarios para la configuración de la impresión. Esto permitirá tener los datos disponibles en el modal sin necesidad de realizar llamadas adicionales al Admin API.

- **Tarea 3:** Mostrar datos del resumen:

  - Descripción: En el modal, mostrar una sección de resumen que presente de manera clara y detallada todos los elementos seleccionados por el usuario, incluyendo el nombre del producto, cantidad, áreas seleccionadas con trabajos y clichés, precios individuales y el importe total. Esta sección debe actualizarse dinámicamente al configurar diferentes elementos de la impresión.

- **Tarea 4:** Ajustar la cantidad de productos:

  - Descripción: Permitir al usuario ingresar la cantidad de productos que desea imprimir. Esta cantidad se multiplicará por el precio de la impresión para calcular el importe total.

- **Tarea 5:** Mostrar áreas de impresión:

  - Descripción: Mostrar todas las áreas de impresión disponibles para el producto, incluyendo una imagen, nombre y medidas ajustables (ancho y largo). Habilitar un checkbox para que el usuario pueda seleccionar las áreas en las que desea imprimir.

- **Tarea 6:** Imagen por defecto para áreas sin imagen:

  - Descripción: Configurar una imagen por defecto para las áreas que no tengan una imagen definida. Permitir al usuario personalizar esta imagen a través del Theme App Extension.

- **Tarea 7:** Mostrar trabajos para cada área:

  - Descripción: Para cada área de impresión seleccionada, mostrar los trabajos disponibles en un desplegable. Permitir al usuario seleccionar el trabajo deseado y especificar el número de colores.

- **Tarea 8:** Incluir campo de observaciones:

  - Descripción: En cada área de impresión, incluir un campo de observaciones donde el usuario pueda agregar notas pertinentes relacionadas con la impresión.

- **Tarea 9:** Seleccionar colores:
  - Descripción: Permitir al usuario seleccionar el número de colores para cada trabajo en un desplegable. Además, proporcionar campos de texto para que el usuario especifique los colores. Estos campos estarán habilitados solo si el cliente selecciona el área correspondiente.
- **Tarea 10:** Marcar cliché de repetición:
  - Descripción: Incluir un checkbox para que el usuario pueda indicar si el cliché es de repetición o no. Este factor afectará al precio final de la impresión.
- **Tarea 11:** Seleccionar áreas:
  - Descripción: Habilitar y deshabilitar los campos de trabajos, colores, medidas, etc., según las áreas seleccionadas por el usuario. Actualizar dinámicamente la información del resumen con los precios correspondientes.
- **Tarea 12:** Seleccionar un trabajo:
  - Descripción: Actualizar la información del resumen al seleccionar un trabajo para un área específica. Además, actualizar las opciones del selector de colores según el trabajo seleccionado.
- **Tarea 13:** Seleccionar número de colores:
  - Descripción: Al escoger el número de colores en el desplegable, proporcionar campos de texto para que el usuario especifique los colores. Actualizar la cantidad de clichés y multiplicar su precio en el resumen.
- **Tarea 14:** Adjuntar imagen por área:
  - Descripción: Permitir al usuario adjuntar una imagen para cada área de impresión. Crear un método API propio en la aplicación para gestionar la subida de imágenes al servidor. Generar un ID aleatorio asociado al producto de impresión y guardar la imagen junto con los demás datos configurados.
- **Tarea 15:** Diseño responsive:
  - Descripción: Asegurar que el diseño del modal y la forma de mostrar los elementos sea óptima y clara en la vista móvil.
- **Tarea 16:** Precios diferenciados para trabajos según colores:
  - Descripción: Implementar precios diferenciados para los trabajos en función del color principal y el resto de colores seleccionados por el usuario. Utilizar variantes para gestionar los precios según sea necesario.

Tareas relacionadas con la Historia de usuario 4:

1. **Tarea 1:** Investigar el Uso de Customized Bundles para Agrupar y Vincular Productos:
   - Descripción: Investigar si los customized bundles son una solución adecuada para agrupar y vincular los productos relacionados en el carrito de manera que no se puedan borrar o editar por separado.
   - Considerar visualmente cómo se mostrarían en el carrito, el checkout y en el historial de pedidos.
   - Si no se considera viable el uso de customized bundles, explorar la posibilidad de utilizar un Theme App Extension de JavaScript para agregar esta funcionalidad al carrito, especialmente enfocado en impedir cambios individuales en los productos relacionados.
2. **Tarea 2:** Guardar Información de Impresión en el Carrito:
   - Descripción: Desarrollar la funcionalidad para guardar toda la información de impresión configurada (medidas, colores, imagen adjunta y observaciones de cada área) en el carrito al añadir productos relacionados.
   - Utilizar las properties de los LineItems para almacenar esta información de manera adecuada.
3. **Tarea 3:** Añadir Productos Relacionados al Carrito:
   - Descripción: Implementar la lógica para añadir al carrito los productos relacionados, incluyendo el producto "normal", los trabajos seleccionados y los clichés asociados, teniendo en cuenta la cantidad y las variantes adecuadas según las selecciones del usuario.
   - Utilizar el ajax API para añadir los productos al carrito.
4. **Tarea 4:** Actualizar la Forma de Añadir al Carrito según Conclusiones sobre Customized Bundles (Tarea Pendiente):
   - Descripción: Dependiendo de las conclusiones de la investigación sobre customized bundles, ajustar la forma de añadir productos al carrito según sea necesario.

Tareas relacionadas con la Historia de usuario 5:

- **Tarea 1:** Modificar el Theme para Mostrar el Detalle de los Bundles de Impresión en el Carrito:
  - Descripción: Desarrollar una sección en el Theme que sea una versión de la sección `main-cart-items.liquid` (sección en la que se muestran los productos del carrito) en la que se muestre el detalle de los bundles de impresión de manera similar a la visualización en el proceso de checkout.

Tareas relacionadas con la Historia de usuario 6:

- **Tarea 1:** Investigar el Uso de `cart-transform` para Personalizar Precios de Líneas:
  - Descripción: Realizar una investigación exhaustiva sobre el uso de `cart-transform` para determinar cómo personalizar los precios de las líneas de productos de forma dinámica.

Tareas relacionadas con la Historia de usuario 7:

- **Tarea 1:** Investigar Opciones para Mostrar el Logo del Cliente en la Zona Adecuada del Producto e implementarlo:
  - Descripción: Realizar una investigación exhaustiva para identificar opciones, librerías u herramientas que nos permitan mostrar una imagen del logo del cliente en la zona adecuada del producto de forma atractiva.
  - Evaluar la viabilidad de cada opción en función de la información disponible sobre el producto de impresión y las necesidades específicas del cliente.
  - Implementar la solución elegida.

Tareas relacionadas con la Historia de usuario 8:

- **Tarea 1:** Limpiar la Interfaz del Administrador de la App:
  - Descripción: Eliminar las opciones demo y botones relacionados en la parte frontal de la interfaz de administración de la app para eliminar funcionalidad innecesaria.
- **Tarea 2:** Agregar un Botón Çrear Campos Necesarios.[en] la Interfaz de Administración:
  - Descripción: Incorporar un botón en la interfaz de administración de la app que llame a un método de la app para crear los metafields necesarios.
- **Tarea 3:** Desarrollar el Método para Crear los Metafields Necesarios:
  - Descripción: Crear un método en la app que utilice llamadas al API de GraphQL para crear los metafields y los metaobject necesarios, controlando si ya existen.
- **Tarea 4:** Implementar la Funcionalidad de Feedback para el Usuario:
  - Descripción: Desarrollar una funcionalidad para mostrar un mensaje de feedback al usuario después de la ejecución del método de creación de metafields, informando sobre el éxito o posibles errores.
- **Tarea 5:** Hacer Configurable la Creación de Metafields:

  - Descripción: Crear un fichero de configuración o constantes que defina qué metafields se crearán, para qué entidad y de qué tipo. Adaptar el código para usar esta configuración, permitiendo así una fácil expansión y reutilización de la funcionalidad en otras aplicaciones.

- **Tarea 6:** Crear el objeto `cartTransform` Automáticamente:

  - Descripción: Implementar la funcionalidad para crear automáticamente el objeto `cartTransform` si es necesario. Esto simplificaría aún más el proceso de configuración para el administrador de la tienda, permitiendo una mayor automatización y reduciendo la carga de trabajo manual.

Tareas relacionadas con la Historia de usuario 9:

- **Tarea 1:** Crear Metafields para "Doble Pasada."[en] Productos:

  - Descripción: Utilizar la funcionalidad de creación automática de metafields para crear un metafield booleano llamado "DOBLE PASADA"para los productos.

- **Tarea 2:** Mostrar el Check para "Doble Pasada":

  - Descripción: Implementar la lógica para mostrar un check junto al trabajo seleccionado en el presupuestador si tanto el producto como el trabajo tienen la opción de "doble pasada."[a]ctivada. Asegurarse de que el check se oculte automáticamente si se cambia el trabajo seleccionado a uno que no tenga la opción de "doble pasada".

Tareas relacionadas con la Historia de usuario 10:

- **Tarea 1:** Crear Metafield para el Importe del Canon Digital:

  - Descripción: Crear un metafield de tipo número decimal, llamado Ïmporte Canon", utilizando la funcionalidad de creación automática de metafields. Este metafield se usará para indicar si un producto tiene canon digital y especificar el importe correspondiente.

- **Tarea 2:** Crear Producto Especial para Añadir el Importe al Carrito:

  - Descripción: Crear un producto especial con un precio de un céntimo, que se utilizará únicamente para añadir el importe del canon digital al carrito de compra. Este producto especial asegurará que el importe del canon se aplique correctamente al total del carrito de compra.

Tareas relacionadas con la Historia de usuario 11:

- **Tarea 1386:** Añadir Líneas de Canon al Carrito y Asociarlas a sus Productos Correspondientes con Bundles:

- Descripción: Desarrollar la lógica para añadir automáticamente líneas de canon digital al carrito cuando se añada un producto con canon. Utilizar bundles para asociar cada línea de producto con su línea de canon correspondiente.
- Iniciar con la implementación para productos "normalesz luego extender la funcionalidad para productos de impresión.
- Controlar la adición de líneas de canon desde la ficha de producto inicialmente, y luego expandir la funcionalidad para otros puntos de entrada al carrito.
- Considerar la utilización del cartTransform existente para integrar esta nueva lógica de bundles, y evaluar la posibilidad de separarlo en una aplicación independiente en el futuro.

6.2.2. Diseño de back-end

Describir el API REST

6.2.3. Diseño de front-end

Mock-ups Diagrama de navegación

6.3. Implementación

6.3.1. Implementación de front-end

6.3.2. Implementación de back-end

6.4. Pruebas

6.5. Despliegue

7. Conclusiones y vías futuras

Puede estar subdividido en dos apartados: (1) conclusiones, (2) trabajo futuro. El trabajo futuro se refiere a carencias actuales del trabajo que están sujetas a mejoras en el futuro, posibles extensiones para dotar de mayor funcionalidad, etc.

8. Bibliografía

[1] Shopify. *Shopify*, dirección: https://www.shopify.com/es/blog/tutorial-shopify (visitado 29-03-2024).

[2] Shopify Developers. *Shopify.dev*, dirección: https://shopify.dev/ (visitado 29-03-2024).

[3] React *React*, dirección: https://ebac.mx/blog/que-es-react (visitado 29-03-2024).

[4] Funcionamiento React *Funcionamiento React*, dirección: https://kinsta.com/es/base-de-conocimiento/que-es-react-js/#qu-es-react (visitado 29-03-2024).

[5] Node y Express *Node y express*, dirección: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction (visitado 31-03-2024).

[6] Azure DevOps *Azure DevOps*, dirección: https://aodatacloud.es/blog/que-es-azure-devops-y-para-que-sirve/#%C2%BFQue_es_Azure_DevOps (visitado 31-03-2024).

9. Anexos

9.1. Manual de usuario

En este apartado se mostrará de una forma visual la funcionalidad de la aplicación y su funcionamiento dentro de una tienda de desarrollo.