



UNIVERSIDAD DE MURCIA  
FACULTAD DE INFORMÁTICA

---

---

# App para pedidos de impresión

## TRABAJO DE FIN DE GRADO

*RUBÉN SÁNCHEZ FERNÁNDEZ*  
*21067018F*

**Tutor académico**  
*FRANCISCO GARCÍA SÁNCHEZ*  
**Tutor empresa**  
*CARMELA POZUELO MONFORT*

*CONVOCATORIA DE JUNIO 2024*

# Índice general

<b>1</b>	<b>Resumen</b>	<b>III</b>
<b>2</b>	<b>Extended abstract</b>	<b>IV</b>
<b>3</b>	<b>Introducción</b>	<b>1</b>
<b>4</b>	<b>Estado del arte</b>	<b>2</b>
4.1	Contexto y análisis de la situación de partida . . . . .	2
4.2	Herramientas y tecnologías empleadas . . . . .	2
4.2.1	Frontend . . . . .	3
4.2.2	Backend . . . . .	6
4.2.3	Otras tecnologías . . . . .	8
<b>5</b>	<b>Análisis de objetivos y metodología</b>	<b>9</b>
5.1	Objetivos . . . . .	9
5.2	Metodología . . . . .	9
5.2.1	Temporalización . . . . .	9
5.2.2	Proceso de desarrollo . . . . .	11
<b>6</b>	<b>Diseño y resolución del trabajo realizado</b>	<b>12</b>
6.1	Análisis . . . . .	12
6.1.1	Historias de usuario . . . . .	13
6.2	Diseño . . . . .	18
6.2.1	Tareas . . . . .	18
6.2.2	Diseño de back-end . . . . .	24
6.2.3	Diseño de front-end . . . . .	25
6.3	Implementación . . . . .	25
6.3.1	Implementación de front-end . . . . .	25
6.3.2	Implementación de back-end . . . . .	25
6.4	Pruebas . . . . .	25
6.5	Despliegue . . . . .	26
<b>7</b>	<b>Conclusiones y vías futuras</b>	<b>30</b>
7.1	Conclusiones . . . . .	30
7.2	Trabajo futuro . . . . .	30
<b>8</b>	<b>Bibliografía</b>	<b>31</b>
<b>9</b>	<b>Anexos</b>	<b>32</b>
9.1	Manual de usuario . . . . .	32

## Índice de figuras

1	Interfaz de administración de tienda Shopify . . . . .	4
2	Selección de tema en la interfaz de administración . . . . .	5
3	Backlogs del proyecto en Azure DevOps . . . . .	12
4	Estados y prioridad de una tarea del Backlog . . . . .	12
5	package.json . . . . .	27
6	Variables de entorno de la aplicación . . . . .	28
7	Configuración de la aplicación . . . . .	29
8	Distribución de la aplicación . . . . .	29
9	Generar link de distribución . . . . .	30
10	Personalizar tema en el panel de administración . . . . .	32
11	Añadir la extensión de tema de la App en el tema de la tienda . . . . .	33
12	Panel de administración de la tienda . . . . .	33
13	Funcionalidades de la parte de administración de la App . . . . .	34
14	Página principal de la tienda de demostración . . . . .	35
15	Página de producto no personalizable . . . . .	35
16	Página de producto personalizable . . . . .	36
17	Pantalla del personalizador del producto seleccionado . . . . .	36
18	Personalizador con el Area 1 seleccionada . . . . .	37
19	Personalizador con varias áreas seleccionadas . . . . .	37
20	Personalizador con trabajo de impresión y colores seleccionados . . . . .	38
21	Personalizador con opciones de repetición de cliché y doble pasada . . . . .	38
22	Personalizador con selección de logo . . . . .	38
23	Botón de carrito . . . . .	39
24	Pantalla del carrito de la tienda . . . . .	39
25	Pantalla de checkout de la tienda . . . . .	39

## 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 y 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 (TFG) ha sido desarrollado en colaboración con Upango, empresa especializada en transformaciones digitales B2B. Durante años ha trabajado con uno de los proveedores de soluciones de tiendas online más populares del mundo como es ePages [1], 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[3] 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, el objetivo que se persigue con este proyecto es crear una aplicación para la plataforma Shopify que sea versátil y escalable, y permita integrarse en las tiendas de los nuevos clientes para ofrecerles la capacidad de proporcionar experiencias de compra de productos personalizados, manteniendo el 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 este caso **Shopify**.

Es importante destacar que la elección de parte de las tecnologías ha estado fuertemente influenciada por la decisión de la empresa de emplear Shopify como base para sus desarrollos. 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. 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 simplifica el proceso de lanzamiento y mantenimiento de una tienda al eliminar la necesidad de buscar proveedores de hosting externos, además de dotar a los usuarios 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 también 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. [3]

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. [4]

#### 4.2.1. Frontend

En cuanto al frontend de la aplicación podemos observar dos partes. Por un lado se puede encontrar 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 la aplicación en las tiendas que necesiten de sus funcionalidades esta parte del frontend se integrará con la página de administración de cada tienda, proporcionando la interfaz y las funcionalidades que se configuren en esta parte del desarrollo. En la Figura 1 podemos observar la interfaz de administración de una tienda Shopify en desarrollo.

Esta parte del frontend ha sido desarrollada empleando la biblioteca de JavaScript **React**. Se trata de una librería de código abierto creada por el equipo de la compañía 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



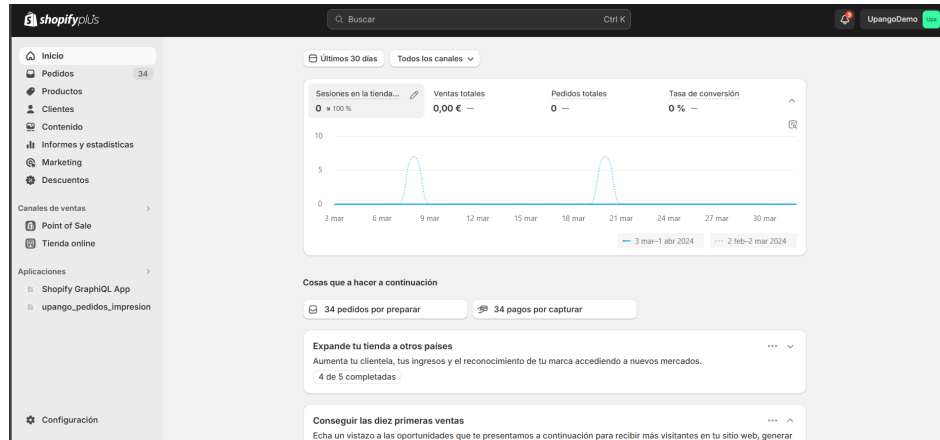


Figura 1: Interfaz de administración de tienda Shopify

más usadas, ya que permite construir interfaces de usuario dinámicas y escalables. Esta se usa en el desarrollo de aplicaciones web y móviles [5].

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 es necesario repetir el código reiterativo, solo habría 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. [6]

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. Se trata de una interfaz de programación 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. De esta forma se proporciona 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. [4]

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 ayudar a crear experiencias que se parezcan al resto del panel de administración.

Proporciona componentes, es decir bloques de construcción reutilizables compuestos de elementos y estilos de interfaz empaquetados a través de código. Además proporciona tokens, es decir valores CSS con nombre que representan decisiones de diseño para elementos como el color, el espaciado y la tipografía se puede aplicar a los diseños para ayudar a unificar las experiencias de los usuarios. También 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 proporciona patrones, soluciones repetibles a problemas comunes de experiencia de usuario (UX, en inglés *Ūser eXperience*) en una situación específica del usuario. El buen uso de estos patrones hace que el panel de administración sea familiar y fácil de usar. [4]

La otra parte del frontend de esta aplicación, es la extensión del 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 personalizados para las tiendas y ajustar los temas existentes para adaptarse a las necesidades de los comerciantes. [4]

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 se muestra 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 se puede 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. [4] 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 la aplicación. Es decir, puede implementarse la aplicación al mismo tiempo en todas las tiendas que la usen, 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. los *bloques*, son archivos en Liquid que actúan como punto de entrada para lo que se desea inyectar en un tema. Recursos, contienen los ficheros CSS, JavaScript y otros contenidos estáticos de la aplicación que se insertan en los temas. Los snippets, son fragmentos de código Liquid reutilizables que se pueden emplear en varios bloques. Por último, una estructura de archivos locales para implementar un sistema de traducciones en los idiomas que se configuren. [4]

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 de este, 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. [4] Además, en las plantillas junto con Liquid se ha empleado **HTML**, **CSS** y **JavaScript** para completar el desarrollo de la extensión del tema. Se hace 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 que proporciona Shopify, la **Shopify Ajax API**. La API de Ajax proporciona un conjunto de puntos finales ligeros para el desarrollo de temas: permite agregar productos al carrito y actualizar el contador de artículos del mismo, mostrar recomendaciones de productos relacionados, sugerir productos y colecciones a los visitantes a medida que escriben en un campo de búsqueda y una larga lista de funcionalidades que serán útiles para interactuar con la tienda.[4]

#### 4.2.2. Backend

En cuanto al Backend de la 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**. Además proporciona un gran número 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.
- 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 proporciona acceso a una amplia gama de paquetes reutilizables, ofrece una excelente resolución de dependencias y permite automatizar gran parte de la cadena de herramientas de compilación.
- 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 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 [7].

Además se ha empleado **Express**, el framework web más popular de Node, que 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 (esto es 'GET', 'POST', 'PUT', 'DELETE', etc.) 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. [7]

En esta parte de la aplicación, se ha hecho uso de la API de administración **GraphQL** que proporciona Shopify. Esta constituye la principal forma en la que las aplicaciones pueden interactuar con la tienda, permitiendo leer y escribir la información de la tienda incluidos los productos, inventario, pedidos, envíos y mucho más, y facilitando ampliar la funcionalidad existente de Shopify. Gracias a esta API es posible conectar el inventario de la tienda con otros marketplaces, ofreciendo infinidad de posibilidades para incluir nuevas funcionalidades en el panel de control. Esta API es compatible tanto con GraphQL como con REST aunque para este desarrollo se empleará 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. [4]

#### 4.2.3. Otras tecnologías

Para la elaboración de este TFG se ha empleado **Visual Studio Code** como entorno de desarrollo tanto para la creación de la aplicación como para la redacción 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 desarrollo colaborar de manera efectiva en equipo y optimizar el proceso de desarrollo de software. Esta herramienta nos ofrece gestión del código fuente, permitiéndonos 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 funcionalidades útiles para el desarrollo como seguimiento, pruebas y gestión del ciclo de vida de los proyectos. [8]

## 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 en base a los requisitos solicitados por antiguos clientes.
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. Analizar algunas funcionalidades o requisitos adicionales que puedan requerir los futuros clientes y diseñar y dividir su implementación en tareas.
6. Implementar estas nuevas funcionalidades siguiendo las tareas para mejorar la aplicación.
7. Desplegar la aplicación en una tienda de pruebas para comprobar su funcionamiento, realizar pruebas manuales y compras ficticias.
8. Documentar todo este procedimiento para plasmarlo en el TFG.

### 5.2. Metodología

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

#### 5.2.1. Temporalización

- **Etapas 1. Análisis inicial.** Esta primera etapa comprende el objetivo 1. En este periodo se realiza un estudio de las tecnologías que necesitamos para llevar a cabo el desarrollo y en base al estudio se va realizando un primer planteamiento de la solución a desarrollar. Esta etapa se puede dividir en las siguientes fases.

- a) Estudiar la plataforma Shopify y todas sus funcionalidades, como controlar la parte de administración de una tienda, aprender a crear aplicaciones, desarrollar temas y configurarlos y demás.
  - b) Estudiar las tecnologías para la primera parte del Frontend en este caso React y sus bibliotecas.
  - c) Estudiar las tecnologías para la segunda parte del Frontend (Extensión de tema), ponerse al día con Liquid, HTML, CSS y JavaScript.
  - d) Estudiar las tecnologías para la parte del Backend en nuestro caso Node.js y su librería Express.
- **Etapas 2. Análisis de requisitos y diseño funcionalidades principales.** Esta etapa comprende los objetivos 2 y 3. Durante esta etapa se realiza un análisis de requisitos básicos que debe tener la aplicación, se crean las historias de usuario correspondientes en base a esos requisitos y se realiza un diseño de cada funcionalidad a implementar dividiendo su desarrollo en tareas.
  - **Etapas 3. Implementación de la aplicación.** Esta etapa comprende los objetivos 4, 5 y 6. Esta es la etapa mas larga del desarrollo, en ella se implementan las tareas diseñadas, y vamos cumpliendo todas esas historias de usuario y requisitos analizados previamente. La implementación tanto de la parte Backend como del Frontend se ha ido alternando y desarrollando correlativamente según estaba diseñado en las tareas. Cabe destacar que el desarrollo de esta aplicación se ha focalizado más en el Frontend que en el Backend, pues la mayor parte de funcionalidades de la misma recaen sobre la parte Front. En esta etapa se ha ido probando en una tienda de desarrollo cada una de las funcionalidades implementadas en las tareas, para ir comprobando el funcionamiento de la aplicación. Una vez desarrolladas y comprobadas todas las tareas se han diseñado, implementado y probado unas casuísticas y funcionalidades extra que los futuros clientes pueden necesitar en sus desarrollos personalizados.
  - **Etapas 4. Despliegue en tienda de muestra.** Esta etapa comprende el objetivo 7. Durante esta etapa, se instala la aplicación en una tienda de muestra y se realiza el diseño y maquetación del tema de la misma. Además se rediseña el frontend (extensión de tema) de la aplicación de modo que el diseño del personalizador de artículos encaje con el diseño general de la tienda. En esta etapa también se realizan pruebas de todo el funcionamiento del personalizador y se realizan pedidos ficticios para comprobar el correcto funcionamiento de todo el desarrollo. Gracias al trabajo de esta etapa hemos conseguido desarrollar una tienda de demostración con la funcionalidad de personalización de artículos, que podemos mostrar a esos posibles clientes para realizarles una demostración del potencial de las funcionalidades de personalización de artículos.
  - **Etapas 5. Documentación y desarrollo del TFG.** Esta última etapa comprende el objetivo 8. Durante esta etapa se lleva a cabo la documentación del

Trabajo de Fin de Grado, se redacta de una forma estructurada todo el procedimiento llevado a cabo para desarrollar la aplicación y se llevan cabo reuniones con el tutor para ir revisando y compartiendo opiniones sobre la documentación.

### 5.2.2. Proceso de desarrollo

En la implementación de este proyecto se ha seguido un proceso de desarrollo enfocado en las metodologías ágiles, caracterizado por su naturaleza iterativa e incremental y por su fácil adaptabilidad a los cambios. Como se ha podido observar en las etapas del desarrollo, hemos implementado las funcionalidades de manera iterativa, lo que nos ha permitido construir las distintas partes de la aplicación paso a paso e ir probándolas en varios ciclos. Esta práctica nos ha permitido la obtención de retroalimentación y la capacidad de realizar ajustes a lo largo del desarrollo según ha sido necesario. Además una vez implementamos y probamos las funcionalidades básicas de la aplicación, estas se ampliaron y se desarrollaron una serie de casuísticas y funciones adicionales para prepreparar la aplicación a posibles requisitos futuros. Esto demuestra la capacidad de adaptación a los cambios según las necesidades de los clientes. De hecho esta es una aplicación con una funcionalidad estandarizada, pero cada cliente solicita desarrollos personalizados que difieren bastante unos de otros, por lo que la adaptabilidad a los cambios de esta aplicación es una característica fundamental para el ámbito en el que se ha creado. Además el uso de Azure DevOps para gestionar, las historias de usuario y las tareas asociadas a esas historias, añade otro elemento característico y fundamental de las metodologías ágiles, especialmente con un enfoque como Kanban o Scrum. Esto se debe a que Azure DevOps, proporciona una interfaz visual para organizar las tareas de este proyecto, y donde todo el equipo tiene visible el progreso y desarrollo del mismo. Nos ofrece una funcionalidad de backlogs <sup>3</sup>, en la cual podemos crear y gestionar una lista prioritaria de elementos que representan el trabajo pendiente de un proyecto, en nuestro caso la aplicación de personalización de artículos. En nuestro marco de trabajo, este backlog contiene todas las funcionalidades, características, mejoras y correcciones que se desean implementar en el producto y estos elementos están descritos en forma de historias de usuario que representan las necesidades del cliente y que contienen las tareas que se deben seguir para el desarrollo. Este backlog es dinámico y evoluciona a lo largo del proyecto, y sus elementos se priorizan en función de su valor para el producto <sup>4</sup> y se ajustan continuamente en función de los cambios en los requisitos o las necesidades del proyecto. Además esta herramienta cuenta también con una funcionalidad que nos permite marcar las tareas con diferentes estados como “Nueva”, “Activa”, “Lista para pruebas”, “Testing”, “Resuelta” y “Cerrada” <sup>4</sup>. Esto nos permite un seguimiento claro y conciso del flujo de trabajo de cada tarea y nos permite una gestión eficiente del trabajo pendiente. También hay que destacar en cuanto al proceso de desarrollo, que tanto la parte de frontend como de backend se ha ido desarrollando y probando correlativamente en función de las necesidades de cada tarea y las funcionalidades a implementar, por lo que como se puede ver, este desarrollo ha ido creciendo iteración tras iteración y tarea tras tarea en todo su conjunto, mejorando en funcionalidad. Este ha sido creado de tal manera que está preparado para crecer en un futuro continuando con este proceso incremental.



Order	Work Item Type	Title	State	Story ...	Value Area	Iteration Path	Tags
3	User Story	Si un producto es de impresión, en la ficha debe haber un b...	Resolved		Business	Shopify	
	Task	Crear un theme app extensión	Resolved			Shopify	
	Task	Mostrar el botón si el producto es de impresión	Resolved			Shopify	
	Task	El texto del botón debe recuperarse de los ficheros de loc...	Resolved			Shopify	
	Task	Botón deshabilitado si falta algún elemento necesario	New			Shopify	
4	User Story	Creación de metafields y metaobjects	Resolved		Business	Shopify	
	Task	Crear a mano los metafields y metaobject necesarios	Resolved			Shopify	
	Task	POSIBLE MEJORA: Que la app cree automáticamente los ...	New			Shopify	MEJORA
5	User Story	Al pulsar el botón de añadir al pedido de impresión se debe...	Resolved		Business	Shopify	
	Task	Abrir un modal	Resolved			Shopify	
	Task	Crear con liquid una estructura javascript para tener prep...	Resolved			Shopify	
	Task	Mostrar los datos del resumen con la información por def...	Resolved			Shopify	
	Task	Cantidad	Resolved			Shopify	
	Task	Mostrar areas	Resolved			Shopify	
	Task	Imagen de area para usar cuando no tenga imagen	Resolved			Shopify	
	Task	Mostrar trabajos de cada area	Resolved			Shopify	
	Task	Mostrar campo de observaciones	Resolved			Shopify	

Figura 3: Backlogs del proyecto en Azure DevOps

1020 Crear con liquid una estructura javascript para tener preparados todos los datos de areas, trabajos, clichés...

Rubén Sánchez Fernández 0 comments Add tag

State	Resolved	Area	Shopify\App Pedidos de Impresion
Reason	New	Iteration	Shopify
Description	StandBy Client	Planning	
	StandBy Develop	Priority	2
	Ready for Test	Activity	
	Testing		
	Resolved		
	Closed		

Figura 4: Estados y prioridad de una tarea del Backlog

## 6. Diseño y resolución del trabajo realizado

### 6.1. Análisis

El análisis de este proyecto se ha centrado en la creación de historias de usuario, estas son narrativas concisas que describen la funcionalidad deseada desde la perspectiva del usuario final. Estas historias de usuario nos han servido como guía para entender las necesidades de los usuarios y traducirlas en funcionalidades concretas que nuestra aplicación debe proporcionar. Nos hemos basado en ellas para crear las tareas que hemos ido siguiendo para implementar esta aplicación.

Cada historia de usuario está compuesta por una descripción detallada de la funcionalidad requerida, así como por criterios de aceptación que nos permiten validar que la funcionalidad ha sido implementada de manera adecuada. Estas son la base sobre la cual hemos construido y hemos evaluado el éxito de nuestra aplicación.

### 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 clichés.<sup>a</sup>sociados 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.<sup>en</sup> Productos y Trabajos**

Como administrador de la tienda, quiero poder gestionar la casuística de "doble pasada.<sup>en</sup> productos y trabajos, para aplicar recargos adicionales en caso de que tanto el producto como el trabajo seleccionados tengan la opción de "doble pasada.<sup>a</sup>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.<sup>a</sup>ctivada 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 "Importe 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\_impression*.

- 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 Crear Campos Necesarios.<sup>en</sup> 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."<sup>en</sup> 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" activada. 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 "Importe 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 "normales" 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

En cuanto a las pruebas realizadas, a lo largo de todo el proceso de desarrollo de la aplicación, se han ido llevando a cabo pruebas continuas con el objetivo de verificar el correcto funcionamiento de las funcionalidades implementadas. Estas pruebas se han ido realizando de manera manual a medida que se iban complementando las diferentes tareas definidas para cada historia de usuario, de modo que la aplicación iba creciendo en funcionalidades a la par que se comprobaba su correcto funcionamiento.

Cada vez que se implementaba una nueva funcionalidad, se realizaban pruebas exhaustivas para asegurar su correcto funcionamiento antes de proceder con la siguiente tarea. Estas pruebas se han enfocado en verificar el correcto funcionamiento tanto del frontend como del backend, pues como hemos comentado anteriormente, el desarrollo de ambas partes de ha ido realizando de forma conjunta.

Es importante destacar que se optó por la realización de pruebas manuales en lugar de la implementación de pruebas automatizadas debido a que se trata de una aplicación relativamente pequeña y con una funcionalidad en la parte de backend bastante limitada, se consideró que el esfuerzo necesario y la inversión en tiempo y recursos para establecer pruebas automatizadas no justificaba su implementación. Esta decisión nos permitió una mayor flexibilidad y agilidad durante el proceso de desarrollo de nuestra aplicación.

Para realizar las pruebas manuales a la par que íbamos desarrollando las funcionalidades de la aplicación, hemos hecho uso de **Nodemon**. Esta es una herramienta de línea de comandos para Node.js, cuya función principal es monitorizar los archivos del proyecto y reiniciar automáticamente la aplicación cuando se detectan cambios en el código fuente [9]. Esta herramienta nos ha eliminado la necesidad de detener y reiniciar manualmente el servidor cada vez que hemos realizado modificaciones en el código, lo que ha agilizado el proceso de desarrollo y nos ha permitido comprobar el funcionamiento de cada tarea a la par que se implementaba, pues podíamos observar como cada cambio reciente se reflejaba automáticamente.

El uso de nodemon, nos ha facilitado tanto la implementación y pruebas de la parte de frontend como la de backend. En cuanto a la parte del frontend nos ha permitido ir creando funcionalidades y dando diseño a esta parte, pudiendo observar los resultados de cada cambio en el código en tiempo real, lo que nos ha agilizado muchísimo el desarrollo. Por lo que respecta al backend mas de lo mismo, hemos podido ir realizando modificaciones y pruebas de las funcionalidades hasta conseguir el resultado esperado, permitiendonos solucionar los errores, mejorar el código y probarlo de forma muy agil

y rápida.

Se ha hecho uso de **Postman** para llevar a cabo las pruebas de los EndPoint de la parte del backend, esta plataforma nos ha permitido enviar solicitudes HTTP a los EndPoints de nuestra aplicación, permitiendo así verificar la correcta funcionalidad de los servicios web. Asimismo, una vez implementadas las llamadas desde la parte del frontend, se realizaron pruebas adicionales para garantizar la correcta comunicación entre backend y frontend y asegurando una experiencia del usuario fluida y sin errores.

Por último añadir que durante todo el proceso de desarrollo, se han ido realizando pruebas de la parte del front del tema (theme app extensión) en diversos navegadores para probar la aplicación desde diferentes entornos y permitirnos identificar problemas de compatibilidad y rendimiento. Además hemos hecho uso de una herramienta clave con la que cuentan la mayoría de navegadores, que nos proporciona la capacidad de simular diferentes tamaños de pantalla y dispositivos actuales, lo que nos ha facilitado el poder probar el diseño responsive de nuestra aplicación a lo largo del desarrollo.

## 6.5. Despliegue

Hasta la fecha, en Upango no se ha recibido ninguna solicitud por parte de los clientes que requiera hacer uso de las funciones de personalización de productos que ofrece esta aplicación. Dado que no hemos tenido la necesidad de implementar esta aplicación en las tiendas en desarrollo para nuestros clientes hasta el momento, tampoco ha sido necesario contratar los servicios de una empresa de hosting y desplegarla en un servidor privado. Del mismo modo, tampoco hemos adquirido un dominio web público específico para esta aplicación.

Por esta razón, hemos desplegado nuestra aplicación de manera local y haciendo uso de las herramientas de las que nos proporciona Shopify. Para desplegar nuestra aplicación, hemos aprovechado las funcionalidades de la Shopify CLI. Esta herramienta cuenta con una interfaz de línea de comandos que nos proporciona diversas funcionalidades para desplegar nuestra aplicación. Empleando la Shopify CLI, hemos ejecutado el comando `npm run dev` que ejecuta el script definido en el archivo `package.json` bajo la clave `dev`. En nuestro caso como se muestra en la Figura 5 estaríamos ejecutando `shopify app dev` el cual es el comando específico para lanzar la aplicación. Este al ejecutarse inicia un servidor local tanto para el backend como para el frontend, proporcionando una dirección URL pública temporal la cual Shopify se encarga de gestionar automáticamente actualizándola según sea necesario. Esta URL nos permite lanzar y compartir nuestra aplicación para probarla durante el desarrollo, con la ventaja de que Shopify se encarga de gestionar su seguridad y privacidad.

En este proceso de despliegue, se hace uso de Cloudflare, concretamente de la funcionalidad Cloudflare Tunnel. Esta funcionalidad permite exponer servicios que se ejecutan en nuestra infraestructura local a través de la red de Cloudflare, lo que nos proporciona de una capa adicional de seguridad y rendimiento, pues en lugar de exponer directamente nuestros servidores locales a internet, los túneles de Cloudflare establecen una conexión segura a través de la red de Cloudflare y protege nuestros servidores de posibles ataques maliciosos, proporcionando una mayor confiabilidad en la entrega de contenido [10].

```

1  {} package.json > ...
2  {
3    "name": "upango-pedidos-impresion",
4    "version": "1.0.0",
5    "main": "web/index.js",
6    "license": "UNLICENSED",
7    "scripts": {
8      "shopify": "shopify",
9      "build": "shopify app build",
10     "dev": "shopify app dev",
11     "info": "shopify app info",
12     "generate": "shopify app generate",
13     "deploy": "shopify app deploy"
14   },
15   "dependencies": {}
16 }

```

Figura 5: package.json

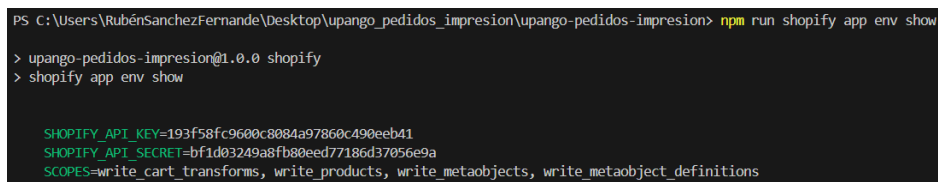
Al crear un túnel entre nuestro entorno local y la tienda de desarrollo y pruebas empleando Cloudflare, podemos permitir que los servicios que se ejecutan en nuestra infraestructura local sean accesibles a través de internet de una manera segura y aprovechándonos de la gran infraestructura de red global con la que cuenta Cloudflare para enrutar el tráfico.

En el caso de que nuestros clientes soliciten los servicios de esta aplicación y necesitemos hacer un despliegue real, seguiríamos el siguiente procedimiento que ya hemos utilizado para desplegar otro tipo de aplicaciones. En primer lugar contrataríamos los servicios de hosting para alojar nuestra aplicación en un servidor y nos haríamos con un dominio o subdominio que esté apuntando a ese servidor. Una vez tenemos el servidor, nos conectamos a él mediante SSH para proceder a configurarlo. En nuestro caso para trabajar cómodamente en este servidor remoto, hacemos uso de herramientas como **Remote Explorer** y **Remote SSH**, estas son extensiones de Visual Studio Code que nos permiten trabajar de una manera más eficiente y cómoda con los archivos del servidor empleando el editor de código. Estas herramientas nos facilitan la conexión al servidor, las configuraciones y despliegue de aplicaciones que realizamos en él, y a la hora de tener que analizar errores y solucionarlos una vez la aplicación está en producción, nos amenizan estas tareas de manipular el código.

Estando ya dentro del servidor, necesitamos instalar varias herramientas esenciales, si no están presentes en este. Entre ellas podemos destacar un servidor web como **Apache**, que es fundamental para servir nuestra aplicación al público; **npm** y **Node.js** que nos permitirán ejecutar el código JavaScript en el servidor y administrar las dependencias de nuestro proyecto a través de npm; **Git**, para poder descargar el código fuente de la aplicación desde nuestro repositorio remoto y en caso realizar modificaciones en el código del servidor poder actualizarlo en el repositorio; y **pm2**, un administrador de procesos para aplicaciones Node.js que garantiza que nuestras aplicaciones se ejecuten



de manera continua gestionando múltiples instancias y permitiendo su reinicio en caso de fallos. Una vez tenemos preparado el entorno, descargamos la aplicación desde el repositorio remoto e instalamos las dependencias necesarias ejecutando `npm install`. Después debemos buscar las variables de entorno que genera y configura Shopify al crear una aplicación, estas se pueden obtener de varias maneras, o bien a través del panel de partners de Shopify o introduciendo el comando **`npm run shopify app env show`**. En la Figura 6 podemos observar dichos tokens. Para poder tener diferentes aplicaciones en el servidor con distintas variables, en lugar de crear variables de entorno como tal, se las pasaremos como parámetros al proceso de node.



```
PS C:\Users\RubénSanchezFernande\Desktop\upango_pedidos_impresion\upango-pedidos-impresion> npm run shopify app env show
> upango-pedidos-impresion@1.0.0 shopify
> shopify app env show

SHOPIFY_API_KEY=193f58fc9600c8084a97860c490eeb41
SHOPIFY_API_SECRET=bf1d03249a8fb80eed77186d37056e9a
SCOPES=write_cart_transforms, write_products, write_metaobjects, write_metaobject_definitions
```

Figura 6: Variables de entorno de la aplicación

A continuación debemos compilar los archivos del frontend, pues en producción debemos compilar los ficheros de react que hay en el directorio `web/frontend` para generar la carpeta `web/frontend/dist` que contiene los compilados. Al hacer esto, ejecutamos `npm install` para instalar las dependencias del frontend como vite, indicamos el valor de la siguiente variable de entorno que necesita para compilar `export SHOPIFY_API_KEY=193f58fc9600c8084a97860c490eeb41` y por último ejecutamos `npm run build` para realizar la compilación.

Una vez hecho esto ya podemos ejecutar la aplicación en modo producción, para ello debemos ejecutar el comando `npm run serve` desde el directorio `web`. Debemos acompañar a este comando de unos parámetros para que cuente con sus variables de entorno, estas variables son la `SHOPIFY_API_KEY`, el `SHOPIFY_API_SECRET`, los `SCOPES`, el `host`, es decir la url de la app y el puerto.

Además para que la app esté siempre en ejecución, debemos lanzar el comando `npm run serve` con sus variables a través de un `pm2 start` y añadiendo un parámetro `name` al final para darle un nombre al proceso y poder identificarlo y gestionarlo.

Llegados a este punto nos quedaría configurar el dominio de la aplicación en el servidor, para ello primero debemos modificar la configuración del servidor web Apache en nuestro caso, para que el dominio apunte al servidor y se haga un mapeo al puerto correspondiente de la aplicación. Esto implica editar archivos de configuración del servidor como el archivo `http_node.conf`. A continuación se deben crear y configurar los certificados SSL para admitir HTTPS, empleando herramientas como Let's Encrypt para generar certificados gratuitos, y una vez configurados los certificados, actualizamos la configuración del servidor para que acepte conexiones HTTPS en el puerto 443. Una vez hecho esto debemos reiniciar Apache para que los cambios surtan efecto y la aplicación pueda ser accedida correctamente a través del dominio configurado.

Por último, debemos ajustar la configuración de la aplicación desde el panel de administración de partners para que la aplicación reconozca el nuevo dominio (Figura ??)

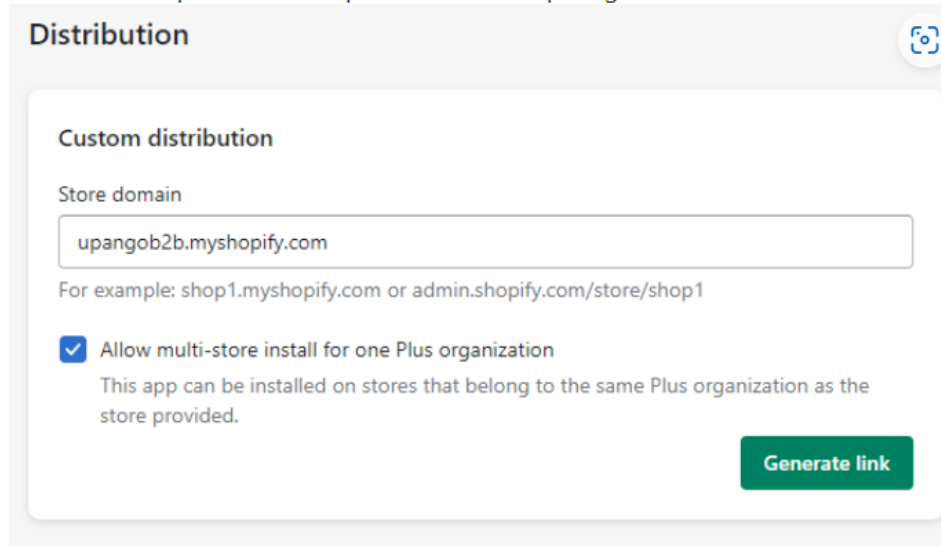
y generar un enlace de instalación de esta aplicación para instalarla en la tienda que lo solicite (Figura 8, Figura 9).

The screenshot shows the 'Configuración' (Configuration) page in the Shopify Partners interface. The left sidebar contains a navigation menu with options: '← TODAS LAS APLICACIONES', 'upango\_pedidos\_impresion', 'Resumen', 'Crear', 'Configuración' (highlighted), 'Acceso a la API', 'Extensiones', 'Publicar', 'Versiones', 'Distribución', 'Supervisar', and 'Información útil'. The main content area is titled 'Nombre de la aplicación' and 'URL'. It contains several input fields and sections: 'Nombre de la aplicación' (upango\_pedidos\_impresion, 24/30), 'Identificador de la aplicación' (upango\_pedidos\_impresion), 'URL de la aplicación' (https://clarke-writers-manga-velocity.trycloudflare.com/), 'URL de preferencias (opcional)' (https://example.com/preferences), and 'Autorización de redireccionamiento de URL(s)' (https://clarke-writers-manga-velocity.trycloudflare.com/auth/callback, https://clarke-writers-manga-velocity.trycloudflare.com/api/auth/callback, https://clarke-writers-manga-velocity.trycloudflare.com/shopify/graphql/auth/callback). There are also links to 'actualizar desde Shopify CLI'.

Figura 7: Configuración de la aplicación

The screenshot shows the 'Distribución' (Distribution) page in the Shopify Partners interface. The left sidebar is the same as in Figure 7, with 'Distribución' highlighted. The main content area is titled 'Distribución' and 'Seleccionar método de distribución'. It contains two options: 'Distribución pública' (Public distribution) and 'Distribución personalizada' (Custom distribution). The 'Distribución personalizada' option is selected, indicated by a blue border and a radio button. Below the options is a green 'Seleccionar' (Select) button. At the bottom, there is a link: 'Más información sobre [distribución de la aplicación](#)'.

Figura 8: Distribución de la aplicación



The screenshot shows a web interface titled "Distribution" with a QR code icon in the top right corner. Below the title is a section labeled "Custom distribution". It contains a text input field for "Store domain" with the value "upangob2b.myshopify.com". Below the input field is a note: "For example: shop1.myshopify.com or admin.shopify.com/store/shop1". There is a checked checkbox labeled "Allow multi-store install for one Plus organization" with a sub-note: "This app can be installed on stores that belong to the same Plus organization as the store provided." At the bottom right of the form is a green button labeled "Generate link".

Figura 9: Generar link de distribución

## 7. Conclusiones y vías futuras

### 7.1. Conclusiones

A modo de conclusión podemos decir que en este Trabajo de Fin de Grado hemos logrado cumplir todos los objetivos propuestos inicialmente. Hemos diseñado y desarrollado una aplicación para la plataforma Shopify que permite la personalización de productos para su posterior compra en las tiendas de los clientes que la empleen. Desde el análisis inicial hasta la implementación y despliegue, hemos seguido una metodología estructurada que nos ha permitido alcanzar cada hito de manera efectiva y eficiente.

Además hemos desarrollado una aplicación con unas funcionalidades estándar al mercado y a los posibles requisitos que nos pueden solicitar en un futuro los clientes. La hemos implementado de tal manera que está preparada para añadir y modificar las funcionalidades fácilmente, lo que la hace adaptable a los requisitos de futuros clientes que pueden necesitar características específicas de esta herramienta de personalización de productos. Hemos creado una base sólida sobre la cual podemos construir y expandir la aplicación según las necesidades del mercado y los clientes.

El análisis detallado de los objetivos, la metodología ágil implementada y el enfoque iterativo e incremental que hemos aplicado, nos han permitido no solo cumplir con los requisitos establecidos, sino que también anticiparnos a posibles necesidades futuras y preparar la aplicación para seguir creciendo de manera continua, aplicando más iteraciones.

### 7.2. Trabajo futuro

---

## 8. Bibliografía

- [1] ePages. *ePages*, dirección: <https://epages.com/es/> (visitado 21-04-2024).
- [2] Shopify Tutorial. *Shopify*, dirección: <https://www.shopify.com/es/blog/tutorial-shopify> (visitado 29-03-2024).
- [3] Shopify. *Shopify*, dirección: <https://www.shopify.com/es/> (visitado 29-03-2024).
- [4] Shopify Developers. *Shopify.dev*, dirección: <https://shopify.dev/> (visitado 29-03-2024).
- [5] React *React*, dirección: <https://ebac.mx/blog/que-es-react> (visitado 29-03-2024).
- [6] Funcionamiento React *Funcionamiento React*, dirección: <https://kinsta.com/es/base-de-conocimiento/que-es-react-js/#qu-es-react> (visitado 29-03-2024).
- [7] Node y Express *Node y express*, dirección: [https://developer.mozilla.org/es/docs/Learn/Server-side/Express\\_Nodejs/Introduction](https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction) (visitado 31-03-2024).
- [8] Azure DevOps *Azure DevOps*, dirección: [https://aodatacloud.es/blog/que-es-azure-devops-y-para-que-sirve/#%C2%BFQue\\_es\\_Azure\\_DevOps](https://aodatacloud.es/blog/que-es-azure-devops-y-para-que-sirve/#%C2%BFQue_es_Azure_DevOps) (visitado 31-03-2024).
- [9] Nodemon *Nodemon*, dirección: <https://geeknomada.blog/nodejs/que-es-nodemon-en-node-js/> (visitado 09-04-2024).
- [10] Cloudflare *Cloudflare*, dirección: <https://www.cloudflare.com/es-es/products/tunnel/> (visitado 11-04-2024).

## 9. Anexos

### 9.1. Manual de usuario

En este apartado se mostrará y explicará de una forma visual la forma de uso de la aplicación desarrollada. Se mostrará tanto el proceso para integrar la aplicación con una tienda, como su funcionamiento dentro de la misma.

En primer lugar, una vez instalada la aplicación en la tienda, debemos dirigirnos al panel de administración. En la Figura 10 podemos observar este panel de administración, donde debemos ir al apartado *Canales de venta* → *Tienda online* → *Tema*. En esta pantalla podemos ver y configurar toda la librería de temas que tenemos disponible. En nuestro caso tenemos publicado el tema *upango-impresion-theme*, por lo que vamos a pulsar en el botón personalizar de este tema para que se nos muestre el configurador de tema y poder añadir al tema las funcionalidades de nuestra aplicación.

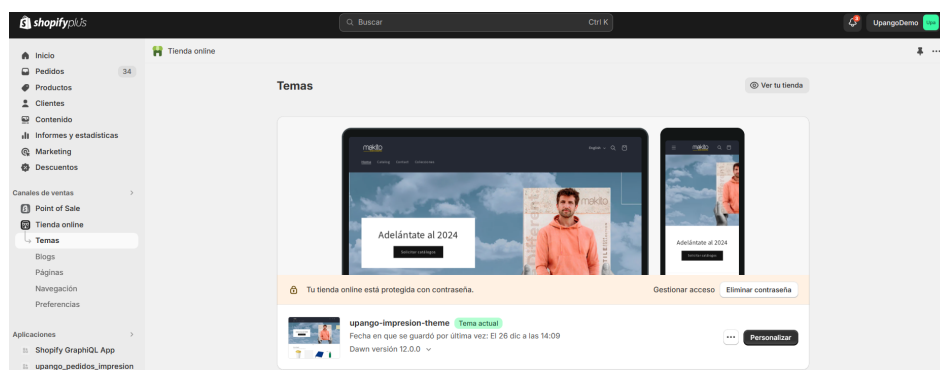


Figura 10: Personalizar tema en el panel de administración

Al instalar la app, se nos habilitan en el configurador todos aquellos componentes de tema que se han desarrollado en la aplicación. En la Figura 11 podemos observar la pantalla del configurador del tema, en ella debemos ir a la plantilla de la página de producto y añadir en el lugar que consideremos más apropiado el bloque que hemos desarrollado en la extensión de tema de la aplicación. Como puede verse, al introducir este elemento, se nos muestra en la pantalla de producto el botón para abrir el personalizador de artículos que lleva detrás toda la funcionalidad.

Una vez añadida la extensión del tema, debemos dirigirnos a la pantalla del panel de administración que hemos desarrollado en la aplicación. En la Figura 12 podemos ver un enlace a esta pantalla en la sección *Aplicaciones* → *upango\_pedidos\_impresion*. En la Figura 13 observamos dicha pantalla, en ella podemos pulsar el botón *Crear Metadatos* para que se creen automáticamente los metacampos de producto necesarios, y el botón *cart-transform* para que se cree automáticamente en la tienda el objeto cart-transform, el cual hará uso de una función de la aplicación, que dotará al carrito y al checkout de la tienda de una funcionalidad y apariencia de paquetes customizados.

Una vez realizadas estas configuraciones y creados en la tienda todos los productos necesarios, trabajos de impresión, tipos de cliché, asociados los productos con sus áreas

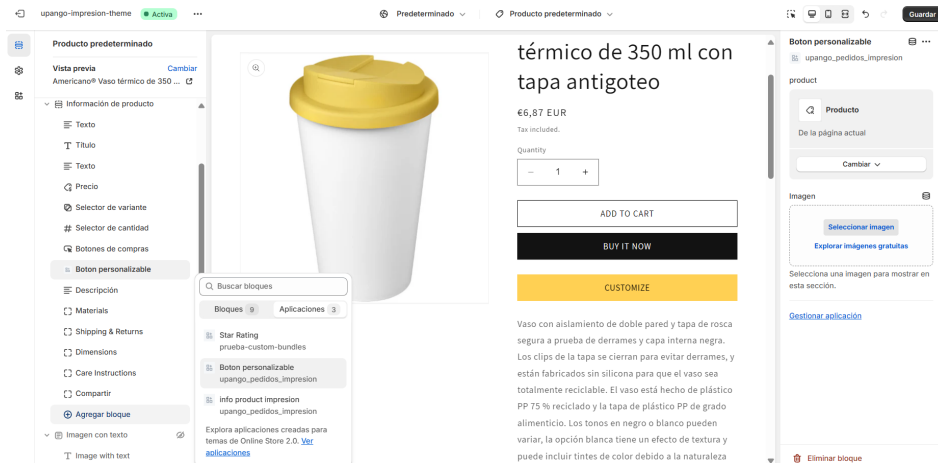


Figura 11: Añadir la extensión de tema de la App en el tema de la tienda

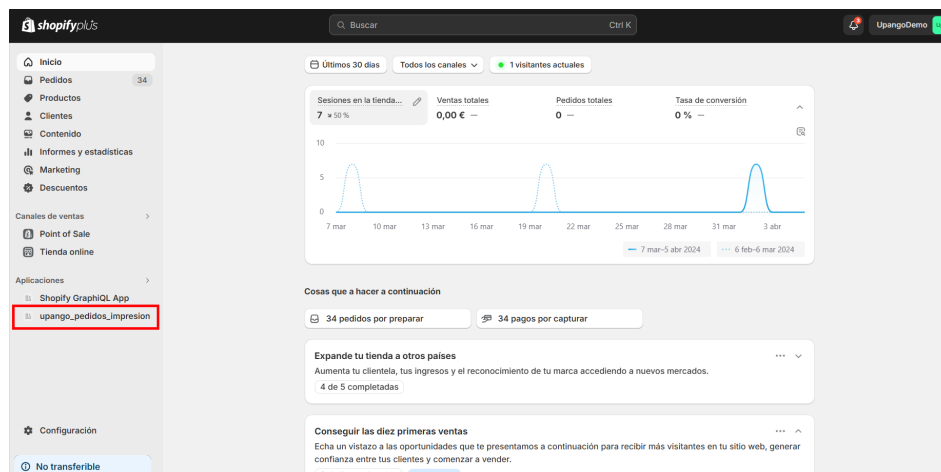


Figura 12: Panel de administración de la tienda

de impresión, y todos los elementos necesarios relacionados con la impresión de artículos, ya podemos hacer uso de la tienda con las funcionalidades de personalización de productos integrada.

En la Figura 14 podemos observar la página principal de la tienda, en ella podemos navegar hacia otros apartados de la tienda, como el catálogo de productos, podemos buscar secciones y productos con el buscador, ir al carrito, seleccionar el idioma y otras funcionalidades. También podemos seleccionar cualquier producto de los que nos aparece en destacados para ir a la página de producto y poder añadirlo al carrito o personalizarlo si se trata de un producto personalizable.

En la imagen 15 podemos ver como hemos navegado hacia una página de producto. En esta, como puede verse, no aparece el botón para personalizar este artículo, lo que significa que el producto no es personalizable y en este caso solo podríamos seleccionar la cantidad deseada y añadir al carrito o comprar directamente. En la siguiente imagen (Figura 16) podemos ver otra página de producto, y en este caso si nos aparece el botón de Personalizar, lo que nos indica que este producto si sería personalizable y cuenta

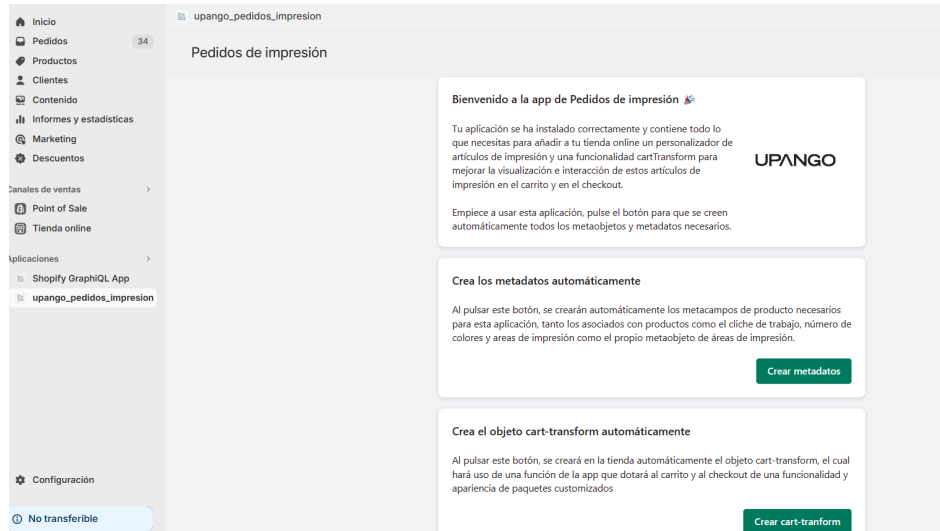


Figura 13: Funcionalidades de la parte de administración de la App

con sus áreas de impresión correspondientes y todos sus atributos de personalización. En esta pantalla podemos comprar este producto y tratarlo como en el caso anterior, y comprarlo sin personalización, o podemos seleccionar la cantidad que deseemos y pulsar el botón de personalizar para configurar y añadir al carrito los productos a través de la interfaz de personalización.

Si pulsamos en el botón de Personalizar, se nos abrirá el personalizador de artículos. En este, como puede en la Figura 17, nos encontramos las distintas áreas de impresión con las que cuenta el producto con sus atributos personalizables y a la derecha podemos ir viendo el resumen del pedido personalizado actualizado. Como se puede observar, ahora mismo en el resumen del pedido solamente nos aparece el producto con las unidades, el precio por unidad y el precio total, pues todavía no hemos seleccionado ni configurado ningún área del producto.

Si seleccionamos un área de impresión, como podemos observar en la Figura 18, podemos ver como se refresca el resumen de pedido y se le agrega tanto el trabajo de impresión seleccionado por defecto, como su cliché asociado. Una vez seleccionada el área podemos configurar todos los elementos que contiene. Además también se pueden seleccionar varias áreas en una misma personalización, como podemos ver en la Figura 19. En ella aparecen varias áreas del producto seleccionadas y en el resumen aparecen reflejados los trabajos y clichés de dichas áreas y su importe total.

En cuanto a la configuración de cada área, como se puede apreciar en la Figura 20, tenemos un par de inputs para introducir las medidas de la impresión que deseamos realizar en el artículo, uno para el ancho y otro para el alto. Además podemos observar un select para seleccionar el tipo de trabajo a aplicar en el producto seleccionado, es decir el tipo de marcaje. El tipo de trabajo seleccionado afectará al selector de colores de su derecha, pues cada trabajo de impresión cuenta con un número máximo de colores con los que puede trabajar, y al modificar el trabajo se renderizará automáticamente el selector de colores. También tenemos unos inputs en los que podemos introducir los colores con los que queremos que se realice la impresión, existen tantos inputs como

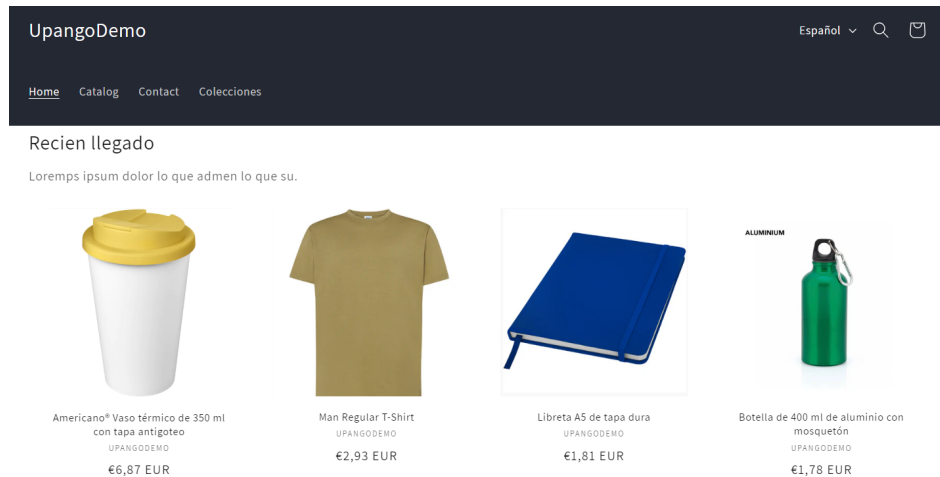


Figura 14: Página principal de la tienda de demostración



Figura 15: Página de producto no personalizable

colores hayamos seleccionado en el select de colores y estos se refrescan al modificar el valor de ese select. En este caso hemos seleccionado el tipo de trabajo *transfer serigráfico*, también hemos introducido que queremos una impresión en 3 colores y hemos seleccionado dichos colores en los inputs. En el resumen de pedido podemos observar como aparece el trabajo de impresión, su cliché asociado y un cargo extra por la adición de 2 colores extra.

En la personalización del área de impresión, también podemos observar dos checks, uno check de repetición de cliché para marcar si ya hemos realizado algún otro pedido similar con el mismo tipo de impresión y otro check de doble pasada para indicar si queremos que la impresión se aplique dos veces. En la Figura 21 podemos apreciar estos dos checks marcados y como puede verse, estos afectan al precio total de impresión. En el resumen puede observarse como al haber marcado el check de repetición de cliché, el precio del cliché se ha reducido debido a que la empresa ahorraría costes al no tener que



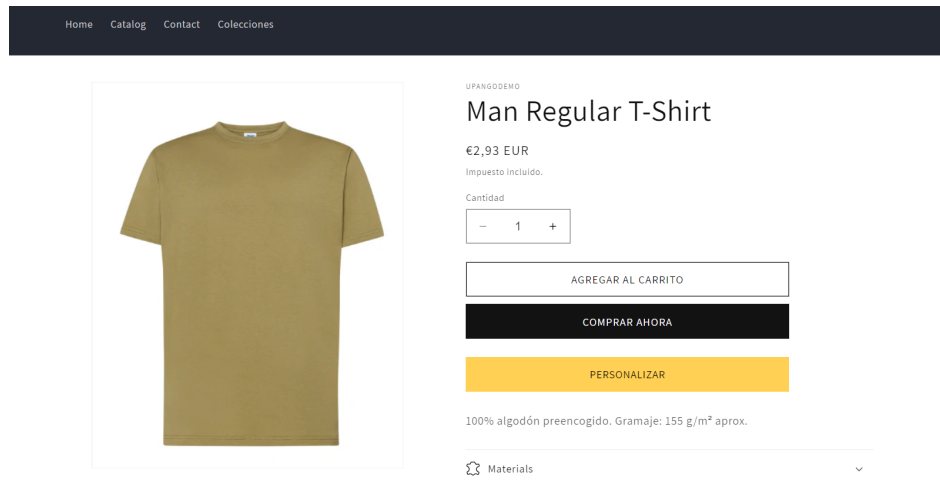


Figura 16: Página de producto personalizable

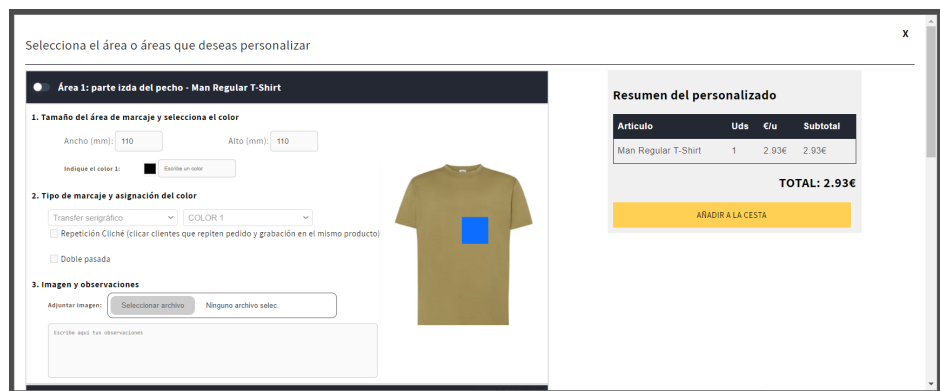


Figura 17: Pantalla del personalizador del producto seleccionado

fabricar un nuevo cliché. Además al haber marcado el check de doble pasada podemos observar como en el resumen aparece un cargo extra por doble pasada.

También podemos seleccionar el logo que se desee imprimir en el producto. En la imagen 22 podemos observar como se ha seleccionado dicha imagen y se nos muestra de forma visual en el personalizador una aproximación de como quedaría ese logo impreso en la prenda que queremos comprar. Esta funcionalidad permite al cliente observar una imagen lo mas realista posible de cual sería el resultado final de la personalización del artículo. Además el cliente también puede introducir unas observaciones referidas a la personalización para dar mas detalles de la misma y tratar ciertos detalles que se nos puedan estar escapando. Una vez personalizada a nuestro gusto la prenda, podemos pulsar el botón de añadir el carrito para añadir los productos con toda su personalización asociada y acorde con lo visualizado en el resumen del pedido.

Una vez añadido el producto personalizado al carrito podemos pulsar el botón de la cesta de la compra ubicado en la parte superior de la página (Figura 23) para dirigirnos a la página del carrito. En dicha página (Figura 24) podemos observar los productos que han sido añadidos. Estos productos que han sido personalizados se añaden al carrito

**Área 1: parte izda del pecho - Man Regular T-Shirt**

1. Tamaño del área de marcaje y selecciona el color  
 Ancho (mm): 110 Alto (mm): 110  
 Indique el color 1:  Escriba un color

2. Tipo de marcaje y asignación del color  
 Transfer serigráfico ☐ Repetición Cliché (clicar clientes que repiten pedido y grabación en el mismo producto)  
☐ Doble pasada

3. Imagen y observaciones  
 Adjuntar imagen:  Ninguno archivo selec.  
 Escriba aquí sus observaciones

**Resumen del personalizado**

Artículo	Uds	€/u	Subtotal
Man Regular T-Shirt	1	2.93€	2.93€
<b>Área 1: parte izda del pecho - Man Regular T-Shirt</b>			
Transfer serigráfico	1	2.61€	2.61€
CLICHE FOTOLITO PANTALLA	1	50€	50€
<b>TOTAL: 55.54€</b>			

Figura 18: Personalizador con el Area 1 seleccionada

**Área 1: parte izda del pecho - Man Regular T-Shirt**

1. Tamaño del área de marcaje y selecciona el color  
 Ancho (mm): 110 Alto (mm): 110  
 Indique el color 1:  Escriba un color

2. Tipo de marcaje y asignación del color  
 Transfer serigráfico ☒ Repetición Cliché (clicar clientes que repiten pedido y grabación en el mismo producto)  
☒ Doble pasada

3. Imagen y observaciones  
 Adjuntar imagen:  Ninguno archivo selec.  
 Escriba aquí sus observaciones

**Resumen del personalizado**

Artículo	Uds	€/u	Subtotal
Man Regular T-Shirt	3	2.93€	8.79€
<b>Área 1: parte izda del pecho - Man Regular T-Shirt</b>			
Transfer serigráfico	3	2.61€	7.83€
Doble pasada	3	0.2€	0.6€
CLICHE FOTOLITO PANTALLA	1	25€	25€
<b>Área 2: parte dcha del pecho - Man Regular T-Shirt</b>			
Transfer serigráfico	3	2.61€	7.83€
CLICHE FOTOLITO PANTALLA	1	50€	50€
<b>TOTAL: 100.05€</b>			

**Área 2: parte dcha del pecho - Man Regular T-Shirt**

Figura 19: Personalizador con varias áreas seleccionadas

en forma de un solo paquete que contiene tanto las unidades de los artículos, como todos los elementos y atributos de la personalización. En la Figura 24 puede observarse un paquete de personalización que hemos añadido, en el cual se han seleccionado 3 unidades del producto, con el trabajo *transfer serigráfico* y una serie de configuraciones que pueden verse mejor detalladas en la página del Checkout. Por último vamos a pulsar el botón del carrito para completar la compra y vamos a dirigirnos al checkout donde se podrá realizar el pedido. Por último, en la Figura 25 podemos observar dicha interfaz para completar el proceso de compra, en la cual se puede visualizar con todo detalle el resumen de todo el pedido, introducir los datos de pago, facturación y dirección de envío y completar el pedido.

**Área 1: parte izda del pecho - Man Regular T-Shirt**

1. Tamaño del área de marcaje y selecciona el color

Ancho (mm): 110 Alto (mm): 110

Indique el color 1: ■ #e60707

Indique el color 2: ■ #e6c011

Indique el color 3: ■ #479d00

2. Tipo de marcaje y asignación del color

Transfer serigráfico COLOR 3

☐ Repetición Cliché (clicar clientes que repiten pedido y grabación en el mismo producto)

☐ Doble pasada

3. Imagen y observaciones

Adjuntar imagen: Seleccionar archivo Ninguno archivo selec.

ESCRIBE AQUÍ TUS OBSERVACIONES

**Resumen del personalizado**

Artículo	Uds	C/u	Subtotal
Man Regular T-Shirt	1	2.93€	2.93€
<b>Área 1: parte izda del pecho - Man Regular T-Shirt</b>			
Transfer serigráfico	1	2.61€	2.61€
Color/es Extra x 2	2	0.2€	0.4€
CLICHE FOTOLITO PANTALLA	3	50€	150€
<b>TOTAL:</b>			<b>155.94€</b>

AÑADIR A LA CESTA

Figura 20: Personalizador con trabajo de impresión y colores seleccionados

**Área 1: parte izda del pecho - Man Regular T-Shirt**

1. Tamaño del área de marcaje y selecciona el color

Ancho (mm): 110 Alto (mm): 110

Indique el color 1: ■ #e60707

Indique el color 2: ■ #e6c011

Indique el color 3: ■ #479d00

2. Tipo de marcaje y asignación del color

Transfer serigráfico COLOR 3

☒ Repetición Cliché (clicar clientes que repiten pedido y grabación en el mismo producto)

☒ Doble pasada

3. Imagen y observaciones

Adjuntar imagen: Seleccionar archivo Ninguno archivo selec.

ESCRIBE AQUÍ TUS OBSERVACIONES

**Resumen del personalizado**

Artículo	Uds	C/u	Subtotal
Man Regular T-Shirt	1	2.93€	2.93€
<b>Área 1: parte izda del pecho - Man Regular T-Shirt</b>			
Transfer serigráfico	1	2.61€	2.61€
Color/es Extra x 2	2	0.2€	0.4€
Doble pasada	1	0.2€	0.2€
CLICHE FOTOLITO PANTALLA	3	25€	75€
<b>TOTAL:</b>			<b>81.14€</b>

AÑADIR A LA CESTA

Figura 21: Personalizador con opciones de repetición de cliché y doble pasada

**Área 1: parte izda del pecho - Man Regular T-Shirt**

1. Tamaño del área de marcaje y selecciona el color

Ancho (mm): 110 Alto (mm): 110

Indique el color 1: ■ #eb0000

2. Tipo de marcaje y asignación del color

Transfer serigráfico COLOR 1

☒ Repetición Cliché (clicar clientes que repiten pedido y grabación en el mismo producto)

☒ Doble pasada

3. Imagen y observaciones

Adjuntar imagen: Seleccionar archivo logo-black.png

Observaciones

**UPANGO**

Figura 22: Personalizador con selección de logo



Figura 23: Botón de carrito

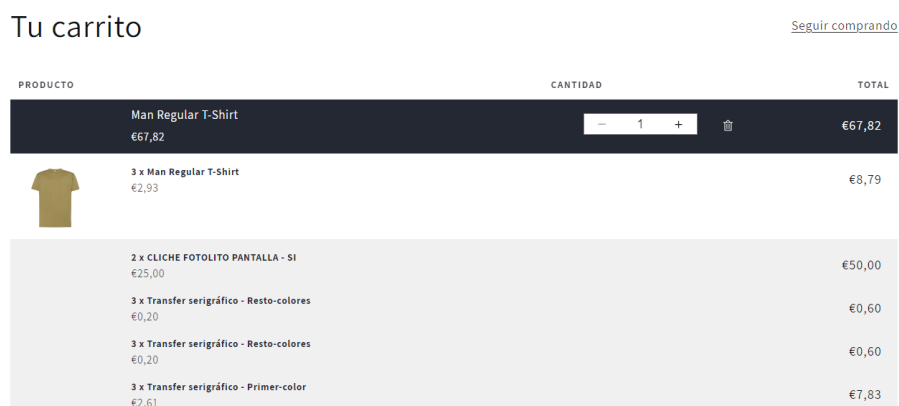


Figura 24: Pantalla del carrito de la tienda

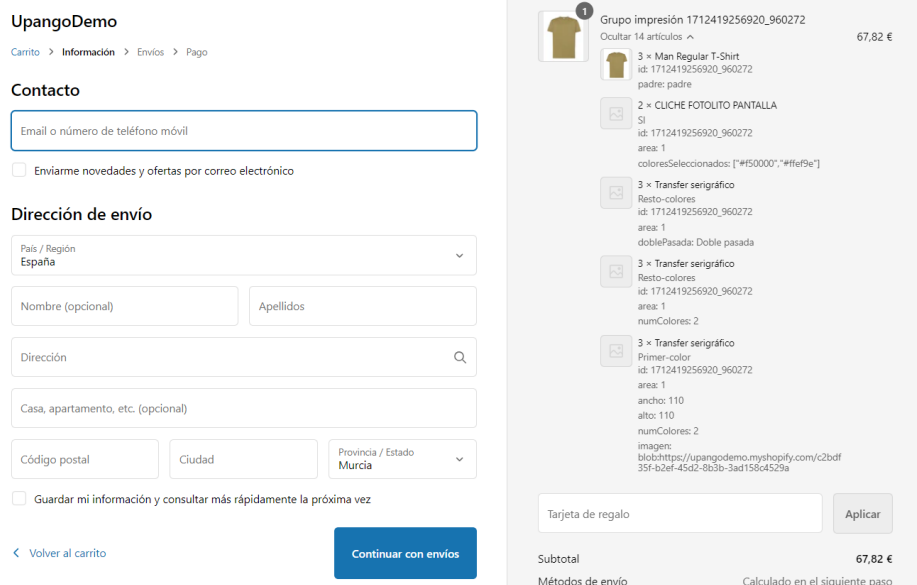


Figura 25: Pantalla de checkout de la tienda