

Portfolio y tienda online para pequeños artistas

Memoria de Proyecto Final de Grado/Máster

Máster Universitario en Desarrollo de sitios y aplicaciones web

TFM / Itinerario profesional

Autor: Eduardo Rey Jara

Consultor: Daniel García Nebot

Profesor: César Pablo Córcoles Briongos

6 de junio de 2022

Créditos, licencia Creative Commons



Esta obra, y su código fuente sito en el repositorio público <https://github.com/SedueRey/nuxt-directus-shop> están sujetos a la [licencia de Reconocimiento-Compartir Igual 4.0 Internacional](#)

Enlace a la demo totalmente estática: <https://nuxt-directus-shop-demo.netlify.app/>

Enlace a la demo conectada con Directus: <https://nuxt-directus-shop.sedurey.com/>

Abstract

Este proyecto ha tratado, desde un principio, de devolver el control sobre sus portfolios y tiendas a los artistas y pequeños artesanos. No pretende ser, todavía, un sustitutivo de otros softwares similares como Wordpress, pero sí tener de una forma sencilla una cartera de trabajos y una tienda a disposición de quien pueda necesitarla, utilizando tecnologías, entre ellas la pasarela de pago, que minimicen los gastos a aquellos artistas que autores que están empezando.

Este proyecto también intenta dar algo de independencia, aunque no pueda ser total, a estas personas con respecto a corporaciones y redes sociales con las que los creadores no siempre tienen por qué estar de acuerdo.

Para el desarrollo del mismo se han utilizado diferentes tecnologías. Un headless cms, Directus, como backend, del que se leen todos los datos. Y como frontend, Nuxt, generando una web totalmente estática basada en HTML, CSS y JavaScript, pero que, a la misma vez, puede actuar como tienda, minimizando gastos en hosting. Todo ello basado y compartido, como software libre y gratuito.

Palabras clave: tienda web, artistas, directus, Nuxt, VueJS, Portfolio, diseño adaptable, software libre, gestor de contenido, API Rest.

Abstract (english version)

This project has been made having in mind to give back control over their portfolios and stores to artists and small artisans. It's not intended to be, at least at this moment, a substitute for other similar software like Wordpress, but to have in a simple way a portfolio and a store available to whoever may need it, using technologies, including the pay platforms, that could minimize costs to those, artists and authors, who are just starting.

This project also tries to give some independence, although it can't be absolute, to these people when we are referring to big companies (including social media ones) with which the creators do not always need to agree.

Different technologies have been used to complete the development of this project. A headless cms, Directus, used as backend, from which all the data is read. And as frontend, Nuxt, generating a totally static web based on HTML, CSS and JavaScript, but also, at the same time, which could act as a full store, minimizing hosting costs. All this based and shared, as free and open source software.

Keywords: web store, artists, directus, Nuxt, VueJS, Portfolio, responsive design, open source.

Agradecimientos

A Sonia, mi mejor mitad, que ha estado a mi lado no sólo con mi vuelta tardía a los estudios de grado y máster sino en la afición al dibujo, dándome siempre todo su apoyo.

Índice

| | |
|---|----|
| 1. Introducción | 10 |
| 1.1 Contexto y justificación del Trabajo | 10 |
| 1.2 Objetivos del trabajo | 10 |
| 1.3 Enfoque y método seguido | 11 |
| 1.4 Planificación del trabajo | 11 |
| 2. Objetivos | 13 |
| 2.1 Principales..... | 13 |
| 2.2 Secundarios | 13 |
| 3. Escenario | 14 |
| 3.1 Redes sociales..... | 14 |
| 3.2 Intereses del autor frente a las grandes empresas | 14 |
| 3.3 Control de datos..... | 15 |
| 4. Contenidos..... | 17 |
| 5. Metodología | 18 |
| 5.1 Metodología Scrum | 18 |
| 5.2 Git Flow | 19 |
| 6. Arquitectura de la aplicación/sistema/servicio..... | 22 |
| 6.1 Backend | 22 |
| 6.2 Frontend..... | 24 |
| 7. Plataforma de desarrollo | 26 |
| 7.1 NGrok..... | 26 |
| 7.2 Hosting | 26 |
| 8. Planificación..... | 27 |
| 9. Proceso de trabajo/desarrollo..... | 29 |
| 9.1 Proceso de trabajo sobre el headless CMS. | 29 |
| 9.2 Proceso de trabajo en Frontend | 33 |
| 10. Prototipos..... | 42 |
| 10.1 Prototipos Lo-Fi..... | 42 |
| 10.2 Elección de paleta de colores y tipografías | 46 |
| 11. Perfiles de usuario | 49 |
| 11.1 Artista | 49 |
| 11.2 Administrador (opcional) | 49 |

| | |
|--|----|
| 11.3 Cliente | 49 |
| 12. Usabilidad/UX | 50 |
| 12.1 Sitemap | 50 |
| 12.2 Estructura de navegación de un cliente..... | 50 |
| 13. Seguridad | 53 |
| 14. Tests de aceptación..... | 54 |
| 14.1 Batería de pruebas como cliente | 54 |
| 14.2 Batería de pruebas como artista / administrador | 56 |
| 15. Instrucciones..... | 58 |
| 15.1 Instrucciones de instalación/implantación | 58 |
| 15.2 Instrucciones de uso de directus. | 61 |
| 16. Proyección a futuro..... | 62 |
| 17. Presupuesto..... | 63 |
| 18. Análisis de mercado | 64 |
| 18.1 Público objetivo | 64 |
| 18.2 Competencia | 64 |
| 19. Conclusiones | 66 |
| Anexo 1. Entregables del proyecto..... | 67 |
| Anexo 2. Código fuente (extractos)..... | 68 |
| Anexo 3. Capturas de pantalla | 76 |
| Anexo 4. Bibliografía..... | 80 |
| Anexo 5. Vita | 81 |

Figuras y tablas

Índice de figuras

| | |
|---|----|
| Ilustración 1: Flujo de trabajo SCRUM. Origen: Platzi..... | 18 |
| Ilustración 2: Uso de Git Flow. Atlassian (2022)..... | 20 |
| Ilustración 3: Estructura de BBDD del proyecto | 23 |
| Ilustración 4: Página de autenticación del backend directus..... | 31 |
| Ilustración 5: Listado de estructura y modelo de datos configurado en el backend | 32 |
| Ilustración 6: Página de artículo, tamaño escritorio..... | 42 |
| Ilustración 7: Página de artículo, tamaño Tablet y móvil | 43 |
| Ilustración 8: Página de categoría y etiquetas, versiones escritorio y móvil..... | 44 |
| Ilustración 9: Listados generales, tamaños Tablet y móvil | 44 |
| Ilustración 10: Página de producto, móvil y otras resoluciones..... | 45 |
| Ilustración 11: Estructura de la tipografía seleccionada | 46 |
| Ilustración 12: Paletas de colores oscuros y claros..... | 47 |
| Ilustración 13: Tipografías y paleta de colores con todas las opciones añadida al backend, modificable por los artistas | 48 |
| Ilustración 14: Sitemap de la tienda de este proyecto | 50 |
| Ilustración 15: Flujo de navegación de un cliente de la tienda | 51 |
| Ilustración 16: Crear un nuevo proyecto en Netlify..... | 60 |
| Ilustración 17: Lugar donde se puede arrastrar el directorio resultante. | 60 |
| Ilustración 18: Ejemplo de pipelines correctos y fallidos en Netlify | 61 |
| Ilustración 19: Comprobación de que el BodySchema enviado en la petición es correcto. | 68 |
| Ilustración 20: Comprobación y limpieza de XSS..... | 68 |
| Ilustración 21: Hook de directus para limpiar la URL y añadir fecha si no se ha añadido. | 69 |
| Ilustración 22: Descarga de imágenes a la carpeta assets. | 70 |
| Ilustración 23: Descarga de las colecciones en ficheros estáticos..... | 71 |
| Ilustración 24: Fichero de configuración para la descarga de assets..... | 72 |
| Ilustración 25: Conexión entre las páginas de Nuxt y los datos almacenados | 73 |
| Ilustración 26: Configuración de la pasarela de pago Stripe dentro de Nuxt..... | 74 |
| Ilustración 27: Microdata de las páginas de artículos..... | 75 |
| Ilustración 28: Página de inicio en esquema de colores oscuros..... | 76 |
| Ilustración 29: Página de inicio en esquema de colores claros..... | 76 |
| Ilustración 30: Lista de productos en castellano..... | 77 |
| Ilustración 31: Página de detalle de producto añadido al carrito..... | 78 |
| Ilustración 32: Carrito de la compra completo y preparado para pagar..... | 79 |
| Ilustración 33: Pasarela de pago Stripe..... | 79 |

Índice de tablas

| | |
|--|----|
| Tabla 1: Diagrama de Gantt del proyecto..... | 28 |
| Tabla 2: Prueba de aceptación 01 como cliente | 54 |
| Tabla 3: Prueba de aceptación 02 como cliente | 54 |
| Tabla 4: Prueba de aceptación 03 como cliente | 55 |
| Tabla 5: Prueba de aceptación 04 como cliente | 55 |
| Tabla 6: Prueba de aceptación 01 como artista | 56 |
| Tabla 7: Prueba de aceptación 02 como artista | 56 |
| Tabla 8: Prueba de aceptación 03 como artista | 56 |
| Tabla 9: Prueba de aceptación 04 como artista | 57 |
| Tabla 10: Prueba de aceptación 05 como artista | 57 |

1. Introducción

La siguiente memoria pretende describir y dejar constancia escrita de los requisitos y objetivos planteados, las metodologías y procesos de trabajo definidos, así como las decisiones adoptadas a lo largo del desarrollo del propio Trabajo Fin de Máster del Máster Universitario en Desarrollo de sitios y aplicaciones web.

1.1 Contexto y justificación del Trabajo

El tema sobre el que se elabora este Trabajo Fin de Máster es el de una implementación de una tienda para pequeños artistas y comerciantes, con el objetivo de que sea fácilmente manejable por éstos.

Una de mis aficiones es el dibujo de ilustraciones y cómics, esto me ha llevado a tener contacto con diferentes artistas y una de las quejas generalizadas es el estado de las redes sociales, que obligan a que la mayoría de pequeños artesanos y artistas tengan que dedicarse a producir mucho contenido para éstas si quieren obtener relevancia en lugar de poder dedicarse a sus propios productos.

Asimismo, las diferentes plataformas que han ido apareciendo para ayudar a estos pequeños artistas, como gumroad, artstation o linktree, suelen tener detrás diferentes intereses que no suelen coincidir con los de estos artistas. A veces estas discrepancias son de motivo ideológico y otras por obligaciones que los interesados bien o no pueden o quieren cumplir. Por ejemplo, condiciones de trabajo de los propios trabajadores de las plataformas o su interés en tecnologías como los NFT, que abren acalorados debates.

Por otro lado, desde el punto de vista de sus clientes, buscar contenidos específicos en redes sociales, aun conociendo al autor y su trabajo, es bastante complicado. Esta complicación es mucho mayor si no recuerdas exactamente algo del trabajo del artista, imposible volver a encontrar algo que hayas visto debido a los algoritmos dirigidos de las redes sociales. Esto hace que dar visibilidad a la página del artista sea también importante.

La intención de este TFM es dar una plataforma propia y sencilla a los artistas que sirva tanto como tienda como de portfolio para ellos, de código abierto y adaptada al stack tecnológico de Node y que les dé más independencia en la venta o alquiler de sus productos, sin tener que depender de nada más allá de un hosting. El resultado final de este TFM pretende ser un software utilizable, útil, instalable y totalmente funcional.

1.2 Objetivos del trabajo

Aunque el listado de los objetivos completos se encuentra detallado en el punto 3 de esta memoria, como concepto general el objetivo de este trabajo fin de master es el de desarrollar una tienda online basada en el stack tecnológico de NodeJS con un frontend que sea atractivo para los clientes, basado en una SPA pre

generada para disfrutar de las ventajas que da a toda tienda online una optimización para buscadores y una velocidad alta de carga de los recursos.

1.3 Enfoque y método seguido

La estructura de este trabajo se realizará en 3 pasos diferentes, correspondientes en el tiempo con las tres diferentes entregas que se definen en el marco de esta asignatura, cada ellas en el plazo aproximado de un mes.

Para cada una de ellas se tomará en cuenta como si fuera un sprint SCRUM, en el que se realizarán para el código diferentes ramas siempre incluyendo una historia de usuario o funcionalidad completa e intentando, siempre, respetar git Flow como flujo de trabajo definido e integrando estas ramas tanto en backend como en frontend.

1.4 Planificación del trabajo

1.4.1 Primera entrega, 1 de marzo de 2022

- En esta primera entrega se realiza la definición del proyecto y de las diferentes etapas del que constará, así como la estructuración inicial de la memoria final, que puede estar sujeta a cambios mínimos de estructura.
- Se decide la licencia de software de la que disfrutará el proyecto, habiéndose decantado por una licencia de uso libre y código abierto de los resultados del mismo.

1.4.2 Segunda entrega, 30 de marzo de 2022

- Preparación de la estructura del sitemap, con diagramas de flujo de usuario.
- Estructura completa de base de datos a implementar en el backend nocode.
- Creación de los repositorios de código públicos.
- Creación de mockups básicos de las páginas públicas del sitio.
- Elección de una paleta de colores para la tienda, funcional y accesible tanto para modo claro como oscuro.
- Entrega de memoria actualizada del proyecto.

1.4.3 Tercera entrega, 8 de mayo de 2022

- Implementación del diseño, con modo claro y oscuro.
- Páginas públicas de eventos, productos y de información básica de la tienda (páginas básicas que se sacarán de la estructura del backend)
- Implementación de un carrito de la compra, sin ventas.
- Comunicación correcta entre las diferentes pestañas que se abran.
- Generación estática de la tienda.
- Configuración del soporte multi idioma.
- Entrega de memoria actualizada del proyecto.

- Prueba con usuarios finales para recoger el feedback sobre el uso.

1.4.4 Entrega final, 6 de junio de 2022

- Formulario de contacto.
- Comprobación de la optimización para buscadores y accesibilidad.
- Posibilidad de objetivos secundarios si el tiempo lo permite.
- Arreglo de posibles bugs.
- Finalización de la memoria del proyecto.

2. Objetivos

El objetivo principal de este trabajo consiste en el desarrollo e implementación de una tienda online para pequeños artistas utilizando tecnologías modernas, pero con facilidad de uso, utilizando como una de las líneas fundamentales, el backend, un software *no code* que pueda ser utilizado de forma sencilla y amigable por estos creadores, siempre pensando en que éstos mantengan su independencia de grandes plataformas en las que no tengan control.

Para la consecución de este objetivo será necesario cumplir, al menos, con los objetivos principales que se listan a continuación:

2.1 Principales

Objetivos clave de este TFM:

- Tienda desplegable fácilmente en cualquier hosting, por ejemplo, netlify.
- Soporte para diferentes idiomas a elegir por los autores. Se realizará la prueba en inglés y castellano.
- Listado de enlaces con todas las RRSS de los artistas.
- Posibilidad de venta online o de proximidad.
- Eventos, online u offline, en los que participará el autor, incluyendo mapa para llegar.
- Gestión completa de productos e imagen del autor, incluyendo diferentes categorías de productos.
- Estado del producto (en venta, agotado, realización bajo demanda).
- Optimización del SEO utilizando para cada página específica microdata de Schema.org
- Comunicación correcta entre las diferentes pestañas que se abran.
- Carro de la compra.
- Generación estática de la tienda para que pueda utilizarse un hosting asequible.
- Diseño con modo claro y oscuro.
- Accesibilidad correcta en todas las páginas (contraste de color, uso correcto de etiquetas HTML).
- Redespiegue bajo demanda.
- Formulario de contacto.

2.2 Secundarios

Objetivos adicionales que enriquecen el TF y que pueden sufrir variaciones.

- Gestión de envíos.
- Animaciones en la transición entre páginas.
- Redespiegue automático cuando haya cambios en los datos.
- Uso de múltiples divisas, marcando una como principal y utilizando un API de conversión para el resto de los que se quieran utilizar.
- Personalización de paletas de colores y fondos por el artista desde el backend.

3. Escenario

El objetivo principal de este TFM es la creación de un proyecto el que pequeños artistas o artesanos puedan tener una tienda online que pueda ser fácilmente manejable y que, además, les permita mantener el control, en la medida de lo posible, de sus datos e intereses.

A continuación, intentaré explicar varios de los problemas a los que los artistas se enfrentan en estos momentos y que pueden tratar de solventarse con este proyecto si se le da una continuidad.

3.1 Redes sociales

En estos momentos, las redes sociales son un escaparate básico para estos artistas y comerciantes, que se ven obligados a producir mucho contenido para tener algo de visibilidad. Los algoritmos de priorización de contenidos de estas plataformas, en especial Instagram, que había priorizado los contenidos de imagen, se ha movido actualmente más hacia el vídeo, promocionando aquellos contenidos que contienen este tipo de multimedia y no mostrando, incluso a la gente que te sigue, los contenidos basados en imágenes.

Aunque no se han dado números oficiales de cuántos artistas, artesanos o fotógrafos han abandonado la plataforma, sí que se han producido muchísimas quejas al respecto en publicaciones cuando desde Meta se anunció que se promocionarían más los contenidos de vídeo.

La idea de que tengan que bailar para su público -literalmente- únicamente para tratar de conseguir alguna venta o que su arte sea visto es muy incómodo para aquellos que han estado compartiendo y conectando en estas plataformas durante años. Se acaba convirtiendo en un trabajo a jornada completa.

De igual manera, encontrar un contenido o a un artista que hayas visto hace tiempo es bastante complicado debido a la búsqueda limitada de todas las redes sociales. Esta falta de visibilidad en la búsqueda se puede evitar teniendo una tienda o portfolio personal con un SEO aceptable. Aunque en Google es bastante complicado poder competir contra grandes tiendas en términos específicos de SEO/SEM, sí que sigue siendo igualmente útil para buscar una ilustración concreta, si esta está bien definida, de un autor concreto.

3.2 Intereses del autor frente a las grandes empresas

En muchas ocasiones, un autor debe apoyarse, de una u otra manera, en diferentes plataformas para poder realizar ventas si no tiene tienda propia. Y a estas plataformas no les interesa los ideales o forma de pensar de sus artistas, son empresas que tiene sus propios intereses, lo que es lógico, pero muchos artistas buscan una alternativa que respete los suyos propios.

Gumroad, ampliamente utilizada por artistas para hacer microcobros, ha recibido críticas en los últimos meses por dos motivos fundamentales. El primero de ellos, el anuncio de que iban a empezar a utilizar NFT, una tecnología que ha sido atacada por muchos artistas, aunque abrazada por otros, como un esquema

Ponzi y una forma de burbuja en la que no se ha respetado, en muchas ocasiones, la negativa de artistas a participar, siendo objeto de plagios y robos.

Para muchos artistas, su postura con los NFT es clara: Si deseas apoyarme, cómprame o encárgame productos. Y evitan cualquier plataforma que les pueda relacionar.

La segunda crítica a Gumroad ha sido la de no respetar el trabajo de los artistas que trabajaban dentro de la empresa. Al parecer, algunos antiguos trabajadores no llegaron a firmar ningún contrato ni se les pagó por sus trabajos artísticos y no se respetó sus derechos de autor. En una plataforma en la que la mayoría de las ventas son de productos digitales de artistas, esto supuso que muchos de ellos empezaran a buscar una alternativa que respetara sus derechos.

Y no solo por el trabajo o el respeto a las relaciones laborales. Hace unas semanas, Artstation, la plataforma más famosa, junto con DeviantArt, para que los artistas puedan compartir su trabajo digital, estuvo en el ojo del huracán tras censurar múltiples trabajos que apoyaban a Ucrania en su guerra contra Rusia. Se eliminaron y censuraron comentarios y arte relacionada con el conflicto. Esto se debe a que parte del equipo de desarrollo de nuevas funcionalidades de la empresa se encuentra en tierras rusas. Esto, como se puede suponer, ha enfadado a numerosos usuarios de la plataforma.

Como un ejemplo algo más extremo, tenemos Slideshare. Si alguna vez una persona ha dado una charla pública, es muy probable que su presentación fuera subida a Slideshare para poder compartirla fácilmente. Desde que LinkedIn vendió la compañía a Scribd, esas diapositivas se encuentran secuestradas tras un muro de pago que no existía cuando esas presentaciones fueron subidas, perdiendo todo el control.

Utilizando un proyecto como el desarrollado en este TFM se dependería de dos puntos fundamentales en la tienda personal: El hosting elegido y la pasarela de pago. Dos puntos que se pueden modificar dado el caso ya que todo el software utilizado en este proyecto es libre y se comparte de la misma manera, para que todo artista (o alguien contratado con conocimientos por ellos) pueda adaptar y mejorar la plataforma.

3.3 Control de datos

La frase *cuando el producto es gratis, el producto eres tú*, ha estado en boca de todos en los últimos tiempos en el Internet, haciendo referencia al verdadero precio que pagamos por la navegación gratuita en internet. Cuando, además, tu forma de trabajo depende de estas plataformas gratuitas estás entregando datos que pueden ser usados para, incluso, competir con el propio artista.

Un nuevo movimiento en Internet está proponiendo que, de nuevo, como en el comienzo de la www, lo que se ha venido a llamar Web 1.0, los usuarios sean los propietarios de la información que generan, almacenándola y guardándola en un sistema independiente de las plataformas donde se use, y que se puedan controlar por parte de los propios usuarios.

Tal y como ha ocurrido con Slideshare, comentado en el punto anterior, los datos de usuarios pueden ser secuestrados y ocultados en un momento dado, o cambiada la política de los mismos y que los artistas (y el resto de usuarios) puedan perder todo el trabajo realizado.

La diferencia con la web 1.0 es que, ahora, hay múltiples herramientas *no-code* que permiten a usuarios con menos conocimientos técnicos poder realizar y tratar los datos que produce y recibe. Todos los datos deben democratizarse para que los usuarios puedan usarlos en sus aplicaciones y servicios que necesiten.

Con el objetivo de democratizar los datos, las plataformas de datos abiertos se construyen para amoldarse a lo que los usuarios necesitan en cada momento, proporcionando la agilidad necesaria para mantenerse moderna y actualizada, y ayudándolos a ejecutar sus objetivos e iniciativas de forma más eficiente y rentable.

Con este proyecto se pretende, igualmente, respetar este movimiento y ayudar a los artistas a tener el control sobre sus propios datos, que no dependan, en la medida de lo posible, de terceros.

4. Contenidos

La estructura de la web que podrá visitar el cliente se compondrá, en todas las páginas, de una cabecera con su menú, cuerpo de la misma y pie de página con los enlaces a las páginas añadidas por el artista en el gestor de contenidos headless.

- La cabecera, además del menú que contiene los enlaces a todas las secciones, tendrá un carro de la compra que se podrá mostrar si existe algún producto añadido al mismo.
- El cuerpo de la página contendrá toda la información de las secciones, listados o detalles. Al tratarse de una Single Page Application, está diseñada para que lo único que se vaya modificando es el contenido de las diferentes secciones, incluso cuando se genera estática y completamente.

Todos estos contenidos son generados a través del backend Headless CMS directus, tal y como se explicará en la sección 7.1 de esta misma memoria del proyecto. La estructura de páginas y contenidos se explicará en el apartado 7.2

5. Metodología

5.1 Metodología Scrum

Para el desarrollo del proyecto vinculado a este TFM así como esta propia memoria, se ha decidido el uso de una metodología ágil, Scrum, que Ken Schwaber y Jeff Sutherland (noviembre de 2017) definieron como el marco de procesos que, desde los años 90, se utiliza para gestionar trabajo en desarrollo de productos complejos, no necesariamente vinculados al desarrollo IT.

Esta metodología busca satisfacer tanto requisitos como necesidades de los clientes, en este caso, pequeños artistas que quieren tener el control sobre sus tiendas y datos, al mismo tiempo que busca reducir la complejidad en el desarrollo de este tipo de productos. Scrum, más que una metodología en sí misma, es un conjunto de procesos y técnicas basadas en un modelo de proceso empírico que permite a los grupos autogestionarse.

Flujo de SCRUM



Ilustración 1: Flujo de trabajo SCRUM. Origen: Platzi

En Scrum, los proyectos se dividen en ciclos temporales cortos, entre 2 y 4 semanas, llamados Sprints. Cada vez que un sprint acaba, debe proporcionarse un resultado que se traduzca en un incremento del producto final entregado. Debido a la naturaleza de este propio TFM y sus entregas mensuales, era la opción más lógica en el desarrollo del proyecto, asumiendo un sprint por cada una de las entregas del proyecto.

Como herramienta de planificación de tareas se ha utilizado Trello. Al ser este un proyecto relativamente pequeño y a realizar por una única persona, se ha optado por realizar una lista de funcionalidades a realizar y marcarlas como "Por hacer", "en curso" o "finalizada". Una tarea se puede poner en curso en el mismo

momento que se empieza a trabajar efectivamente en ella, pero no se pondrá nunca una tarea como “finalizada” hasta que la rama git que contiene su código no se encuentre completamente mezclada en el repositorio principal.

Si se encuentra un bug en una rama ya mezclada, se puede optar por dos soluciones: bien reabrir la tarea volviéndola al estado de partida “por hacer” o bien abrir una nueva con el bug. En este proyecto se han usado ambas opciones. Si la tarea se había cerrado hace poco y se iba a trabajar inmediatamente en ella, se volvía a mover a “por hacer”. En el caso que fuera algo que no se pudiera abordar en ese momento, se creaba una nueva tarea. Estas tareas no eran únicamente tareas de desarrollo que implicaran modificación del código fuente de la aplicación, también se han incluido tareas de gestión para la creación de la documentación y repositorios de código, evitando que ningún paso pudiera olvidarse y, de esta manera, teniendo claro el trabajo hecho y el que quedaba por hacer.

Para poder realizar una nueva funcionalidad en la metodología Scrum es necesario definir correctamente historias de usuario. Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final. Su propósito es definir correctamente la manera en la que al cliente le ayudará esta funcionalidad y, definirla, asimismo, de forma que el encargado de desarrollarla pueda saber hasta dónde debe llegar antes de que esté completa. Estas historias de usuario no son exclusivas de Scrum, sino que son ampliamente utilizadas en otras metodologías ágiles, como, por ejemplo, Kanban.

5.2 Git Flow

Como se ha comentado en esta misma memoria, se ha utilizado Git como control de versiones ya que, actualmente, es el más utilizado del mundo. Git es un sistema distribuido, que permite que, en cada copia local del proyecto del desarrollador, haya una copia completa del mismo, que sirve, de esta manera, como copia de seguridad e historial completo del proyecto, al contrario de lo que sucedía con sistemas de control anteriores como Subversion o CVS. Al tener tan alta adopción por parte de las empresas de desarrollo, se ha convertido en un estándar de facto.

Pese a que este desarrollo se ha realizado por una única persona, se ha pensado que la mejor forma de trabajar en el mismo era utilizar Git Flow, al igual que se utilizaría en un proyecto grande desarrollado por múltiples personas y/o equipos.

Git Flow es un flujo de trabajo sobre Git diseñado por Vincent Driessen que consiste en un conjunto de buenas prácticas y metodologías para el buen uso de las ramas de desarrollo en los proyectos, que deben ser respetadas por cada uno de los miembros del equipo de desarrollo. Esta metodología de trabajo se basa en los siguientes principios:

5.2.1 Las ramas main y develop son intocables

Estas ramas deben ser de sólo lectura para todos los desarrolladores del equipo. **Main** (anteriormente *master*) corresponde al código que está ahora mismo en producción o, en el caso de los proyectos de software libre como el que se ha realizado en este TFM, aquel código que puede ser utilizado sin problemas en un entorno de producción una vez descargado y configurado.

La rama **develop**, por otra parte, tiene el código que ha sido finalizado, con sus modificaciones, corrección de bugs y nuevas características que se incluirán en la siguiente salida a producción (es decir, la próxima mezcla a main)

Es importante que estas ramas permanezcan siempre de sólo lectura, para que todos los cambios se realicen sobre ramas independientes y sea fácil tanto añadirlas como descartarlas si, por el motivo que sea, deben eliminarse del repositorio.

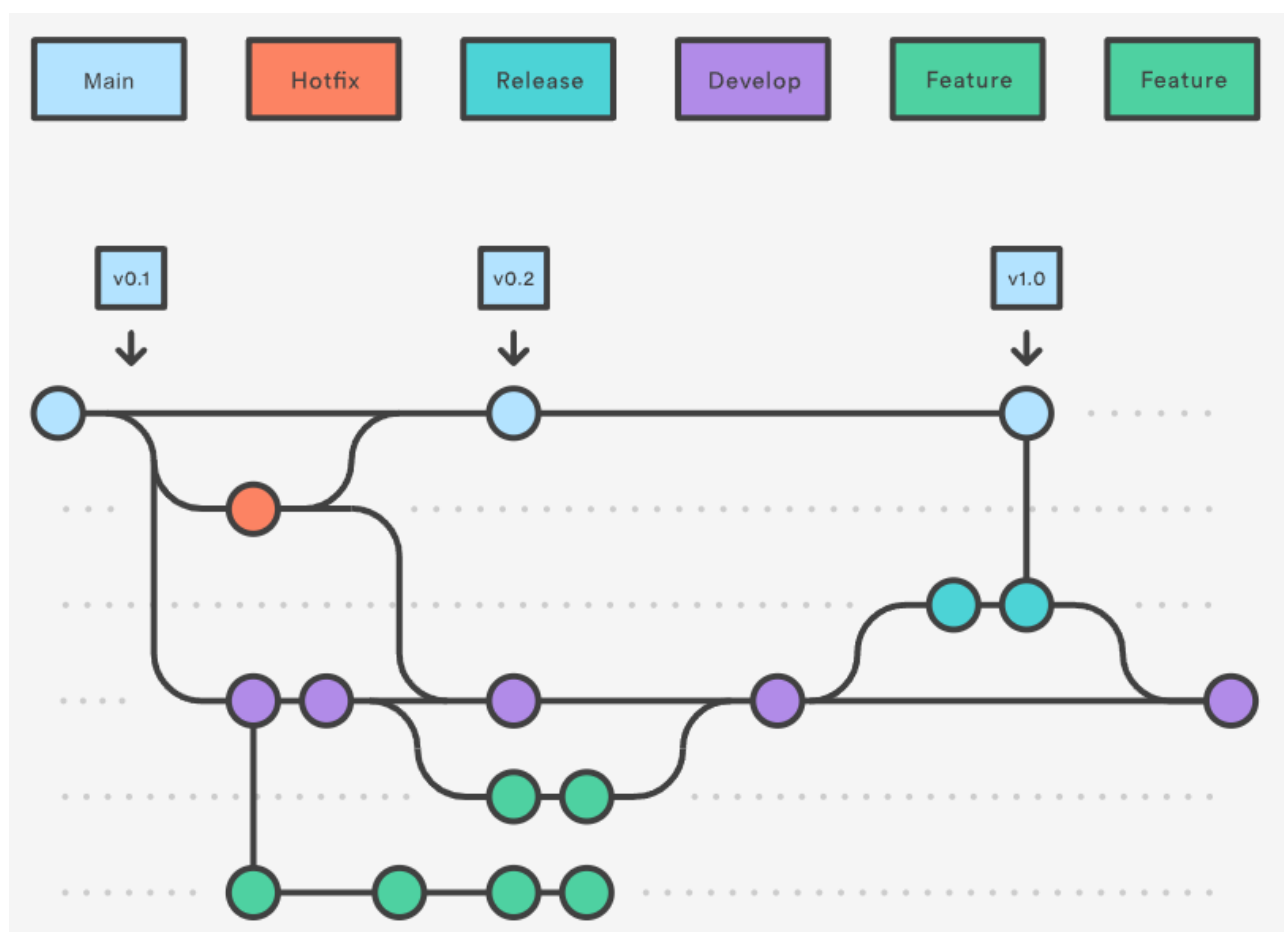


Ilustración 2: Uso de Git Flow. Atlassian (2022)

El paso de modificaciones desde la rama **develop** a la rama **main** se llama *release* (liberación del código). Todos estos nombres de ramas pueden estar sujetos a modificaciones dependiendo de cada empresa y equipo de desarrollo, para adaptarlos a su realidad.

En el caso de este desarrollo, se ha estado realizando todo el trabajo sobre ramas desde y hacia **main** porque no ha habido una versión en producción en estos momentos. Una vez se ha entregado y está preparado para ser clonado y usado, se ha creado una rama **develop** a partir de la cual se añadirán las nuevas características y arreglo de bugs sobre la aplicación manteniendo la rama main como la opción estable que pueda ser utilizado por cualquier artista que lo desee, con o sin ayuda de alguien con más conocimientos en IT.

5.2.1 Todo el trabajo se realiza mediante ramas

Todo cambio sobre el repositorio general, como se ha comentado en el punto anterior, no se puede realizar sobre las llamadas ramas estables, sino que se deberá crear una nueva rama para cada uno de los cambios que se desee hacer, sea un cambio para una nueva funcionalidad o para un arreglo de un posible error en el código.

Al utilizar una rama independiente para estos desarrollos, sean cortos de una o varias horas o grandes desarrollos de un sprint completo, éstos están más controlados y es más sencillo revisar todo lo modificado antes de mezclar esta nueva modificación de código contra la rama de desarrollo (**develop**) a través de una petición de mezcla (la nomenclatura de estas peticiones suele cambiar, las más comunes son *Pull Request* y *Merge Request*)

Los tres tipos de ramas que considera Git Flow son los siguientes:

- Rama de funcionalidad o *feature*: Son aquellas ramas que contienen el desarrollo completo de una modificación sobre el código fuente. Pueden contener el arreglo de un bug si no es urgente su arreglo. Son ramas que se mezclan, una vez finalizadas, contra develop.
- Rama de arreglo de bug urgente o *hotfix*: Si hay que realizar un arreglo urgente en el código que se está ejecutando en producción y, de alguna manera, saltarse por esta urgencia, el flujo normal de mezcla, se realizará una rama de este tipo. Estas ramas se mezclarían dos veces: una contra master, para arreglar el fallo urgente en producción y, una vez arreglado, se mezclaría desde master a develop para llevar este cambio (el orden inverso que se realizaría en un flujo normal de desarrollo).
- Rama de *release*: Una vez las nuevas funcionalidades y desarrollos se han completado, se realiza una rama que manda todas estas nuevas características a producción. Suelen ser paso programados con el consenso del equipo y, en Scrum, suelen coincidir con el final de cada uno de los Sprints antes de realizar la reunión de retrospectiva y fin de sprint, enseñando las nuevas funcionalidades conseguidas.

6. Arquitectura de la aplicación/sistema/servicio

Este proyecto se puede dividir en dos líneas muy diferenciadas, pero que se comunicarán entre sí a través de APIs, el backend y el frontend. A continuación, se detallarán las diferentes tecnologías con las que se construirán cada una de ellas.

6.1 Backend

La estructura del backend se desarrollará sobre la plataforma Directus.io, que es un headless CMS, esto es, un gestor de contenidos separado del frontend y no ligado a ningún esquema de vistas específico, pudiendo conectarse a cualquier aplicación SPA, en este caso, a través de API Rest y GraphQL.

Este CMS comenzó estando desarrollado en PHP, pero, para su versión 9 se ha realizado una migración completa a Node, soportando asimismo TypeScript.

Ventajas del uso de Directus como Headless CMS:

- Sobre todo, y por lo que se ha elegido esta solución, es que es de software libre y código abierto, se puede instalar en hosting propio o elegir una de sus soluciones de pago.
- Existe una versión community disponible para pequeños proyectos como el que aquí se desarrolla. No obstante, se opta por instalar en servidor propio para poder comprobar la viabilidad de una elección de usuario que desee conservar todos sus datos disponibles.
- Está basado en una plataforma potente, pero basada en la experiencia de usuario y con una amplia documentación, tanto para la parte de administración como para un usuario final que no tenga tantos conocimientos.
- Se puede instalar bien sobre una base de datos existente o comenzar desde cero.
- Seguridad en la autenticación, soporta protocolos conocidos y soportados como OAuth2, OpenID, 2FA, SCIM, static API tokens, entre otros.

En el caso de este TFM, Directus se instala y configura sobre una Base de datos SQL como es MariaDB.

6.1.1 Estructura final de la Base de datos.

Opciones

Funciona como un singleton. En este modelo de datos se guardarán aquellas variables de configuración para la generación estática de la tienda, como la definición de los colores, las tipografías, títulos o descripciones.

Social Media

Para evitar tener que depender de otras plataformas como linktree, en este modelo de datos se guardarán todos los enlaces de las redes sociales del artista, que podrá enlazar desde sus perfiles. Se añade una

opción para marcar si tiene o no contenido para adultos para avisar antes de navegar a la plataforma elegida.

Idiomas

Para ayudar con las traducciones de los demás modelos de datos que se definirán a continuación, se añaden los idiomas que soporta la plataforma. En este primer desarrollo, se opta por el inglés y el castellano.

Etiquetas y categorías

Para facilitar la navegación entre los diferentes productos y contenido se opta por añadir tanto etiquetas como categorías para los mismos.

Posts

Separados en páginas estáticas y blog, permiten añadir contenido que sea fácilmente indexable por buscadores, lo que da mayor visibilidad a la página y la tienda.

Eventos

Un artista suele tener que acudir a promociones, bien sean sesiones de firmas, bien sean convenciones. Es una forma de dar visibilidad a estos eventos y se pueda saber dónde está en todo momento sin tener que intentar encontrarlo en redes sociales.

Productos.

Finalmente, la parte importante de la tienda, donde se podrán añadir los productos con su precio para realizar los pedidos a través de Stripe. Estarán relacionados con los posts para que se pueda, si hay una serie de artículos que consiguen más visibilidad, promocionar en ellos los artículos que se deseen.

En la siguiente imagen se puede ver una estructura simplificada de la Base de datos, tal y como se hubiera tenido que definir si no se usara una herramienta que permite abstraerse levemente de ella.

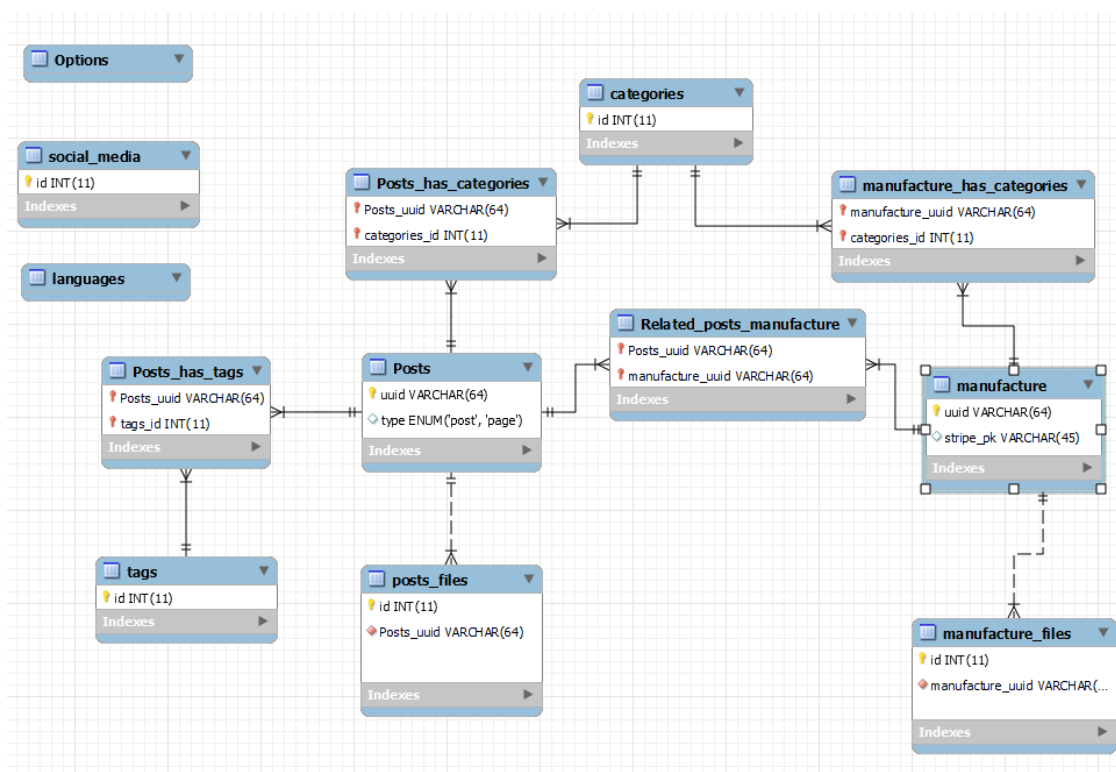


Ilustración 3: Estructura de BBDD del proyecto

6.1.2 Ventajas de la implementación mediante Directus

En el caso de haber optado por una solución de backend que implicara un desarrollo completo de una solución en lugar de utilizar una herramienta no-code como directus es que, eligiendo esta segunda opción, un usuario con algo de conocimiento puede añadir o modificar la estructura de la tienda utilizando mucho menos tiempo.

Con la infraestructura que la plataforma nos provee se pueden añadir o modificar campos de forma muy rápida, añadiendo una capa de abstracción que permite poder pivotar rápidamente entre diferentes elecciones para la plataforma.

6.2 Frontend

El frontend se generará como una SPA (Single Page Application), pero generada parcialmente como un sitio estático para beneficiarnos del SEO en buscadores.

Como stack tecnológico se valoró utilizar la última versión del framework Nuxt, Nuxt 3, que actualmente se encuentra en sus últimas fases de la beta pública con la salida de una versión RC (*Release Candidate*) ya muy similar a la versión finalizada, pero, tras el comienzo de la implementación de este proyecto, ésta no estaba todavía preparada para un uso intensivo, precisamente, en la generación estática de sitios con lo que, finalmente, se utilizará la versión 2, ampliamente utilizada y testeada por la comunidad.

Nuxt es totalmente modular, con el uso tanto de módulos ya implementados, como del propio uso de librerías y *plugins* generales de VueJS, por lo que podemos añadir funcionalidades de manera sencilla. Al igual que Directus, es de código abierto y dispone de una muy activa, en la cual podemos consultar o buscar información sobre problemas que hayamos tenido en nuestro código, pero, a diferencia de Angular y ReactJS no hay una gran empresa detrás, sólo el trabajo de la propia comunidad.

Ventajas del uso de NuxtJS:

- La estructura de carpetas ya viene definida y es estandarizada, lo que supone una ventaja ya que cualquier desarrollador que aterrice en el proyecto tendrá una estructura conocida y documentada.
- El enrutamiento de NuxtJS es realmente sencillo si se compara con el uso de VueJS como librería.
- La configuración de un proyecto recién creado ya está preparada para salir a producción.
- El framework ya provee tanto de SSR como de generación estática de sitios.
- La documentación es amplia y se encuentra actualizada.
- Nos permite realizar Code Splitting, de forma que podemos dividir el código en múltiples partes con el propósito de que la aplicación cargue de manera más rápida, ya que solo se cargan los componentes que cada página necesita.

Además, para el definir el estilo final de la aplicación de este TF se utilizará, para el CSS, una mezcla entre el framework TailwindCSS pero sobre metodología BEM e intentando suplir los puntos flojos con CSS puro (la gestión del modo oscuro, por ejemplo, y el uso de variables CSS para la definición de todos los colores que se utilicen, para hacerlo escalable).

7. Plataforma de desarrollo

7.1 NGrok

Para poder conectar desde una red externa al directus que se ha instalado en una nube propia y protegida por firewall, se ha necesitado hacer uso de ngrok. Esta herramienta en la nube nos permite exponer, de forma muy sencilla, servicios locales o protegidos detrás de un firewall, como es el caso, una URL pública de forma segura. NGrok trabaja a través de distintos cortafuegos y NATs configuradas, por lo que se puede utilizar desde cualquier lugar.

Gracias a esta plataforma se ha podido exponer este desarrollo y configuración local a Internet, permitiendo que la web no estática que se ha desplegado en un hosting convencional tenga comunicación con el backend para recibir los datos.

Esta solución no hubiese sido necesaria si se hubiera optado por la solución community de directus, pero al haberse optado por la opción que más control pudiera dar sobre sus datos a los artistas, se ha necesitado utilizar ngrok.

Aunque tal y como se ha explicado en el apartado del presupuesto de esta memoria ngrok ha tenido un coste por el desarrollo, para un artista que empiece la solución básica gratuita podría resultar más que suficiente.

7.2 Hosting

7.2.1 *Hosting compartido*

La solución más común para las webs en otras opciones disponibles de gestores de contenidos es un hosting compartido donde se instalan la aplicación. Para la opción no completamente estática de la web se ha optado por esta solución de hosting,

7.2.2 *Netlify*

Netlify, tal y como se nos ha comentado durante todo este máster, es una plataforma que está específicamente pensada para automatizar proyectos web estáticos, por lo que era la solución, tal vez obvia, para el despliegue de la versión totalmente estática de este desarrollo.

8. Planificación

A continuación, y en detalle, se desarrollarán todas las diferentes tareas y subtareas que se han realizado a lo largo del desarrollo de todo este TFM, y que se vieron brevemente en el punto 1.4 de esta misma memoria.

Cada uno de los sprints en los que se ha dividido el desarrollo se corresponden a cada una de las pruebas de evaluación continua, haciendo coincidir comienzo y fin de estos sprints con las fechas de las entregas, por simplicidad en la gestión del tiempo.

Durante el transcurso de la primera entrega se realizó la definición del proyecto, así como cada una de las características que se entregarían al final de cada sprint. Se decide, asimismo, la licencia de software con la que se desarrollará el proyecto, una licencia Creative Commons Attribution-ShareAlike 4.0 internacional. Con estos dos puntos se llega a la entrega del 1 de marzo de 2022.

La segunda entrega, que llega hasta el 30 de marzo de 2022, consiste en la configuración de toda la estructura de datos correspondiente al headless backend directus, formando todo el contenido que finalmente hidratará el contenido de la web, sea el despliegue completamente estático o generada automáticamente en el momento de la carga conectándose a directus. También se realizan los mockups básicos necesarios antes de comenzar el desarrollo del frontend, así como elección de paletas de colores básicas, claras y oscuras, que se añaden por defecto al backend, aunque podrán ser modificada al gusto por cada artista.

La tercera entrega ha sido, en su mayoría, para el desarrollo de la parte frontend de la aplicación, añadiendo una conexión entre ambas partes que permite hacer una copia estática de todo el contenido. Estas páginas incluyen páginas públicas, eventos, productos y páginas obligatorias que debe llevar cada tienda online. Los detalles de la implementación de estas páginas se encuentran en sus apartados correspondientes de esta memoria, por lo que no es necesario extender este punto al respecto. Este proyecto, cuya versión sería ya Release Candidate, se entrega a tiempo para el 8 de mayo de 2022.

El cuarto y último sprint, correspondiente con la entrega final del 6 de junio de 2022, se utiliza en su mayoría para terminar de forma correcta esta memoria y pulir aspectos que quedaban pendientes en labores de programación de entregas anteriores.

Una tarea común a las entregas de la PEC 2 y 3 ha sido el testeo de los requisitos mientras se iban desarrollando y que se han publicado en los tests de aceptación del punto 14 de esta memoria

Resumiendo, esta aplicación ha ocupado un total de 35 jornadas de trabajo, 7 semanas, para la configuración correcta del backend y desarrollo de un frontend que se comunicara con él y que fuera

relevante para el artista que quiera empezar su web, es decir, para la parte de desarrollo del proyecto. Para la escritura de esta memoria habría que añadir, además, un total de 11 jornadas de trabajo adicionales, haciendo un total del proyecto de 375 horas de trabajo.

Este proyecto, por la propia naturaleza del mismo, ha sido realizado por una única persona, pero podría haber sido paralelizado de forma que, tras una toma de requisitos correspondiente a la primera entrega, la parte backend y frontend, e incluso la documentación misma, podrían haberse realizado en paralelo por personas o equipos distintos, por lo que, si bien las jornadas de trabajo no se hubieran reducido (pudiendo, incluso, aumentar por el tiempo gastado en las labores de comunicación entre estas personas / equipos) el tiempo de entrega sí hubiera sido menor.

En la tabla 1 se puede ver un esquema de desarrollo en forma de diagrama de Gantt con estas tareas paralelizadas y separadas por semanas. Por consistencia con este mismo punto de la documentación, las semanas se definen tal y como se ha desarrollado el proyecto, sin la paralelización de tiempos comentada en el párrafo anterior.

| | PEC 1 | PEC 2 | PEC 3 | Entrega final |
|--|-------|-------|-------|---------------|
| Documentación | | | | |
| Toma de requisitos | | | | |
| Configuración y adaptación de directus | | | | |
| Desarrollo del frontend | | | | |
| Tests plataforma | | | | |

Tabla 1: Diagrama de Gantt del proyecto

9. Proceso de trabajo/desarrollo

9.1 Proceso de trabajo sobre el headless CMS.

9.1.1 *Eligiendo la versión community*

Finalmente, tal y como se había anunciado, directus provee de una versión gratuita, llamada “Community Cloud” que permite probar la plataforma sin muchos límites. Por ello, se probó primero a utilizarla, probando si era suficiente para este proyecto.

Una de las limitaciones que tiene esta versión es que apaga la máquina que ofrece después de tres días sin uso, lo que, para pequeños artistas, no afectaría, ya que se podría volver a arrancar en el momento en el que se necesite la generación del sitio, pero afectaría a la forma en la que un artista podría enseñar si un producto o no está agotado, ya que toda esa generación de datos debería ser en el momento de la generación estática del sitio, y no modificable sin regenerar, lo que puede ser incómodo para alguien sin muchos conocimientos.

Además, tal y como se comentó en el punto 1 de esta memoria, los artistas volverían a depender de la infraestructura de un tercero para gestionar sus datos, lo que conllevaría el cambiar un proveedor por otro. Aunque se empezaron a añadir la estructura de datos sobre esta plataforma, se decidió volver a la versión autogestionada.

9.1.2 *Eligiendo finalmente una versión self hosted*

Instalando dependencias.

Estos pasos no serían necesarios si se eligiera un hosting profesional, en el que ya nos proveerían con estas dependencias, por lo que podríamos pasar directamente al siguiente punto.

En este caso, que seremos nuestro propio proveedor de servicios, el primer paso para desarrollar sobre directus y poder ejecutarlo en una versión autogestionada es instalar las dependencias necesarias para que pueda ejecutarse correctamente, en este caso, node y una base de datos MariaDB.

Se ha instalado node en el servidor a través de un comando apt-get, *apt-get install nodejs*, que ha instalado la versión de node 16 (LTS)

Para la instalación de la base de datos se ha optado por realizarlo mediante una imagen Docker, para tenerla actualizada de forma fácil y segura, ya que, al contrario que pasa con NodeJS, no se necesita como tal en el desarrollo, sino que es únicamente el lugar donde se guardará la información. Docker provee una capa muy ligera de virtualización para ejecutar servicios o procesos aislados. Para conseguirlo, Docker empaqueta el servicio y todas las librerías que pueda necesitar en un contenedor virtual que se puede ejecutar en cualquier servidor, evitando tener que ensuciar el sistema operativo anfitrión.

Con estas dos dependencias, ya podemos pasar a instalar directus.

Instalando directus.

Como se ha comentado, directus está basado, a partir de su versión 9, en JavaScript, habiéndose migrado desde PHP. Para instalar un servidor headless directus sobre nuestro repositorio es muy sencillo. En el directorio que queramos instalarlo únicamente tenemos que ejecutar un comando npm init, en este caso:

```
npm init directus-project nuxt-directus-shop
```

Una vez se ejecuta este comando hay que seguir las indicaciones, donde se nos pregunta, en orden, los siguientes datos para la configuración:

1. Tipo de base de datos que vamos a utilizar (en este caso, MariaDB)
2. Datos de conexión a la base de datos (servidor, usuario y contraseña)
3. Usuario y contraseña para el usuario de administración
4. Si deseamos usar HTTPS o no.

Una vez terminado, se generan tanto los datos necesarios en la base de datos como un fichero de proyecto de NodeJS (*package.json*) y se instalan las dependencias necesarias en el directorio *node_modules*.

Arrancando directus.

Una vez se ha instalado todo, se puede arrancar el servidor con *npx directus start*. Para que sea más sencillo el arranque de esta aplicación se ha añadido también un comando dentro del fichero *package.json* para poder ejecutarlo de forma más estandarizada con otros proyectos JavaScript, mediante *npm run start*.

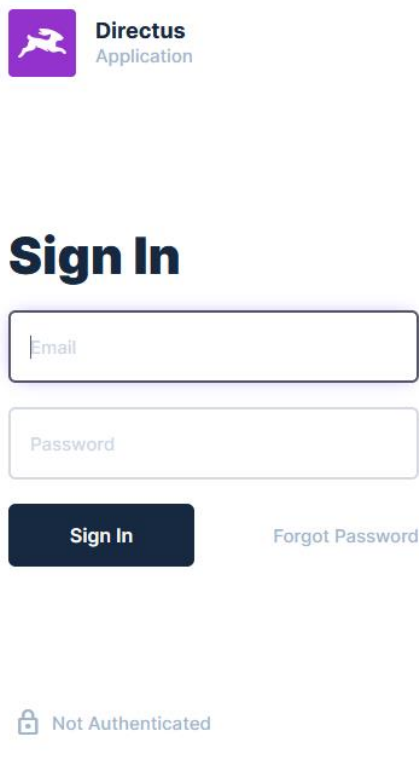
También se ha añadido un comando adicional. Debido a que el desarrollo de directus está muy activo, es bastante común que se tenga que actualizar a una versión posterior y sea necesario ejecutar un comando adicional antes de poder acceder a la aplicación. Similar a Laravel, es necesario realizar migraciones de los datos cuando actualizamos directus. El comando es *npx directus database migrate:latest*, e, igualmente, para que sea más sencillo, se ha añadido en *package.json* como *npm run update*.

Accediendo a la aplicación

Con el servidor en funcionamiento, ya se puede acceder al proyecto Directus y empezar a usarlo de forma normal.

Si no se ha cambiado nada de la configuración por defecto en el fichero de variables de entorno *.env* se arrancará en el puerto 8055, lo que significa que podemos navegar a <http://localhost:8055> e iniciar sesión utilizando las credenciales de administrador que se configuraron en el primer paso.


Una vez accedamos, podemos ver una pantalla como esta:



Directus
Application

Sign In

Sign In [Forgot Password](#)

 Not Authenticated

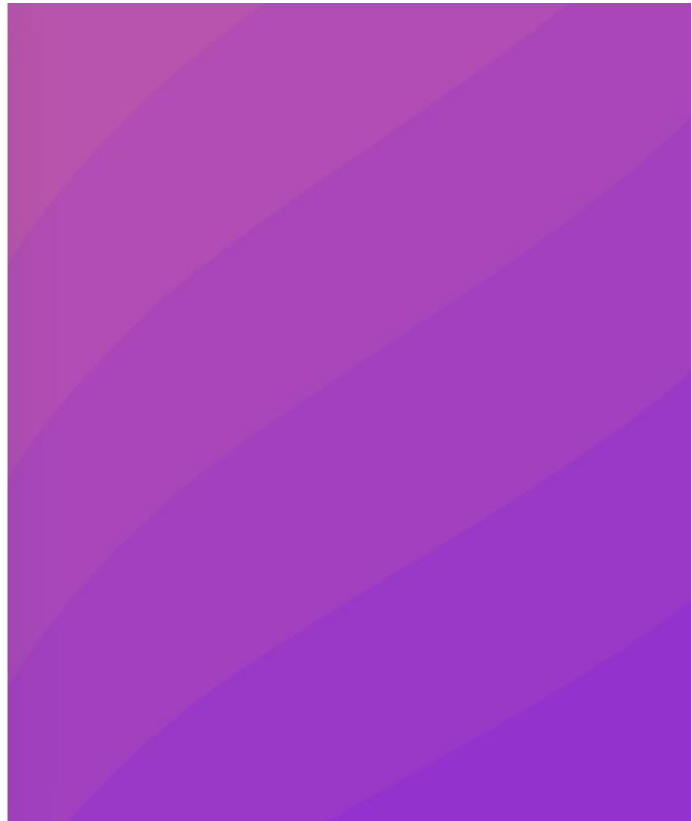


Ilustración 4: Página de autenticación del backend directus

Creando los modelos de datos.

Dentro de directus se llama modelo de datos a lo que tradicionalmente serían las tablas de una base de datos, pero se crean todas con un editor WYSIWYG en el que se puede elegir, modelo a modelo y atributo a atributo, cuál es el tipo y las relaciones de una manera muy visual.

Tras añadir todos los modelos de datos, la estructura que resulta es la siguiente, donde podemos observar tanto los modelos principales, que se pueden agrupar en carpetas, como modelos que, pese a haber sido creados por el administrador, no se mostrarán como tales a la hora de añadir los datos, ya que se incluyen en los propios formularios de datos.

En este caso, son las categorías relacionadas, los ficheros que se vinculan a posts y productos y las traducciones de los campos que se pueden rellenar en múltiples idiomas. Resumiendo, todas aquellas que, en un esquema tradicional de entidad relación, corresponderían a tablas intermedias de n:m (traducciones, categorías y etiquetas) o de 1:n (en el caso de los ficheros).

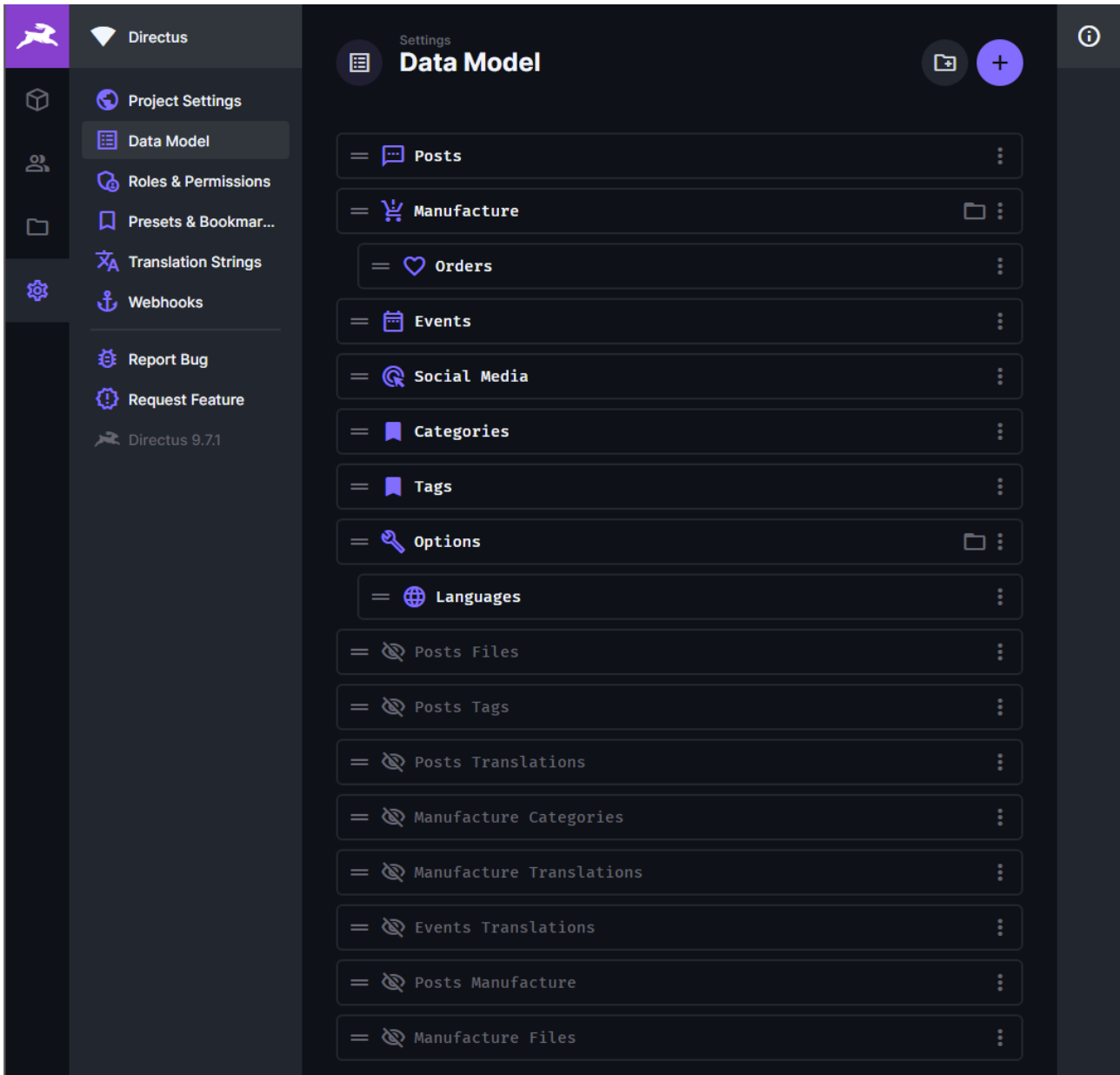


Ilustración 5: Listado de estructura y modelo de datos configurado en el backend

Aplicando extensiones a los modelos de datos

Directus, tanto en su versión community como en la autogestionada, permite añadir librerías y extensiones propias que puedan aumentar la funcionalidad existente en la misma. En el caso de esta aplicación se ha optado por añadir dos extensiones similares, que actúan en modo de filtro para los modelos de datos de posts/páginas y productos (manufacture).

Cuando un usuario no sepa exactamente qué le interesa como URL para este tipo de modelo de datos y guarda con ellas vacías, se autogenerará una URL única basada en el título en la traducción del idioma por defecto y, cuando hablamos sobre posts/páginas, en el año en el que se publica. Se ha utilizado un método similar al que Wordpress utiliza en la creación de este tipo de datos, que ayuda al usuario inexperto. Por supuesto, si se desea añadir una url personalizada, siempre que sea única, se dejará libertad al usuario.

9.2 Proceso de trabajo en Frontend

El primer paso que se ha realizado para crear la estructura de la SPA que corresponderá al front de la página final del artista que se nutra de los datos almacenados en el CMS headless explicado en el punto anterior es la creación del proyecto Nuxt. Como se ha comentado anteriormente en esta misma memoria, se ha optado finalmente por la utilización de la versión 2 de este framework VueJS de software libre ante la falta de estabilidad de su versión 3 en estos momentos.

Con el comando `npx create-nuxt-app` se puede crear un proyecto vacío, en el que, siguiendo los pasos, se puede elegir qué se añade al comenzar el proyecto. En el caso de este TFM, se han elegido y configurado las siguientes opciones en el proceso de creación:

- **Nombre del proyecto** (importante, no debe contener espacios en blanco, ya que se utilizará como nombre además en el fichero `package.json`)
- Como lenguaje, **JavaScript** en lugar de TypeScript. Al tener que usarse Vue 2 sobre Nuxt 2 como framework principal, el soporte de éstos para TypeScript no es completo, así que se ha optado por este lenguaje.
- Como UI Framework, **TailwindCSS**
- En la elección de módulos, se ha seleccionado **Axios** únicamente
- Como validador de código, ESLint, con configuración por defecto.
- Sin framework de Tests.
- Como modo de renderizado, SSR/SSG para poder generar tiendas totalmente estáticas a elección del artista que vaya a hacer uso.
- Se desplegará en hosting estático.
- Sin herramientas de desarrollo extras.
- Sin integración continua.
- Con Git como versionado.

Es importante destacar que, si se está desarrollando sobre Windows y se tiene instalada el interfaz de comandos de Git, éste se puede utilizar para todo el desarrollo del proyecto excepto para la creación del propio proyecto, que será necesario el uso del propio interfaz de comandos CMD de Windows, para poder hacer uso del soporte de teclado que nos permite navegar entre todas las opciones que el cliente de comandos nos da.

Una vez creado el proyecto, cuando necesitemos desarrollar cualquier funcionalidad únicamente tendremos que escribir `npm run dev` para empezar. Se ha añadido, asimismo, un alias para este comando, `serve`, para aquellos más familiarizados con el desarrollo sobre VueJS puedan utilizarlo, ya que se usa este comando en lugar de `dev`.

9.2.1 Comunicación entre backend y frontend

El primer paso que se ha seguido en el desarrollo, tras hacer el downgrade de la versión 3 a la versión 2 del framework, fue la comunicación entre directus como backend para poder nutrir de contenido a la SPA e hidratar de esta forma todas las páginas que se creen a partir de las colecciones que se han creado en el backend.

Esta comunicación se ha realizado en base a dos líneas diferenciadas: Comunicación directa con el servidor headless que el dueño de la tienda quiera utilizar (community o autogestionado) en cada llamada que se realice en la web, simulando un entorno tradicional como podría ser un CMS como Wordpress o bien una pre generación estática de contenidos que puedan ser usados de forma totalmente estática sin necesidad de mantener el servidor backend de directus en funcionamiento.

Esta sería una funcionalidad importante ya que, recordemos, la versión community se apaga tras un tiempo sin usarse por lo que no siempre estará disponible para hacer una llamada para recibir datos. En tiendas pequeñas o que se han empezado hace poco y están comenzando puede llegar a ocurrir, por lo que es necesario tener toda la información disponible para mostrar al posible cliente.

En ambos casos sí que es reseñable que las imágenes se generarán, en todo caso, de forma estática y se guardarán en un directorio en disco. Esto es debido a que el tiempo de respuesta de un servidor, ya sea apache o nginx, para ficheros estáticos, siempre será mucho menor que el de una llamada a backend, mejorando de esta manera el tiempo de respuesta de la tienda, que también influye en el SEO de la misma.

Gracias a la potencia de directus, se pueden generar de forma dinámica todas estas imágenes en múltiples formatos, tamaños y estructuras, que se configurarán, para esta comunicación, en el fichero `.directusshop.config.js`, en el que se pueden definir los nombres asociados a cada tipo de imagen que se descargará para cada uno de los recursos multimedia, dónde se guardarán y con qué calidad. Estas imágenes se descargarán tanto en JPG como en el nuevo formato de imágenes WebP. Como este último formato todavía no está totalmente soportado por todos los navegadores y por cuestiones de retrocompatibilidad, se entregará al cliente la versión que su navegador soporte.

Para generar estas imágenes de forma sencilla se ha añadido un nuevo comando a npm, llamado `shop:images`

Toda la información de las imágenes se almacena según su id interno en directus en carpetas diferenciadas y, dentro de éstas, el nombre del tamaño que se utilizará. Así vienen igualmente vinculadas con las colecciones de backend.

Igualmente, para generar el contenido estático, en este fichero de configuración se podrán añadir las colecciones que se quieren guardar en disco, en qué directorio y qué datos para cada una de ellas. Si en las

colecciones hay un campo url se creará, asimismo, un fichero que guarde los datos de sus detalles. Toda esta información se guarda en ficheros JSON generados a partir del API del backend para que sean fácilmente consumibles por una llamada asíncrona dentro de páginas y componentes.

Estas colecciones incluyen, como no podría ser de otra manera, las preferencias que el administrador del sitio haya usado para colores y tipografías, que se incluyen en cada página.

Estas colecciones se generarían estáticamente mediante el comando `shop:static`. El que se utilice una u otra forma de generación y uso de estos ficheros estáticos se controla a través de una variable de entorno, para que pueda ser modificada en tiempo de transpilación si se utiliza un sistema de integración y despliegue continuos.

9.2.2 Plugins utilizados en el desarrollo

Para la realización del proyecto se han utilizado varios plugins y módulos ya desarrollados por la comunidad VueJS, que han podido ser utilizados para el funcionamiento correcto de la tienda:

Módulos Nuxt:

Axios:

Módulo que encapsula la librería de llamada AJAX Axios, que permite añadir más funcionalidades a las llamadas estándar, como interceptores de cada llamada para poder añadir cabeceras personalizadas para el backend. En estos momentos del proyecto no se le da todo el uso que su potencial permite, pero se ha añadido para poder aumentar las funcionalidades del frontend de la tienda de forma más cómoda.

Internacionalización i18n:

Uno de los requisitos básicos del proyecto debía ser que funcionara en más de un idioma, ya que cualquier tienda actualmente se enfrenta a un mercado totalmente global en el que los usuarios pueden ser de cualquier parte del mundo, especialmente si las ventas se realizan con productos digitales. Por defecto se han añadido dos idiomas, inglés y castellano, pero sería fácilmente ampliable a más idiomas añadiendo una configuración en `nuxt.config.js` y su fichero de idiomas correspondiente dentro de la carpeta locales.

Sitemap:

Este módulo de Nuxt genera automáticamente el `sitemap.xml` dinámico para el proyecto, que ayudará al posicionamiento en buscadores de la tienda. Estos sitemaps pueden ser consumidos, por ejemplo, por la herramienta de Google para hacer que sea indexado con mayor rapidez.

Vue Social Sharing:

Crea un componente que permite personalizar el estilo y marcado para cada enlace para compartir en redes sociales que se quiera añadir. Se ha encapsulado dentro de otro componente que se explicará en su

sección correspondiente de esta misma memoria. Este plugin entrega un componente sin renderizado y con el mínimo de HTML y CSS necesario para que pueda ser personalizado.

Plugins Vue:

Vue click outside:

Directiva que permite que, si se clica fuera de un elemento determinado (por ejemplo, modales, selectores de fecha o similares) se lance un evento también sobre el componente que tiene la directiva para modificar su estado interno.

Vuex Persisted State

Para evitar la pérdida de información en la recarga de la página, pensando en que si un usuario navega fuera por error no pierda sus preferencias o el estado de su carrito de la compra. Se guarda en el localStorage del navegador, siempre teniendo en cuenta que no se debe, por seguridad, guardar información que pueda ser usada como un vector de ataque sobre el usuario.

Vuex Shared mutations:

Este plugin se utiliza para que se pueda compartir información entre diferentes ventanas y pestañas del navegador. Si un cliente añade en una pestaña un elemento en el carro de la compra, sería muy incómodo que, al cambiar de pestaña, este elemento no estuviera en el carro (cosa que aún ocurre en muchas tiendas online) por lo que se ha añadido y configurado este plugin de Vue para que esto no ocurra.

Vue Stripe: Se explicará ampliamente en la sección de la forma de pago.

9.2.3 Plugins desarrollados específicamente para el proyecto:

Plugin directus API:

Se ha realizado este plugin para encapsular todas las llamadas que se hagan a la información, bien de forma estática bien al propio directus como backend. Inyectando como función cada una de las llamadas a las colecciones (bien para el listado general, bien para los detalles) comprueba si es una llamada estática (con lo que devuelve el contenido del fichero json correspondiente) o es una posible llamada asíncrona, con lo que devuelve el resultado de la promesa de la llamada correspondiente de Axios con un async/await.

¿Por qué encapsularlo en un plugin? Si el artista desea conectar la tienda a cualquier otro CMS que pudiera querer utilizar (por ejemplo, un WordPress propio) únicamente tendría que pedir que le desarrollaran un plugin que devolviera funciones con el mismo nombre y resultados con el mismo formato de objeto, haciendo que siguiera funcionando toda la tienda igualmente, pero con una nueva fuente de datos compatible.

La idea principal de este plugin ha sido añadir una capa de abstracción para la recogida de los datos, evitando que se dependa excesivamente de un backend determinado que obligue a rehacer todo el proyecto si, por el motivo que sea, deja de utilizarse, dando al artista un mayor control sobre su tienda y sus datos.

9.2.4 Páginas y templates

Las páginas de Nuxt son componentes especiales, con acceso a funciones específicas añadidas para este tipo de componentes, que se guardan dentro del directorio pages. Al guardarse dentro de esta carpeta, se autogenera una estructura de rutas de la aplicación, de forma transparente y sin necesidad de configurar VueRouter. Además, nombrando el componente como `_slug`, éste tomará todas las páginas de esa ruta que no hayan sido definidas específicamente. Para este proyecto, se han definido las siguientes rutas:

Dentro del directorio raíz:

- `Index`, que muestra el artículo (post) cuya ruta es `/`
- `Linktree`, con la lista de redes sociales del artista
- `Cart`, con el listado del carrito de la compra de la tienda
- `Checkout-error` y `checkout-sucess`, para los estados tras la compra de los productos. Si se ha cancelado o ha habido error y si ha ido bien, respectivamente.
- `Posts`, con el listado completo de artículos publicados separados por año. Si recibe como parámetro la categoría de filtrado, sólo muestra los que coinciden con ésta.
- `Contact`, que incluye el formulario de contacto que se mostrará si la web no es totalmente estática, esto es, si tiene conexión con un `directus` levantado en un hosting accesible públicamente.
- `_slug`, que tratará el detalle de cada uno de los artículos y páginas, todas ellas con contenido de `microdata` para facilitar el SEO de la misma.

Directorio products:

- `Index`, que muestra un listado de todos los productos.
- `_slug`, que mostrará el detalle de los artículos, pudiendo añadirlos al carrito. Especialmente en estas páginas es importante destacar que se ha añadido el `microdata` para optimización para buscadores. Una captura se encuentra, a modo de ejemplo, en los ejemplos de código fuente.

Directorio events:

- `Index`, que muestra un listado de todos los eventos activos.
- `_slug`, que mostrará el detalle de los eventos, incluyendo un mapa de `openMaps` con la localización del mismo si se ha añadido ésta en `directus`, con su `microdata` correspondiente.

Asimismo, se han utilizado diferentes layouts de Nuxt para el desarrollo de la aplicación. Los layouts son archivos que envuelven a las páginas y que sirven para aplicar diferentes estructuras a las mismas. Para

esta aplicación, además del diseño por defecto, se han añadido específicos para la lista de redes sociales, el detalle de artículos y páginas y el detalle del producto.

9.2.5 Componentes desarrollados

Menú principal (mainMenu)

Menú responsive de la aplicación. En móvil consta de un menú hamburguesa que, al abrirse el menú, se convierte en unas aspas para cerrarlo. En escritorio se mantiene fijo en la parte superior de la página. Debido a la naturaleza de las animaciones y diseño, y a las limitaciones de la librería TailwindCSS, se ha optado por realizarlo con una mezcla entre ésta y SCSS puro.

Pie de página (mainFooter)

Menú simple que hace las veces de pie de página.

Ítem del pie de página (mainFooterPage)

Cada uno de los enlaces del menú de pie de página se han encapsulado en un componente propio para poder hacer uso de forma efectiva de los computed de Vue ya que, si se hubiera realizado dentro del pie de página hubiera sido necesario un método, que no se cachea, lo que sería un mal patrón de software dentro de la librería VueJS

Tarjeta de artículo para el listado (postCard)

Se ha definido este componente para mantener una constancia en el diseño de todos los elementos que componen el listado completo de artículos. Esto nos permitirá, de igual manera, poder modificar el diseño de forma más sencilla, facilitando la escalabilidad de la página.

Tarjeta de producto para el listado (productCard)

Utilizado, al igual que en los artículos, para mantener una constancia en el diseño de todos los elementos que componen el listado de productos.

Tarjeta de evento para el listado (eventCard)

Igualmente, para eventos.

Enlace de redes sociales (socialMediaLink)

Pensando igualmente en la escalabilidad de la aplicación, en estos momentos es únicamente un enlace externo a una red social, pero, al definirlo como un componente independiente, se puede permitir añadir más funcionalidades de forma más sencilla. Por ejemplo, pedir confirmación antes de abrir un enlace NSFW o añadir diferentes clases dependiendo de la red social o de más atributos añadidos en el backend directus.

Cambiador de tema entre claro y oscuro (themeSwitcher)

Permite cambiar las preferencias del usuario por si desea cambiar el tema de oscuro a claro y viceversa al navegar por la página. Se encarga, asimismo, de gestionar el guardado en el store de Vuex con estas

preferencias. Si prefiere el modo oscuro aparece una luna, y si prefiere el claro, un sol mediante emojis UTF-8.

Modal para compartir en redes sociales (shopMultipleShare)

Componente para encapsular el que provee el plugin instalado vue-social-sharing explicado en esta misma memoria y que monta un listado completo de las redes sociales a las que se quiere compartir en forma de menú desplegable.

Carro de la compra (shopCart)

Cuando se añade algún producto al carro de la compra, este componente crea una interfaz para comprobar el listado de los mismos y el precio total que habría que pagar al terminar la compra. Como se ha explicado anteriormente, se comparte la información entre todas las pestañas abiertas en el dominio en el navegador.

Elemento del carro de la compra (cartListItem)

Para cumplir con el correcto patrón de diseño de software en VueJS que consiste en que no se utilicen métodos con parámetros para filtrar o calcular cambios de información sobre el estado, sino que se deben crear componentes hijos que utilicen computados para esta labor, se ha creado este componente para cada uno de los productos que haya en el carro, incluyendo los botones para añadir uno, quitar uno o quitar todos los elementos de un producto del carro de la compra.

Galería de imágenes desde directus (directusGallery)

En el backend, como se ha explicado anteriormente en esta memoria, en la parte concerniente a la configuración de las colecciones de directus, cada post, evento o producto puede tener asignada una o más archivos multimedia, que en esta parte de la aplicación se muestran a través de una galería multimedia. Cada una de estas galerías muestran una miniatura por cada una de las imágenes si hay más de una y la imagen activa con mayor tamaño. Los tamaños se vinculan al nombre con el que se han definido cuando se ha descargado la imagen en la generación estática del contenido.

Imagen individual desde directus (directusImage)

Para cada una de las imágenes individuales se ha creado un componente específico, basado en la etiqueta picture, donde se entrega la imagen en formato JPG o en Webp dependiendo de si el navegador soporta o no formatos más modernos. Además, si se clica sobre ella se lanza un evento que puede ser aprovechado por la galería para marcar la imagen como la activa.

Imagen de directus añadida como fondo (directusBackgroundImage)

No siempre se puede utilizar el componente anterior para mostrar una imagen, ya que hay lugares en los que una picture no sirve porque se desea poner la imagen como fondo. Para esos casos se ha utilizado este componente, que establece, utilizando CSS-in-JS, el estilo para el fondo de una capa.

9.2.6 Forma de pago

Para poder realizar cobros utilizando este proyecto se ha optado por la plataforma Stripe. Stripe es un sistema de pago online que se puede integrar directamente en cualquier página web de una tienda online, aunque esta web o tienda sea totalmente estática, como es el caso de este proyecto.

Sería, por esto, un sistema de pago independiente, como existe también otros como PayPal, pero con la diferencia de que no envía al comprador a otra web externa para finalizar el pago, sino que el formulario de pago está dentro de la propia Stripe si así se desea. Sólo se necesita una cuenta y añadir, en el caso de este proyecto, los productos en la plataforma. Al añadir los productos nos da un código único de precio que se añade a directus para poder vincular nuestro producto de la tienda estática con el futuro pago en la plataforma.

Los pagos con Stripe se realizan siempre a través de un intermediario financiero, bien sea tarjeta de crédito bien sea Google o Apple pay, por lo que los clientes no necesitan tener creada una cuenta en la plataforma, al contrario de lo que pasa con Paypal, únicamente deben rellenar el formulario con sus datos (que se comparten con el artista para saber dónde se deben entregar los pedidos sean digitales o no) y los de su forma de pago para el cobro.

Como toda plataforma, Stripe conlleva unos precios que se deben tener en cuenta y que se traducen en una serie de comisiones, que varían según el país.

En España, las comisiones de Stripe son las siguientes:

- Pagos con tarjetas europeas: 1,4% + 0,25 € por transacción
- Pagos con tarjetas no europeas: 2,9% + 0,25 € por transacción

Estos precios están muy por debajo de lo que tiendas como Etsy cobran a los artistas, un 6.5% por cada transacción, además del 4 % + 0,30 € por actuar como pasarela de pago. Sin contar con que, si se desean herramientas más allá del paquete estándar, tienen un coste adicional de 8.5€ / mes.

En esta parte del proyecto, incluso intentando tener en posesión todos los datos que se puedan, tal y como se ha comentado en el punto 4 de esta memoria, es difícil por no decir imposible, el no depender de una gran compañía para la gestión de cobros y pagos, sea Stripe la elegida o un banco con el que trabaje el artista. Se ha intentado, de esta manera, que la solución elegida sea la más simple para quien dirija la tienda, no penalizando a pequeños artistas que no puedan acceder a grandes beneficios con un banco o contratar a alguien específicamente para llevar estos temas.

Para la implementación de esta pasarela de pago sobre el proyecto se ha hecho uso de Vue Stripe, disponible tanto para Nuxt como para el uso en aplicaciones hechas con VueJS directamente. Este plugin

está ampliamente soportado por una comunidad activa, con lo que no se corre el riesgo de que no sea migrado a futuras versiones de Nuxt (cuya migración, por cierto, está casi terminada)

En la página del carrito de la compra del proyecto se ha añadido uno de los componentes que provee este plugin, stripe-checkout, al que se le ha pasado como propiedades el listado de productos con el formato que espera el mismo, traduciendo a través de un atributo computado (que, recordemos, se cachean mientras el atributo vinculado no se modifique) desde los elementos del carrito de la compra filtrando todos aquellos que no puedan ser vinculados (por seguridad y evitar nulos, ya que no se podrían haber añadido en primera instancia al carrito si el atributo stripePricePK es nulo).

Los elementos que espera este componente son bastante sencillos, siendo una dupla de precio (al que se le añade el identificador único de precio provisto por Stripe) y la cantidad de productos que se desean comprar.

Esto nos redirige a la plataforma donde se pedirán todos los datos y se completará el cobro al cliente, avisándonos automáticamente de la venta a través de un correo electrónico con los datos del comprador.

9.2.7 Formulario de contacto

Esta página de formulario de contacto, aunque está definida dentro de las páginas creadas en la carpeta de páginas, creo que merece un lugar aparte y una explicación más detallada ya que es la única que se comporta de esta manera de entre todas las páginas definidas en este punto.

El formulario de contacto únicamente será funcional en la web del artista que tenga una conexión directa con un directus totalmente accesible a través de la red, es decir, de una web que no esté completamente generada estáticamente. Debido a que, en el caso del formulario de contacto, la respuesta se graba en el headless backend a través de una llamada API, si es totalmente estática éste no tiene sentido. Es por esto que, si se intenta acceder a la URL de /contact (o su equivalente en castellano /es/contact) la web automáticamente redirigirá al usuario al inicio.

La función que se comunica con el backend se ha añadido en el mismo plugin que recoge la información de directus, ya que también conecta con la plataforma. Conceptualmente podría haberse creado también como plugin independiente, pero se ha optado por esta opción.

10. Prototipos

10.1 Prototipos Lo-Fi

10.1.1. Posts

Post o página en tamaño escritorio

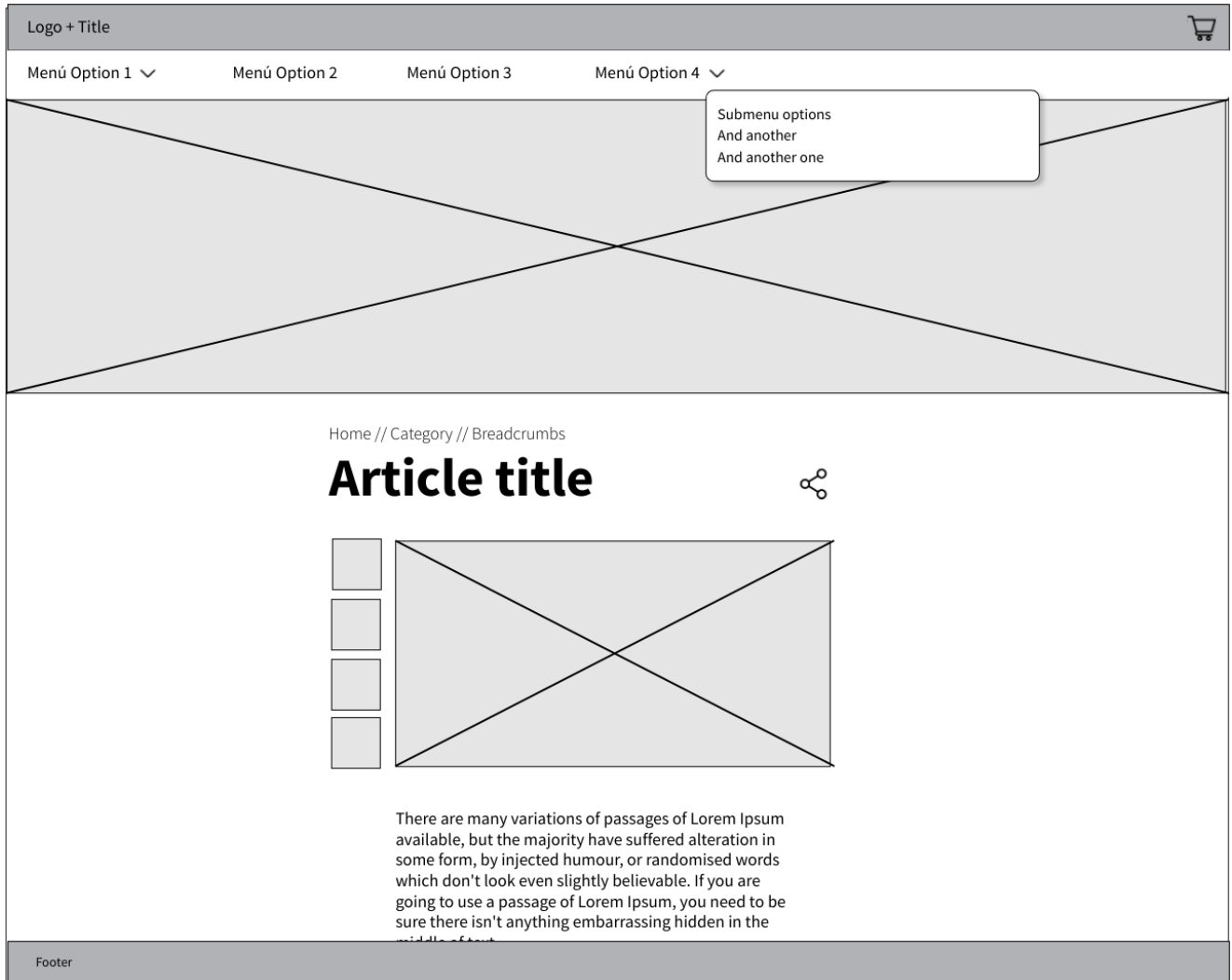


Ilustración 6: Página de artículo, tamaño escritorio

Tamaño móvil, con y sin artículos relacionados:

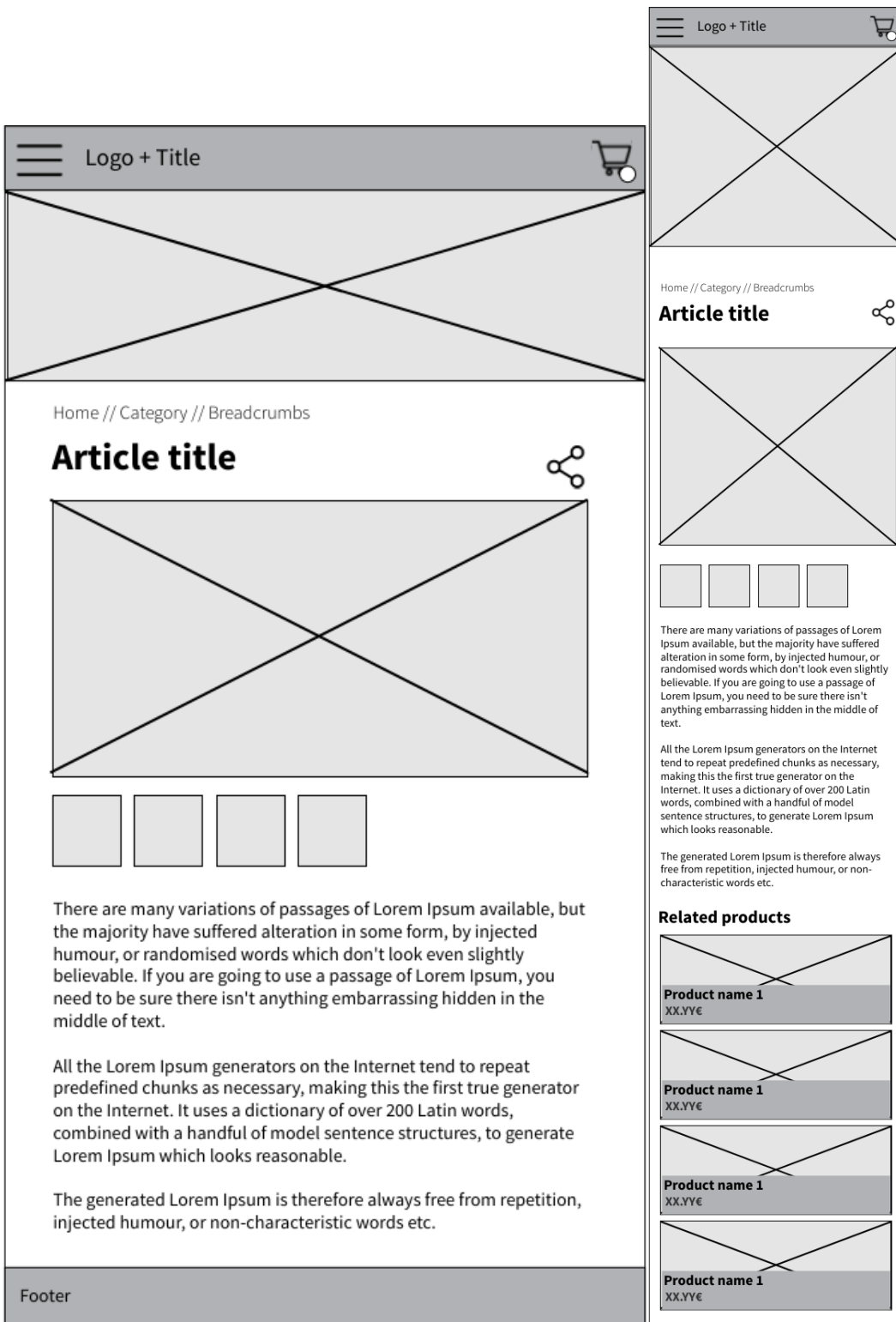


Ilustración 7: Página de artículo, tamaño Tablet y móvil

10.1.2. Categorías y etiquetas

Versión escritorio y móvil, respectivamente:

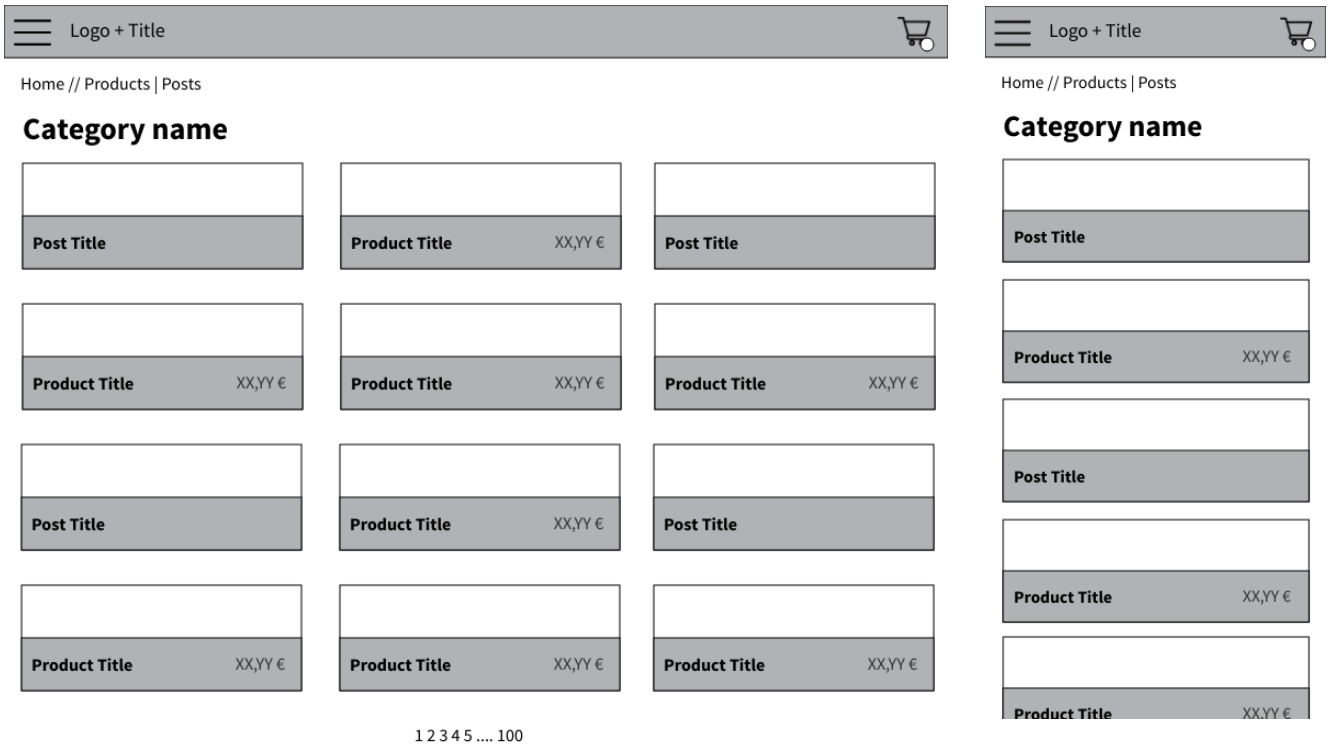


Ilustración 8: Página de categoría y etiquetas, versiones escritorio y móvil

10.1.3 Listados

Tablet y móvil, respectivamente

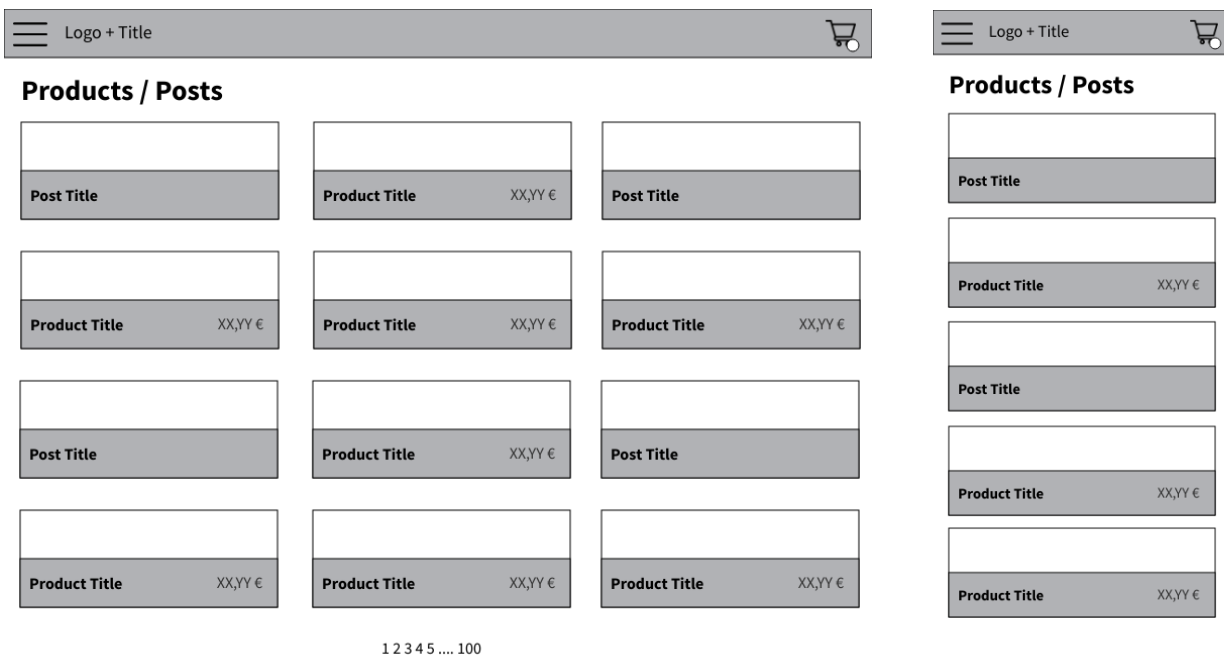


Ilustración 9: Listados generales, tamaños Tablet y móvil

10.1.4 Página de producto

Versión móvil, con el botón de añadir a la cesta fijo y la versión para otras resoluciones

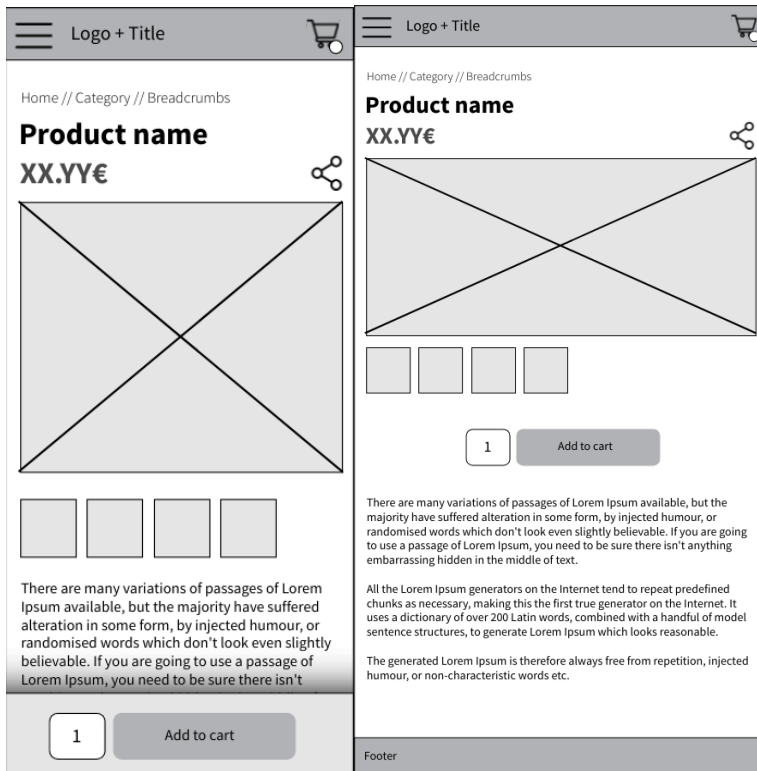


Ilustración 10: Página de producto, móvil y otras resoluciones

10.2 Elección de paleta de colores y tipografías

10.2.1 Elección de la tipografía

Para la elección de la tipografía por defecto, que se podrá modificar por parte de los artistas que utilicen la plataforma, se ha optado por tres tipografías distintas, cada una con un único peso para que la carga de la web no se resienta. Estas han sido **Montserrat**, para los títulos, **Lora Italic** para los textos secundarios y, para los párrafos y texto no destacado, **Hind Madurai**. En la imagen a continuación se puede ver cómo funcionan juntas en una página web que respete pesos y márgenes:

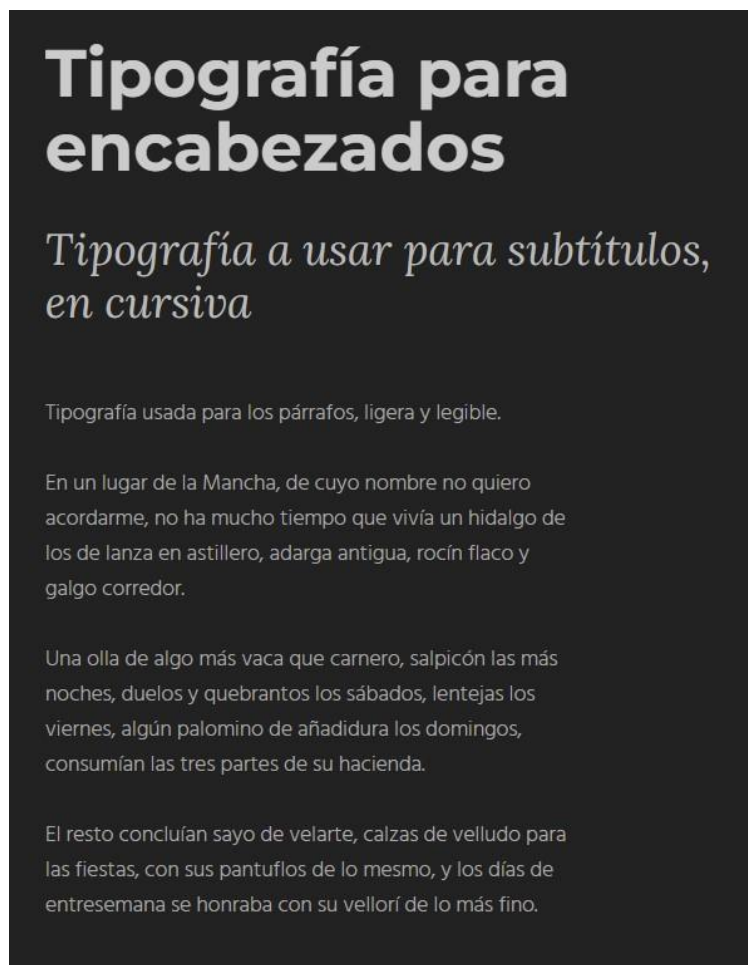


Ilustración 11: Estructura de la tipografía seleccionada

10.2.2 Elección de las paletas de colores

Para la imagen por defecto de la tienda se han elegido dos paletas de colores diferenciadas, una para si los clientes prefieren utilizar el modo oscuro, que se usará por defecto, y otra para el modo claro de la misma, utilizando colores similares, pero ajustando tanto el naranja como el azul para que se respeten los criterios de accesibilidad AA.

Se han elegido tonalidades para los fondos y textos por colores que son neutros, pero caen hacia los colores claros, basados en el marrón, ya que nos proporcionan limpieza y serenidad a la vez que transmiten amplitud y limpieza si se respetan, en los diseños, los espacios blancos en la navegación. Basada en esta elección se ha escogido el color de los titulares, que mantiene la misma gama cromática.

Por contraste con esta gama se ha elegido el azul que dará un toque de contraste en el color en enlaces, botones y en aquellos lugares que se necesite un toque que atraiga al cliente.



Ilustración 12: Paletas de colores oscuros y claros.

Una vez se han elegido estas paletas de colores similares para ambos modos, se han añadido a las opciones de la tienda como código CSS que se importará, al igual que las tipografías, en la generación del sitio estático.

Una de las ventajas, como hemos comentado, del headless cms que se ha añadido como backend es la versatilidad del tipo de campos que se pueden añadir. En este caso, como se puede ver en la siguiente imagen, se ha optado por dos campos distintos en los que se añade directamente CSS, uno para las tipografías y otra para las paletas de colores.

Estas paletas de colores, definidas mediante variables CSS se utilizarán después para generar el estilo que utilizará el framework TailwindCSS

Typographies

```

1 /* Paste here your CSS Fonts */
2 /* Default: Montserrat + Lora + Hind Madurai */
3 @import url('https://fonts.googleapis.com/css2?
  family=Hind+Madurai:wght@300&family=Lora:ital@1&family=Montserrat:wght@700&display=swap
  ');

```

CSS Palette

```

1 :root {
2   --primary: #2d302c;
3   --secondary: #E6E2DD;
4   --tertiary: #e99276;
5   --quaternary: #9bd8db;
6   --menu: #252825;
7 }
8 @media (prefers-color-scheme: light) {
9   :root {
10    --primary: #E6E2DD; /* Light grey */
11    --secondary: #373A36; /* Dark brown */
12    --tertiary: #5c3021; /* Dark orange */
13    --quaternary: #0d5e62; /* Medium blue */
14    --menu: #c3b3ad;
15   }
16 }
17 body.prefers-color-dark {
18   --primary: #2d302c;
19   --secondary: #E6E2DD;
20   --tertiary: #e99276;
21   --quaternary: #9bd8db;
22   --menu: #252825;
23 }
24 body.prefers-color-light {
25   --primary: #E6E2DD; /* Light grey */
26   --secondary: #373A36; /* Dark brown */
27   --tertiary: #5c3021; /* Dark orange */
28   --quaternary: #0d5e62; /* Medium blue */
29   --menu: #c3b3ad;
30 }

```

Ilustración 13: Tipografías y paleta de colores con todas las opciones añadida al backend, modificable por los artistas

11. Perfiles de usuario

11.1 Artista

El artista será el principal usuario de la plataforma, ya que deberá ser el o la que se encargue de añadir los artículos, arte y páginas de los que constará la página, dejando a su gusto la plataforma y encargándose de gestionar los pedidos que se hagan a través de la misma.

Será el artista el que, bajo su responsabilidad, deberá encargarse asimismo de respetar la ley que dependa en su país acerca de las ventas y las páginas que deben mostrarse para cumplirla. En el caso de la Unión Europea, y de cara a que se pueda cumplir correctamente la RGPD, se han generado páginas por defecto en el modelo de datos de accesibilidad, aviso legal y política de envío y devoluciones, tal y como se puede ver en el *sitemap* publicado en esta misma memoria en el apartado 15.1. En estas páginas se recomienda contactar con un asesor o abogado si no se conocen bien los requisitos de la ley.

11.2 Administrador (opcional)

Si el artista no está conforme con la funcionalidad por defecto de la plataforma y desea ampliar la misma, puede tener que contactar con un desarrollador que se encargue de modificar el modelo de datos y la funcionalidad de la web hasta que cumpla con las expectativas del o de la artista.

Debido a la licencia elegida, se espera que las modificaciones se publiquen como código abierto de la misma manera, si bien las extensiones de directus o de nuxt que sean creadas desde cero y no pertenezcan a la funcionalidad por defecto de la misma no deberían ser compartidas, aunque, si se crea una buena comunidad de software libre, sería una forma de mejorar la plataforma, aunque no siempre se consigue.

11.3 Cliente

La columna vertebral de toda página en la que se vayan a realizar ventas. Pensando en estos clientes, la web en todo momento debe tener criterios de publicación de accesibilidad, usabilidad y buenas prácticas. Como requisito de este proyecto no se obliga al usuario a registrarse en la plataforma, protegiendo sus datos, pero sí se entiende que, en un futuro, podría, de forma opcional, acceder a una parte privada en la que pudiera gestionar sus pagos, datos y pedidos.

12. Usabilidad/UX

12.1 Sitemap

En la siguiente imagen se puede ver el *sitemap* de la tienda que se implementará en las siguientes entregas una vez se realice el front de la misma:

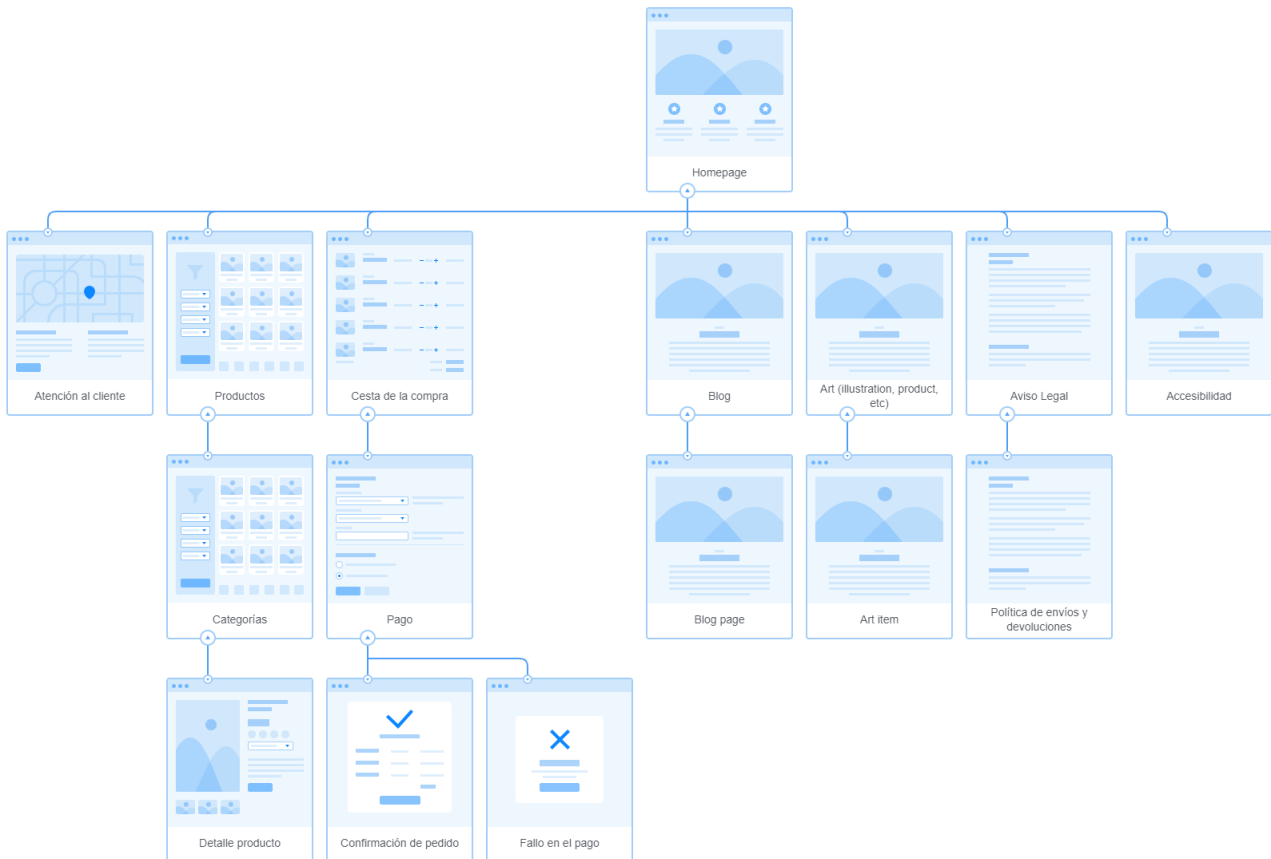


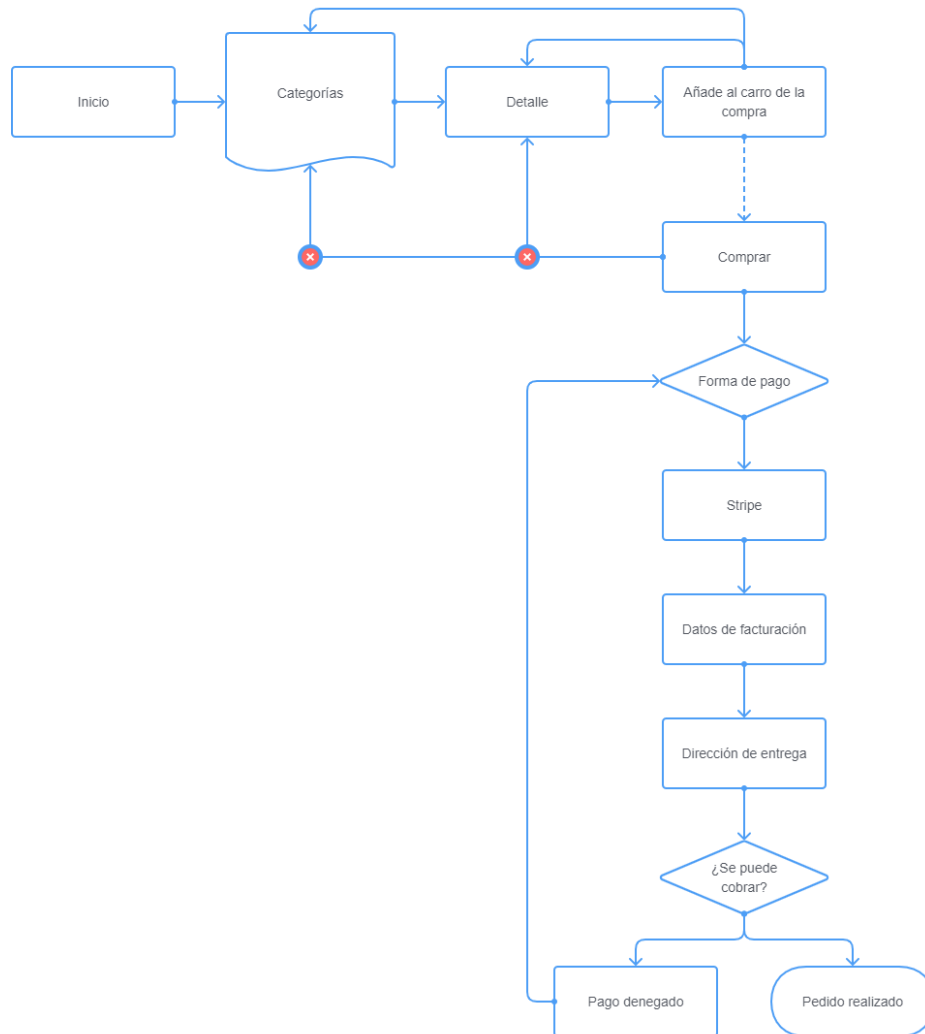
Ilustración 14: Sitemap de la tienda de este proyecto

Vería interesante, como alternativas a estas navegaciones, pensando además en una estructura más de grafo y menos de árbol estricto de un *sitemap*, añadir enlaces a compra directa desde cada uno de los detalles de producto, Incluso, en pantallas grandes, manteniendo visible lo que hay en la cesta según se navega, incluso con diferentes pestañas.

12.2 Estructura de navegación de un cliente

En la siguiente imagen se puede ver el diagrama de flujo que se seguirá en la aplicación:

TFM Flow



27 April 2022

Ilustración 15: Flujo de navegación de un cliente de la tienda

Como comienzo, se toma la página de inicio, a partir de la cual se puede navegar hacia las categorías y llegar hasta la página del detalle.

Las páginas de categorías serían una página múltiple. Una vez en la página de detalle del producto se podrá añadir a la cesta / carro de la compra. Desde este estado se muestran varias opciones:

1. Navegar de vuelta a otra categoría distinta, para buscar otro producto
2. Navegar de vuelta a la página de detalle para poder comprar más unidades del producto.
3. Navegar hacia el proceso de compra. La flecha discontinua nos muestra una condición. Esta sería que no se puede navegar al proceso de compra si no hay productos añadidos en la cesta.

Los puntos 1 y 2 permitiría al usuario poder realizar compras múltiples.

Una vez comenzado el proceso de compra no se permitiría volver a buscar más elementos para añadir al carrito, esto se representa como dos flechas con un signo de prohibición hacia detrás. No es tampoco exactamente lo que se ve en la lectura del tema, pero creo que de esta forma se queda claro.

El siguiente paso del proceso de compra, la recolección de los datos de entrega, lo he dividido en dos pasos, el primero los datos personales para la facturación y el segundo la recolección de los datos de entrega, para que sea más sencillo para el usuario.

En el proceso de pago se utilizará Stripe, que permite un uso totalmente estático de su tienda siempre y cuando los códigos únicos de producto se añadan también en esta plataforma. Si por cualquier motivo no se logra o se cancela el cobro, se devuelve al usuario al estado anterior.

Si todo va ok, se acaba el proceso y el artista recibirá un correo avisando de esa compra.

13. Seguridad

En la elección de Directus como headless CMS y de Nuxt como framework frontend se ha tenido en cuenta que, debido a que el público objetivo de esta aplicación no tiene por qué tener un conocimiento amplio en cuanto a cuestiones de seguridad, deben ser soluciones que tengan amplio soporte y comunicación abierta sobre errores y fallos de seguridad.

Durante el desarrollo de este TFM Directus ha desarrollado hasta 8 versiones nuevas de su producto, varias de ellas por errores de seguridad. Actualizarla mediante la integración continua del sistema es muy sencillo tanto para entendidos como para usuarios con menor conocimiento. Es más, cuando arranca el sistema, si hay alguna actualización de software éste avisa, para que el usuario esté al tanto de las nuevas versiones.

En el caso del headless CMS habría que realizar los dos siguientes pasos:

1. Actualizar la nueva versión en el fichero package.json. Pese a ser un fichero normalmente para desarrolladores, en este caso, al ser pequeño, es bastante manejable. Sólo habría que actualizar el valor en la versión de "directus"
2. Ejecutar el script `npm run update`, esto actualizará la librería y hará las migraciones pertinentes sobre la BBDD, tal y como otros frameworks como Laravel en PHP, hacen.

Si el usuario deseara seguir con una versión únicamente en local porque su web es totalmente estática, podría pasar sin actualizar durante un tiempo, aunque siempre se recomienda, por supuesto, tener la última versión del software.

Por otra parte, el uso de Nuxt como generador de sitios estáticos, si se elige esa opción, evita muchos problemas de seguridad en aplicaciones convencionales. Al no depender de conexiones a un servidor que envíe los datos, sino que se generan ficheros HTML que se hidratan en el propio momento de la creación, por lo que se minimizan los posibles ataques.

Al depender de una plataforma externa para los pagos también hace que no se deban guardar datos personales de clientes en servidores propios, evitando posibles brechas en ataques.

Por supuesto, todo dependerá de las buenas prácticas en gestión de contraseñas que pueda tener el artista que se encargue de mantener el sitio.

14. Tests de aceptación

Con la intención de comprobar el correcto funcionamiento, más de cara al cliente final que accede a la web generada e intenta comprar que en sí al artista que alimenta los datos, ya que son los clientes la razón de ser de cualquier web que contenga productos, un escaparate y, sobre todo, una forma de pago, se han realizado una serie de tests de aceptación.

Estos tests de aceptación se realizan para comprobar funciones o características que se conocen de antemano, por lo que son medibles y cuantificables. En un futuro, mediante tests e2e, se podrían automatizar y realizar pruebas de regresión para evitar que cualquier cambio en la plataforma tenga efectos secundarios adversos no deseados.

Las desventajas de este tipo de tests son dos, fundamentalmente. La primera, es necesario una planificación detallada e importantes recursos en cuanto a tiempo mientras no se encuentran automatizados. La segunda, como sólo se buscan posibles errores conocidos, se podría ignorar problemas importantes.

14.1 Batería de pruebas como cliente

| | | |
|--|--|----|
| Escenario | El usuario puede cambiar de idioma la web | |
| Código | TA.01 | |
| Precondiciones | Ninguna | |
| Pasos | Resultados esperados | |
| Acceder a cualquier página de la web | La web se ve en el idioma del usuario o, en su defecto, inglés si éste no está soportado | Ok |
| Pulsa sobre el botón de cambio de idioma | La web y su contenido cambian automáticamente de idioma | Ok |

Tabla 2: Prueba de aceptación 01 como cliente

| | | |
|---|---|----|
| Escenario | El usuario puede ver la web en claro u oscuro | |
| Código | TA.02 | |
| Precondiciones | Ninguna | |
| Pasos | Resultados esperados | |
| El usuario accede a la web, a cualquiera de sus páginas | La web se ve en la preferencia de sistema operativo, claro u oscuro, del usuario | Ok |
| El usuario pulsa sobre la luna o el sol | La web cambia de paleta de colores de claro a oscuro y viceversa | Ok |
| El usuario recarga la web | La aplicación recuerda para el usuario la paleta de colores deseada, clara u oscura | Ok |

Tabla 3: Prueba de aceptación 02 como cliente

| | | |
|---|--|----|
| Escenario | Compra de un producto | |
| Código | TA.03 | |
| Precondiciones | Ninguna | |
| Pasos | Resultados esperados | |
| El usuario accede a cualquier parte de la web | La web carga correctamente | Ok |
| El usuario accede a la página de Productos | Se ve el listado de productos de la web, marcando aquellos que se pueden comprar | Ok |
| El usuario accede a uno de los productos | Se carga el detalle del producto con precio y características. | Ok |
| El producto está agotado | El botón de comprar permanece deshabilitado | Ok |
| El producto tiene stock | El botón de comprar está disponible | Ok |
| El cliente pulsa sobre el botón de comprar | Aparece el producto en el carrito, que pasa a ser clicable | Ok |
| El cliente va al carrito completo | Puede modificar el carro de la compra añadiendo o eliminando unidades | Ok |
| El cliente pulsa sobre el botón de pagar | Se redirige a la web de la pasarela de pago con precio y unidades correctas | Ok |
| El cliente cancela el pago | Se le muestra una página de error | Ok |
| El cliente completa el pago | Se le muestra una página de confirmación | Ok |

Tabla 4: Prueba de aceptación 03 como cliente

| | | |
|---|---|----|
| Escenario | Localización de eventos | |
| Código | TA.04 | |
| Precondiciones | El usuario ha accedido a la web | |
| Pasos | Resultados esperados | |
| El usuario pulsa sobre el menú de Eventos | Se muestra el listado de enlaces a los eventos disponibles del artista | Ok |
| El usuario pulsa sobre uno de los eventos | Se muestra el detalle con un mapa de localización del evento donde se encontrará el artista | Ok |

Tabla 5: Prueba de aceptación 04 como cliente

14.2 Batería de pruebas como artista / administrador

| | | |
|--|-------------------------------|----|
| Escenario | Login del artista en directus | |
| Código | TA.05 | |
| Precondiciones | Usuario no autenticado | |
| Pasos | Resultados esperados | |
| El artista accede a la web configurada de directus | Aparece pantalla de Login | Ok |
| El artista rellena los datos. | | |
| - Si son incorrectos | Aparece pantalla con error | Ok |
| - Si son correctos | Accede a su zona privada | Ok |

Tabla 6: Prueba de aceptación 01 como artista

| | | |
|---------------------------------------|--|----|
| Escenario | Añadir nueva entrada | |
| Código | TA.06 | |
| Precondiciones | El artista está autenticado | |
| Pasos | Resultados esperados | |
| Accede a la pestaña de Posts | Se ven los posts ya añadidos | Ok |
| Pulsa sobre el botón + | Se ve el formulario completo | Ok |
| El artista rellena los datos y guarda | La nueva entrada se muestra en el listado | Ok |
| | La nueva entrada se muestra en el listado de entradas de la web pública. | Ok |

Tabla 7: Prueba de aceptación 02 como artista

| | | |
|---|---|----|
| Escenario | Añadir nuevo producto | |
| Código | TA.07 | |
| Precondiciones | El artista está autenticado y ha recibido el código de producto de stripe | |
| Pasos | Resultados esperados | |
| Accede a la pestaña de Manufacture | Se ven los productos ya añadidos | Ok |
| Pulsa sobre el botón + | Se ve el formulario completo | Ok |
| El artista rellena los datos, incluyendo el código de stripe y guarda | El nuevo producto se muestra en el listado | Ok |
| | El nuevo producto se muestra en el listado de entradas de la web pública. | Ok |

Tabla 8: Prueba de aceptación 03 como artista

| | | |
|-----------------------|-----------------------------|--|
| Escenario | Añadir nuevo evento | |
| Código | TA.08 | |
| Precondiciones | El artista está autenticado | |
| Pasos | Resultados esperados | |

Portfolio y tienda online para pequeños artistas, Eduardo Rey Jara

| | | |
|--|--|----|
| Accede a la pestaña de Events | Se ven los eventos ya añadidos si los hubiera | Ok |
| Pulsa sobre el botón + | Se ve el formulario completo | Ok |
| El artista rellena los datos. Incluyendo el mapa del lugar | El nuevo evento se muestra en el listado | Ok |
| | El nuevo evento se muestra en el listado de entradas de la web pública con la misma situación en el mapa que se ha añadido en el formulario. | Ok |

Tabla 9: Prueba de aceptación 04 como artista

| | | |
|--|--|----|
| Escenario | Añadir nuevo idioma a la traducción | |
| Código | TA.08 | |
| Precondiciones | El artista está autenticado | |
| Pasos | Resultados esperados | |
| Accede a la pestaña de Options > Languages | Se muestran los idiomas disponibles | Ok |
| Pulsa sobre el botón + | Se ve el formulario completo | Ok |
| El artista rellena los datos del idioma | El nuevo idioma se muestra en el listado | Ok |
| | El nuevo idioma se muestra disponible en todas las pestañas anteriores de administración | Ok |
| | El idioma no se encuentra disponible por defecto en la web, ya que habría que traducir esas cadenas. | Ok |

Tabla 10: Prueba de aceptación 05 como artista

15. Instrucciones.

Información con pasos detallados acerca de cómo se debe instalar/implantar el servicio/aplicación. Las presentes instrucciones deben también acompañar al servicio/aplicación en un archivo contenido en sus directorios.

15.1 Instrucciones de instalación/implantación

Las instrucciones que se han seguido para instalar las dependencias en Linux se han definido en el punto 9.1.2 de esta memoria, por lo que en este apartado se especificará, y asimismo se incluirá traducido al inglés junto al código fuente, la instalación de NodeJS y MariaDB en Windows / Mac, los sistemas operativos más utilizados por los artistas.

15.1.1 Instalar BBDD

Si el artista no opta por la versión community de directus y desea tener todo el control de sus datos, necesitará instalar una base de datos MariaDB en su propio ordenador. Para ello, debería seguir lo siguientes pasos:

1. Acceda al sitio web de [MariaDB](#) y descargue la última versión de la base de datos para su sistema operativo.
2. Ejecutar el instalador descargado.
3. Seguir paso por paso las indicaciones del instalador. En el caso de que no se tenga muy claro qué cambiar o qué no, lo mejor es dejar los datos por defecto.
4. Cuando se pida usuario y contraseña, elegid la que se desee. Se puede dejar por defecto, pero es importante recordarlas porque serán las que haya que poner en los ficheros de variables de entorno de nuestras aplicaciones.
5. Para usuarios no familiarizados con la gestión de Base de datos, recomendamos dejarla como servicio. Esto evitará que haya que arrancarla manualmente cada vez que tengamos que acceder a la misma.

15.1.2 Instalar Node

Para poder utilizar el gestor de paquetes npm que permitirá al artista acceder a todo de forma local, se debe instalar Node, siguiendo unos pocos y sencillos pasos.

1. Entrar en <https://nodejs.org/es/download/> y descargar el instalador de Node.js en el sistema operativo deseado.
2. Ejecutar el instalador que hemos descargado de la web. Simplemente debemos ir siguiendo cada uno de los pasos. En el caso de tener dudas al respecto de alguna opción que nos pregunte, lo mejor es dejar la opción por defecto.
3. Una vez finalizada la instalación, podemos irnos a nuestra terminal favorita (si el usuario está en Windows y no conoce el término, debe ir al menú principal y buscar “símbolo de sistema”) y probar que se ha instalado correctamente escribiendo el comando “npm -v”

15.1.3 Configurar Nuxt

Para que Nuxt arranque correctamente, debemos renombrar el fichero de la raíz `.env.example` a `.env` y cambiar los valores que hay dentro con las direcciones web que se utilizarán cuando tengamos nuestra web publicada (`Nuxt_ENV_BASE_URL` y `Nuxt_BASE_URL`), dónde está configurado nuestro Directus, al que nos conectaremos (`DIRECTUS_BACKEND_URL`) y, si deseamos una tienda, la clave pública que nos provee la pasarela de pago Stripe en `STRIPE_PK`. Una vez hecho esto, podemos arrancar con `npm run dev` o `npm run serve`

15.1.4 Transpilar web

Una vez hayamos terminado de configurar nuestros productos, añadidas las páginas que deseemos y creamos que ya podemos subir la web para que los clientes puedan acceder, hay que irse a la carpeta en la que tenemos este proyecto, acceder a la parte frontend que se encuentra en la carpeta `nuxt-frontend`, abrir un interfaz de comandos y ejecutar:

`npm run shop:generate`

Este comando descargará todo el contenido necesario para generar la web html que podremos desplegar en nuestro hosting favorito. En el siguiente punto se mostrará cómo desplegarlo en Netlify, una plataforma especializada en el alojamiento y automatización de proyectos web estáticos.

15.1.5 Desplegar en Netlify

Hay dos formas en las que se puede desplegar sobre Netlify. Una de ellas, la que suele ser más usada por personas con conocimientos más avanzados, es configurar un pipeline a través de un repositorio de git en el que la propia plataforma lee y compila. Así funciona también alternativas como Vercel.

Pero Netlify tiene una segunda opción, que le serviría a todos aquellos para los que leer el párrafo anterior pueda haberle causado migrañas. Una vez transpilada la web, se puede, simplemente, arrastrar el directorio `dist` resultante sobre la web de Netlify y se desplegará automáticamente. Para ello, habría que seguir únicamente dos pasos.

El primero de ellos, crear un nuevo sitio, eligiendo sobre el desplegable la opción “desplegar manualmente” (`Deploy manually`) tal y como se ve en la imagen a continuación.

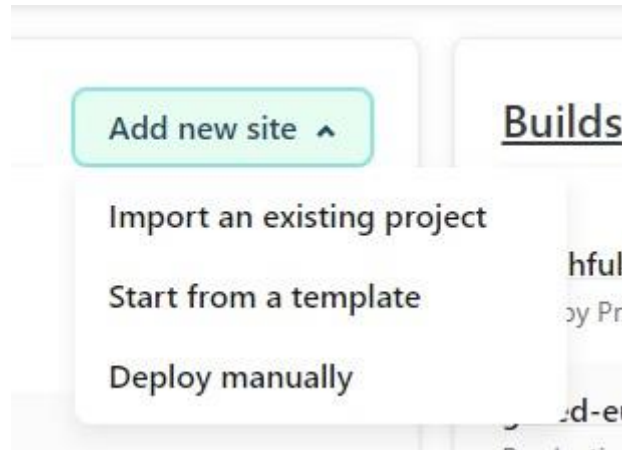


Ilustración 16: Crear un nuevo proyecto en Netlify

Hecho esto, únicamente tenemos que arrastrar el directorio resultante en el lugar que marca la web y que se puede ver en la imagen 17 y esperar que se despliegue:

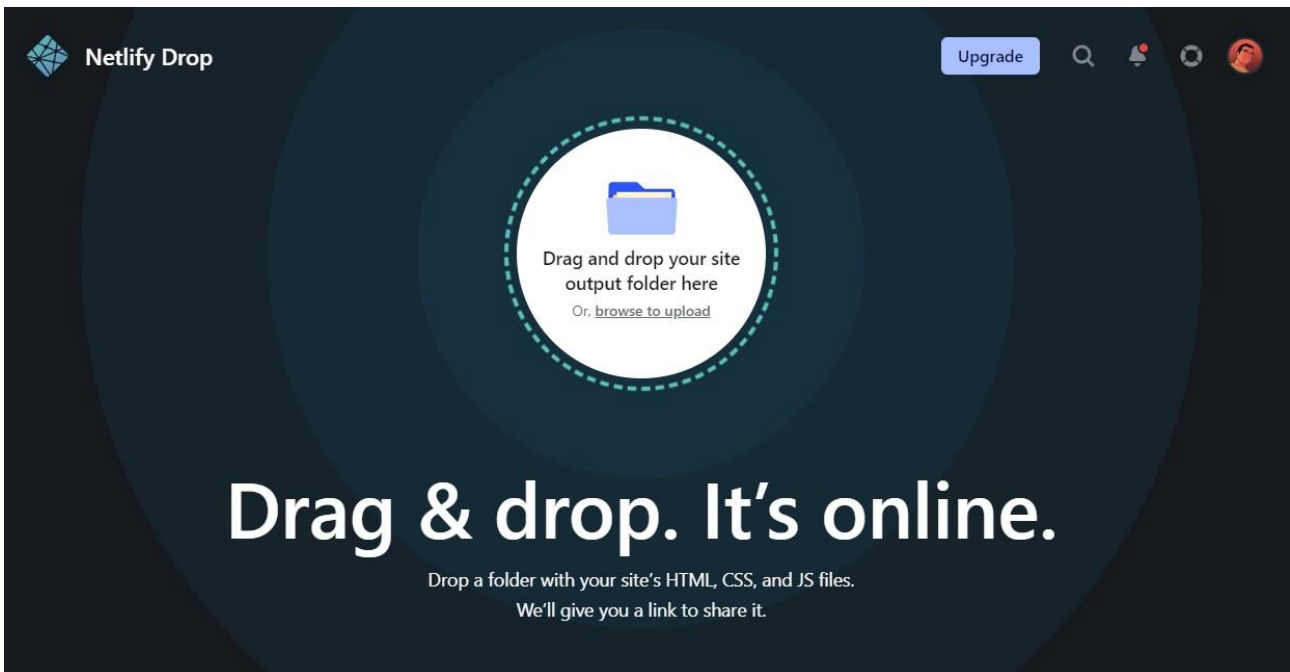


Ilustración 17: Lugar donde se puede arrastrar el directorio resultante.

Y ya sólo quedaría esperar. Si por el motivo que fuera tardase mucho en desplegar (más de unos pocos minutos), se debe comprobar que cualquier extensión del navegador no está bloqueando la subida, como, por ejemplo, un bloqueador de publicidad. Si esto sucede, sólo habría que cancelar, desactivar la extensión y volver a subir el directorio.

Una vez se haya desplegado, una imagen similar a la de la imagen 18 se verá en la pantalla, donde se muestra también un ejemplo de pipeline fallido.

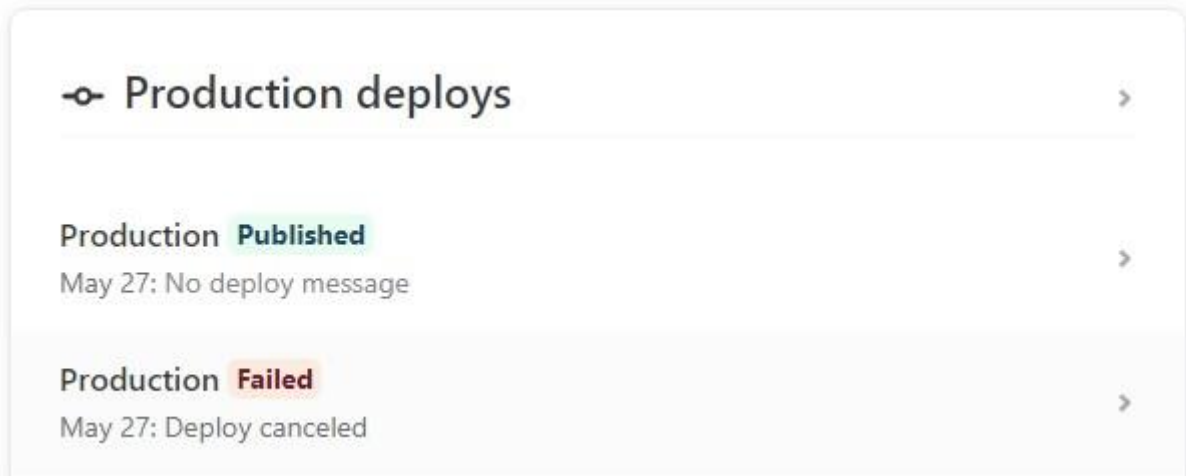


Ilustración 18: Ejemplo de pipelines correctos y fallidos en Netlify

15.2 Instrucciones de uso de directus.

La documentación disponible para la versión más reciente de la plataforma está siempre disponible en <https://docs.directus.io>, y los documentos específicos de la versión se incluyen en la propia aplicación como módulo central, por lo que se puede acceder a ellos desde la propia interfaz de administración disponible para el artista.

Ambos proceden del código fuente y está disponible en formato markdown en el propio repositorio de la plataforma en GitHub, lo que permite que aprender a usar la plataforma sea más sencillo.

Asimismo, desde esta documentación podemos encontrar enlaces en vídeo detallado con tutoriales para todos los niveles, que sería imposible reproducir detalladamente en esta memoria.

16. Proyección a futuro

Este proyecto nace con la idea de plantear una solución a largo plazo para artistas, tal y como se ha comentado en anteriores puntos de esta memoria. Es por esto que se ha planteado a largo plazo, más allá de la terminación de este TFM.

Lo primero que se realizará será buscar un nombre comercial. En tanto en cuanto no tengo formación en marketing o finanzas, se buscará la ayuda de profesionales para lograr un nombre que pueda ser fácilmente recordable. Si bien el nombre actual es totalmente correcto (una tienda en nuxt y directus) no es para nada atractivo de cara a buscar clientes o financiación.

Como objetivos técnicos, se mantendrán actualizadas todas las librerías y frameworks de los que se hace uso. Tal y como se comentó en el punto 7.2 de esta memoria, la versión que se intentaba usar de Nuxt para este proyecto era la 3, pero, al no estar todavía totalmente preparada, se optó por su versión más estable. Una vez salga la versión estable, se migrará a ella, evitando así tener un proyecto inicial que arrastre deuda técnica desde tan pronto.

Además, se añadirán las funcionalidades que los artistas puedan solicitar. Entre las primeras están el añadir otras formas de pago, mayores opciones de personalización y soporte para más idiomas.

Además de lo comentado en párrafos anteriores del nombre del proyecto, con el marketing en mente y pensando en una posible monetización del producto, pese a que se haya pensado y se seguirá actuando como un proyecto de software libre y abierto, también está planeado el realizar una web del producto generada con el propio sistema, con las funcionalidades bien detalladas y tutoriales de instalación y uso para todos los niveles, aunque para aquellos con menos conocimientos informáticos se pueda dar un servicio de configuración a terceros. Es decir, no se plantea cobrar por funcionalidades nuevas o posibles adaptaciones, que serán compartidas como código fuente, sino en posible ayuda y soporte *premium* a usuarios.

17. Presupuesto

Al haberse realizado este proyecto enteramente utilizando software libre y gratuito para el desarrollo de la aplicación y desplegarse en una nube privada, los costes de infraestructura y recursos son de únicamente 20€ + IVA por el uso de ngrok profesional para la conexión a la nube privada durante la duración de este desarrollo, actualmente cuatro meses.

La estimación de tiempo de desarrollo del proyecto es un total de 375 horas de trabajo (véase punto 9).

Suponiendo que el trabajo se hubiera realizado como freelance para un tercero, como un proyecto privado, el precio por hora de un desarrollador full stack que pueda realizar tanto el desarrollo de los plugins de backend como de la web frontend Nuxt completa sería de 45€ + 21% de IVA.

Sabiendo que la retención a priori del 15% la realizaría directamente la empresa cliente al erario público serían 38€ netos por hora. El pago total quedaría de la siguiente manera:

- Precio en bruto desarrollo: 16.955,00 €
- Gastos infraestructura: 80 €
- Retenciones 15%: 2543,25 €
- I.V.A. 21%: 3560,55 €
- Total: **20.515,55 €**
- Ingreso neto en cuenta del freelance: **17.972,30 €** (Bruto + IVA - Retenciones)

18. Análisis de mercado

En este apartado de la memoria del proyecto se intentará ahondar en las diferentes aplicaciones que, por su misma naturaleza y cercanía, pueden considerarse competencia de este proyecto, aunque no sean iguales o tengan las mismas funcionalidades.

18.1 Público objetivo

Como se ha descrito anteriormente en esta memoria, el público objetivo de este proyecto son aquellos artistas que tengan inquietud sobre el uso de sus datos y de los que no se aprovechen empresas que no están alineadas con su forma de pensar o actuar

18.2 Competencia

18.2.1 Redes sociales

Como se ha comentado en el punto 4.1 de esta memoria, las redes sociales se han convertido en un escaparate para todo artista o comerciante que quiera tener visibilidad de su obra. Sólo en España, con 55 millones de líneas móviles en un país de poco más de 47 millones, hay 40 millones de usuarios de redes sociales, siendo Facebook e Instagram las que, con un 73% y un 71% de usuarios respectivamente, las más usadas.

Es innegable que, con estos datos, cualquier artista querrá estar en estas redes, sirviendo de escaparate, pero no todos quieren vender a través de sus Marketplace opacos, por lo que este proyecto puede servir para centralizar las ventas y los productos, evitando las tasas de venta de estas plataformas.

18.2.2 Plataformas online de venta

Dentro de esta categoría entrarían webs como Etsy o Artstation. Son un mercado completo, con todo lo que se puede querer excepto lo ya comentado en el punto 4 de esta memoria, las empresas no tienen por qué alinearse con lo que el artista desea y, además, las tasas de venta que cobran estas plataformas son mucho más altas que si son autogestionadas usando el proyecto con el método de pago de Stripe, tal y como se ha comentado en el punto 10.2 de esta memoria.

Además, tanto Etsy como Artstation tienen limitaciones en cuanto a los productos que se pueden vender. Solo permite la venta de productos artesanales, materiales para hacer manualidades y otros trabajos manuales y artículos vintage o de coleccionismo con más de 20 años de antigüedad. Tampoco permiten exportar los datos por lo que, una vez empiezas a usarlas, quedas como cliente cautivo (el que no quiere o no puede cambiar de producto, servicio o proveedor por otro por el coste que conlleva ya sea en dinero, esfuerzo o incomodidad).

Además, en estas plataformas toda tu competencia se encuentra a un clic de distancia de ti, incluso teniendo productos relacionados en tu propio perfil, con lo que las ventas pueden ser complicadas de finalizar.

18.2.3 Plataformas de distribución de servicios digitales

Estas plataformas, como gumroad, se benefician de la simplicidad de la UI y del uso para mostrar los productos y guiarte hasta su venta. No funciona como escaparate de producto o del artista, únicamente como plataforma simple de compra.

Como desventaja con respecto a este tipo de plataforma es la simplicidad de la que tal vez puede carecer en un principio Directus, pero que suple con mayor versatilidad y posibilidades de personalización. En esta primera fase de desarrollo que corresponde a la elaboración de este TFM, la interfaz simple de este tipo de plataformas es una ventaja, pero hay que entender que es un proyecto que acaba de empezar, y que, utilizándolo, se tiene un mayor control de datos propios.

18.2.4 Wordpress

Como último punto, aunque, sin duda, sea el principal competidor para este proyecto, es el CMS de código abierto Wordpress. Este gestor de contenidos es el más famoso y extendido en Internet. Por su sencillez de uso, de entre todos los sitios web que existen en Internet, tiene una cuota de mercado del 43%. Es decir, casi una de cada dos webs que existen están hechas o montadas sobre esta plataforma. No sólo domina el mercado de blogs o webs personales, sino, gracias a sus extensiones, también tiene una gran cuota de otros mercados, como el comercio electrónico.

Además, la cuota de mercado de Wordpress no sólo no se ha estancado, sino que ha seguido subiendo año a año desde 2011, donde tenía un 13.1% hasta los últimos datos recogidos de 2019 con el 43% antes nombrado.

Aunque el crecimiento de otros creadores de sitios web hospedados como Squarespace, Wix y Shopify es algo que va aumentando poco a poco, todo se resume en una conclusión: WordPress es, sin lugar a dudas, la forma más popular del mundo para crear un sitio web.

En estos momentos del desarrollo de esta nueva plataforma que se intenta comenzar con este modesto TFM sólo se puede aspirar a llegar a ser una opción más de entre las existentes, aunque de momento no se pueda mirar cara a cara a otros gestores de contenido como el que se comenta en este punto, aunque todo ha tenido un comienzo. Por la naturaleza, asimismo, de este proyecto, los creadores de sitios hospedados no serían competencia, ya que, por lo que se ha comentado durante toda la memoria, el creador perdería su libertad y autonomía.

19. Conclusiones

Este proyecto vinculado al trabajo final del máster universitario en desarrollo de sitios y aplicaciones web en el que he estado matriculado los dos últimos años me ha supuesto un gran reto que he disfrutado enormemente. Pese a que hubiera preferido poder contar con la última tecnología de Nuxt 3, para aprender a utilizar una versión que no estuviera algo anticuada, el uso de Nuxt, en general, me ha resultado muy cómodo, tal vez más que las diferentes prácticas que he realizado en Angular durante el transcurso de estos dos últimos cursos académicos.

Creo que, en general, he podido plasmar la idea que tenía de empezar un proyecto que fuera de software libre que devuelva el control a los usuarios de sus datos, que en estos momentos están siendo secuestrados por grandes corporaciones, centrándome primero en un nicho de mercado que conozco, los pequeños artistas y artesanos. Y, además, también considero que se han cumplido los objetivos que me había autoimpuesto al comienzo del desarrollo.

Si me gustaría decir que, pese a que he intentado seguir a rajatabla la metodología de git flow, sobre todo al comienzo del desarrollo he sido bastante más laxo en la nomenclatura de las ramas de lo que hubiera debido. Esta metodología está pensada, sobre todo, en la comunicación de equipos. Al ser hecho por una única persona, normalmente se cae en vicios que se corregirían dentro de un equipo de desarrollo.

Finalmente, creo que el proyecto me ayudará a avanzar de muchas maneras en el futuro de mi vida profesional, dándome la posibilidad de realizarme y ayudar a estos pequeños artistas ahora y en un futuro, ya que me gustaría que este proyecto no muriera aquí y pudiera actualizarlo y lograra que se convirtiera en una alternativa útil para quien lo quiera utilizar.

Anexo 1. Entregables del proyecto

En este proyecto se entregan las siguientes carpetas y ficheros:

- PAC_FINAL_mem_ReyJara_Eduardo.docx, este documento, con la memoria completa del mismo.
- PAC_FINAL_def_ReyJara_Eduardo.pptx, con la presentación utilizada durante la defensa del proyecto.
- PAC_FINAL_prs_ReyJara_Eduardo.pdf, con la presentación destinada a inversores.
- Informe de autoevaluación.
- PAC_FINAL_prj_ReyJara_Eduardo.zip, que incluye asimismo los siguientes documentos y carpetas:
 - nuxt-directus-shop-main.zip, que contiene todo el código fuente del proyecto.
 - Carpeta Extractos_CodigoFuente, con las imágenes añadidas en el Anexo 2, para su correcta visualización si no es posible en esta memoria.

Todo ello separado en dos carpetas, documentación y proyecto.

Anexo 2. Código fuente (extractos)

Comprobación de la validez de los datos que se envíen a través de un API para Directus, en este caso, formulario de contacto. Se usa la librería de Node Joi.

```
1 const Joi = require("joi");
2
3 const bodySchema = Joi.object({
4   username: Joi.string().required(),
5   usermail: Joi.string().required(),
6   usersubject: Joi.string().required(),
7   usertext: Joi.string().required(),
8 });
```

Ilustración 19: Comprobación de que el BodySchema enviado en la petición es correcto.

Para evitar que en el formulario de contacto se pueda enviar código malicioso, se filtra todo el XSS que pueda llegar con bodyParser. Fichero: headless-directus\extensions\hooks\json-body-xss-clean\index.js

```
1 const bodyParser = require('body-parser');
2
3 module.exports = function registerHook({init}, { logger }) {
4   init('middlewares.before', async ({app}) => {
5     app.use(bodyParser.urlencoded({
6       extended: true,
7     }));
8     app.use(bodyParser.json());
9   });
10 };
```

Ilustración 20: Comprobación y limpieza de XSS

Cuando se añade una nueva entrada como post, se comprueba si se ha añadido una fecha de publicación y se añade automáticamente si no. Además, también se limpia la URL de caracteres no válidos normalizando a inglés. Hay hooks similares para cada una de las diferentes colecciones de Directus.

Fichero: headless-directus\extensions\hooks\posts\index.js

```
1 module.exports = function registerHook({filter, action}) {
2   filter('posts.items.create', async (input) => {
3     const now = new Date();
4     let {url, published_date} = input;
5     if (!published_date) {
6       published_date = now.toISOString();
7     }
8     if (!url) {
9       let title = input.translations &&
10        input.translations.length > 0 &&
11        input.translations[0].title
12        ? input.translations[0].title || '' : '';
13       if (title !== '') {
14         title = title.normalize("NFD").replace(/[\u0300-\u036f]/g, "");
15         try {
16           title = title.replace(/[\ \/:;,.&$=\'\\"\\%·#]/ig, '-');
17         } catch(e) {
18           console.log(title, e);
19         }
20         url = `${title.toLowerCase()}`;
21       }
22     }
23     return {...input, url, published_date};
24   });
25 };
```

Ilustración 21: Hook de directus para limpiar la URL y añadir fecha si no se ha añadido.

Guardado en disco, de forma estática, de todas las imágenes de directus para la generación del sitio.

Fichero: nuxt-front\.directus-connection\scripts\images.js

```
1  const getAllImages = async () => {
2    const url = `${host}files/?fields=${config.images.fields.join(',')}`;
3    await axios.get(url)
4      .then((response) => response.data)
5      .catch(function (error) {
6        console.log('error: ', error.response.statusText);
7      })
8      .then((content) => {
9        if (content) {
10         fs.rmSync(`${config.images.exportDir}/`, { recursive: true });
11         fs.mkdirSync(`${config.images.exportDir}/`, { recursive: true });
12         fs.writeFileSync(`${config.images.exportDir}/files.json`, JSON.stringify(content.data));
13         const sizes = config.images.sizes.map((el) => el.name);
14         const formats = config.images.formats;
15         content.data.forEach(el => {
16           fs.mkdirSync(`${config.images.exportDir}/${el.id}/`, { recursive: true });
17           console.log(` - Downloading file -> ${el.title}`);
18           formats.forEach(format => {
19             sizes.forEach(async name => {
20               await saveImageToFile({
21                 imageid: el.id,
22                 format,
23                 name,
24                 title: el.title,
25               })
26             })
27           });
28         });
29       }
30     });
31 }
```

Ilustración 22: Descarga de imágenes a la carpeta assets.

Similar, pero para las diferentes colecciones. Fichero: nuxt-front\directus-connection\scripts\static.js

```

1  const getStaticContent = async (collection) => {
2    // dependiendo de la coleccion
3    const url = `${host}items/${collection}?fields=${config.static.fields[collection]}`;
4    await axios.get(url)
5      .then((response) => response.data)
6      .catch(function (error) {
7        console.log('error: ', error.response.statusText);
8      })
9      .then((content) => {
10     if (content) {
11       try {
12         fs.rmSync(`${config.static.exportDir}${collection}/`, { recursive: true });
13       } catch (error) {
14         console.log(`${collection} has no folder`);
15       }
16       console.log('Founded: ', content.data.length, 'elements in collection', collection);
17       content.data.forEach(element => {
18         const { url } = element;
19         if (url) {
20           const contentToFile = JSON.stringify(element);
21           writeContentToFile({
22             url,
23             content: contentToFile,
24             collection,
25           });
26         }
27       });
28       console.log('File for collection ', collection);
29       const baseDestination = `${config.static.exportDir}${collection}.json`;
30       fs.writeFileSync(baseDestination, JSON.stringify(content.data), { flag: 'w+' });
31     }
32   });
33 };
34
35
36 const collections = config.static.collections;
37 collections.forEach((el, index) => {
38   console.log(`Downloading -> ${el} ... ${index + 1} of ${collections.length}`);
39   getStaticContent(el);
40 });

```

Ilustración 23: Descarga de las colecciones en ficheros estáticos.

La configuración de ambas descargas iría en el fichero `nuxt-front\directusshop.config.js`

```

1  module.exports = {
2    static: {
3      collections: ['posts', 'events', 'social_media', 'options', 'manufacture'],
4      exportDir: './src/static/directus-mocks/',
5      fields:{
6        posts: '*.*,related_product.manufacture_uuid.url,tags.*.name',
7        events: '*.*',
8        social_media: '*.*',
9        options: '*.*',
10       manufacture: '*.*,category.categories_id.name',
11     },
12   },
13   images: {
14     exportDir: './src/static/directus-files/',
15     fields: ['id', 'title'],
16     formats: ['jpg', 'webp'],
17     quality: 50,
18     sizes: [
19       // Outside makes bigger images
20       // Inside makes smaller images with big black borders
21       // Cover is better for thumbnails
22       { width: 64, height: 64, fit: 'cover', name: 'xs' },
23       { width: 128, height: 128, fit: 'cover', name: 'sm' },
24       { width: 568, fit: 'outside', name: 'md' },
25       { width: 650, height: 400, fit: 'cover', name: 'mdcard' },
26       { width: 900, height: 250, fit: 'cover', name: 'cover' },
27     ]
28   },
29 };
30

```

Ilustración 24:Fichero de configuración para la descarga de assets

La conexión con directus desde Nuxt se hará bien si es estático o bien si es directa a través del plugin que se encuentra en `nuxt-front\src\plugins\directusApi.js`, donde, entre otras más específicas, se encuentra esta función universal para descargar las colecciones de forma genérica.

Como se puede ver, si está hecha de forma que sea totalmente generada offline no hace llamada mediante axios sino que utiliza una inyección de dependencias usando require.

```
1 directusCollections.forEach((el) => {
2   inject(`getAll${capitalizeFirstLetter(el)}`, async () => {
3     const isFullStatic = app.$config.isFullStatic
4     if (isFullStatic) {
5       return require(`../static${staticBaseUrl}${el}.json`)
6     } else {
7       const url = `${app.$config.backendUrl}${apiDirectusBaseUrl}${el}?fields=*. *`
8       const items = await app.$axios.get(url)
9         .then(r => r.data)
10        .then(r => r.data)
11      return items
12    }
13  })
14  inject(`get${capitalizeFirstLetter(el)}ByUrl`, async (urlItem) => {
15    const isFullStatic = app.$config.isFullStatic
16    let cleanUrlPage = urlItem.replaceAll('/', '')
17    if (isFullStatic) {
18      if (cleanUrlPage === '') {
19        cleanUrlPage = 'index'
20      }
21      return require(`../static${staticBaseUrl}${el}/${cleanUrlPage}.json`)
22    } else {
23      const url = `${app.$config.backendUrl}${apiDirectusBaseUrl}${el}?fields=*. *&filter[url][_eq]=${urlItem}`
24      const item = await app.$axios.get(url)
25        .then(r => r.data)
26        .then(r => r.data.length > 0 ? r.data[0] : {})
27      return item
28    }
29  })
30 })
```

Ilustración 25: Conexión entre las páginas de Nuxt y los datos almacenados.

Me gustaría destacar, asimismo, la forma de integrar Stripe en el carrito, utilizando un par de computed de Vue para sacar el formato requerido por la plataforma.

Fichero: nuxt-front\src\pages\cart.vue

```
1 export default {
2   name: 'CartPage',
3   mixins: [head],
4   data () {
5     this.pk = process.env.STRIPE_PK
6     return {
7       translate: {
8         title: this.$t('shoppingcartpagetitle')
9       },
10      token: null,
11      successURL: process.client && `${window.location.origin}${window.location.pathname}?state=success`,
12      cancelURL: process.client && `${window.location.origin}${window.location.pathname}?state=error`,
13      redirectState: ''
14    }
15  },
16  computed: {
17    ...mapState(['cart']),
18    ...mapGetters(['totalCart']),
19    currency () {
20      return this.$store.state.options
21        ? this.$store.state.options.currency || ''
22        : ''
23    },
24    currencyIcon () {
25      return this.$store.state.options
26        ? this.$store.state.options.currency_icon || ''
27        : ''
28    },
29    queryState () {
30      return process.client && (new URLSearchParams(window.location.search).get('state') || '')
31    },
32    lineItems () {
33      return this.cart.filter(el => el.stripePricePK !== null).map((el) => {
34        return {
35          price: el.stripePricePK,
36          quantity: el.qty
37        }
38      })
39    }
40  },
41  /* El resto del componente*/
42 }
```

Ilustración 26: Configuración de la pasarela de pago Stripe dentro de Nuxt

En todos los templates de los componentes de las páginas, para cada colección, se ha añadido microdata para facilitar el SEO. En la siguiente imagen se puede ver el template completo de un artículo, con su itemscope e itemtype sacado de Schema.org, incluyendo las migas de pan.

El fichero se encuentra en: nuxt-front\src\pages\products_slug.vue

```

1 <article
2   class="product"
3   itemscope
4   itemtype="http://schema.org/Product"
5 >
6 <!-- eslint-disable vue/no-v-html -->
7 <div class="product__inner">
8   <h1 class="product__title" itemprop="name">
9     {{ translate.title }}
10  </h1>
11  <shop-multiple-share
12    :url="`products/${product.url}`"
13    :title="translate.title"
14    :description="translate.excerpt"
15  />
16  <ol class="breadcrumbs" itemscope itemtype="https://schema.org/BreadcrumbList">
17    <li
18      itemprop="itemListElement"
19      itemscope
20      itemtype="https://schema.org/ListItem"
21    >
22      <a itemprop="item" href="/">
23        <span itemprop="name">{{ $t('appName') }}</span></a>
24        <meta itemprop="position" content="1">
25      </li>
26      <li class="separator">
27        //
28      </li>
29      <li
30        itemprop="itemListElement"
31        itemscope
32        itemtype="/products"
33      >
34        <a
35          itemscope
36          itemtype="https://schema.org/WebPage"
37          itemprop="item"
38          itemId="/products"
39          href="/products"
40        >
41          <span itemprop="name">{{ $t('products') }}</span></a>
42          <meta itemprop="position" content="2">
43        </li>
44        <li class="separator">
45          //
46        </li>
47        <li
48          itemprop="itemListElement"
49          itemscope
50          itemtype="https://schema.org/ListItem"
51        >
52          <span itemprop="name">{{ translate.title }}</span>
53          <meta itemprop="position" content="3">
54        </li>
55      </ol>
56      <h2 v-if="translate.excerpt" itemprop="slogan" class="product__excerpt" v-html="translate.excerpt"></h2>
57      <div v-if="category" class="product__metadata">
58        {{ $t('postUnder') }}
59        {{ $t('category')(category) }}
60      </div>
61      <p class="product__stock" itemprop="availability" :class="product.stock">
62        {{ $t(`${product.stock}description`) }}
63      </p>
64      <div class="product__content">
65        <div class="product__price">
66          <span :class="product.stock">
67            <span itemprop="price">
68              {{ product.price }}
69            </span>
70            <abbr :title="currency" itemprop="priceCurrency" class="product__currency">
71              {{ currencyIcon }}
72            </abbr>
73          </span>
74          <button
75            type="button"
76            class="product__addToCart"
77            :class="product.stock"
78            :disabled="product.stock === 'soldout'"
79            @click="addToCart"
80          >
81            Add to cart
82          </button>
83        </div>
84        <div class="product__data">
85          <span v-if="hasGallery">
86            <directus-gallery itemprop="image" :images="product.gallery" />
87          </span>
88          <div class="product__body" itemprop="description" v-html="translate.description"></div>
89        </div>
90      </div>
91    </div>
92  </article>

```

Ilustración 27: Microdata de las páginas de artículos.

Anexo 3. Capturas de pantalla

Todas las capturas de este Anexo están sacadas del entorno de producción de la tienda. Asimismo, todas las imágenes de ilustraciones que aparecen en éstas han sido realizadas por el autor de este TFM y simulan el proceso completo de pago hasta llegar a la pasarela.




Ilustración 28: Página de inicio en esquema de colores oscuros



Ilustración 29: Página de inicio en esquema de colores claros.

Pequeña tienda de ejemplo Inicio Sobre mi Posts Eventos Redes sociales Productos EN


Lista de productos




Póster ganador concurso ilustración de arquitectura y comic 30 €
Firmado y numerado.



Cómic "Consecuencias" 2 €
Galardonado en el certamen Tebeox 2020, ahora descargable



Calendario de pared 2022 12.95 €
Calendario 2022 con 12 ilustraciones.



Encargo dibujo digital A3 600dpi 350 €
Dibujo personalizado A3 en formato digital a 600dpi. 3 pruebas de bocetos.

Ilustración 30: Lista de productos en castellano

Small shop example Home About me Blog posts Events Social media Products ES 

"Consequences" Comic

[Small shop example](#) // [Products](#) // "Consequences" Comic

awarded in the Tebeox 2020 contest, now available for download

Posted under Artworks

This product is currently available



"Consequences" Comic
1 x 2.00 € **2.00 €**

2.00 €

[View all items in cart](#)

Ilustración 31: Página de detalle de producto añadido al carrito.

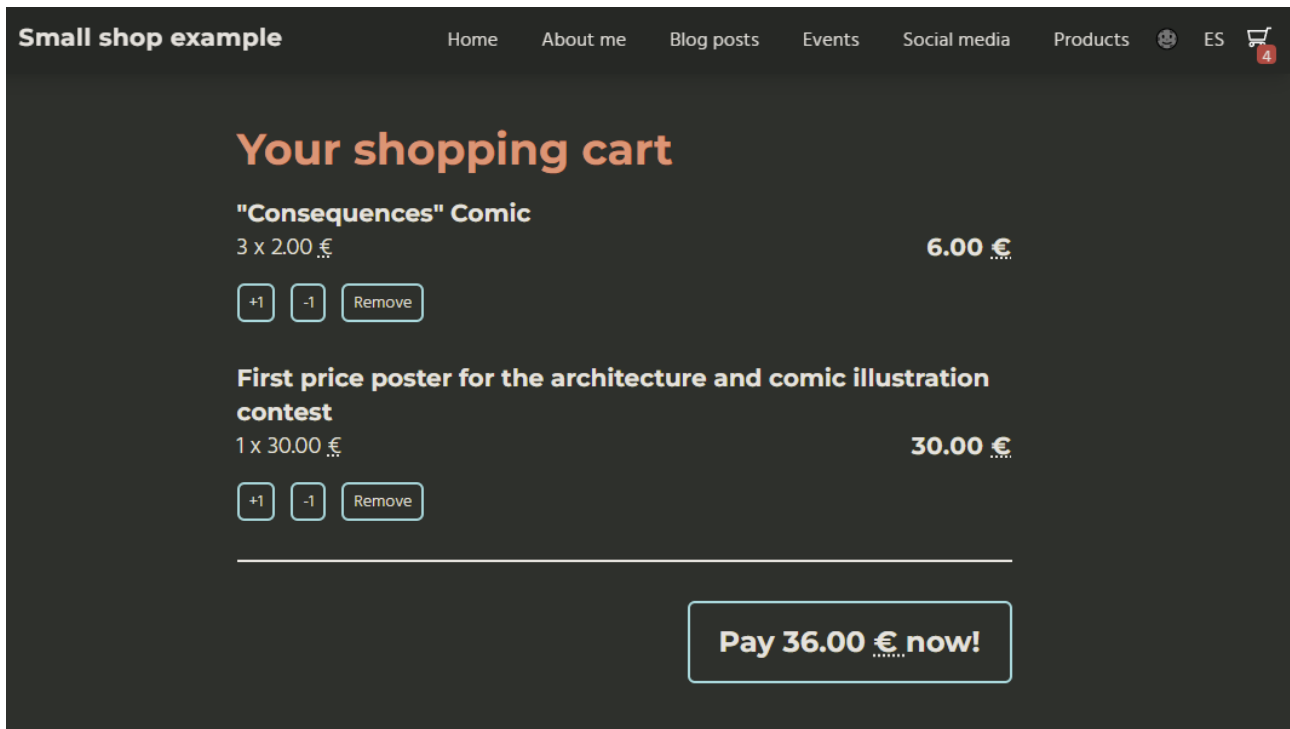


Ilustración 32: Carrito de la compra completo y preparado para pagar

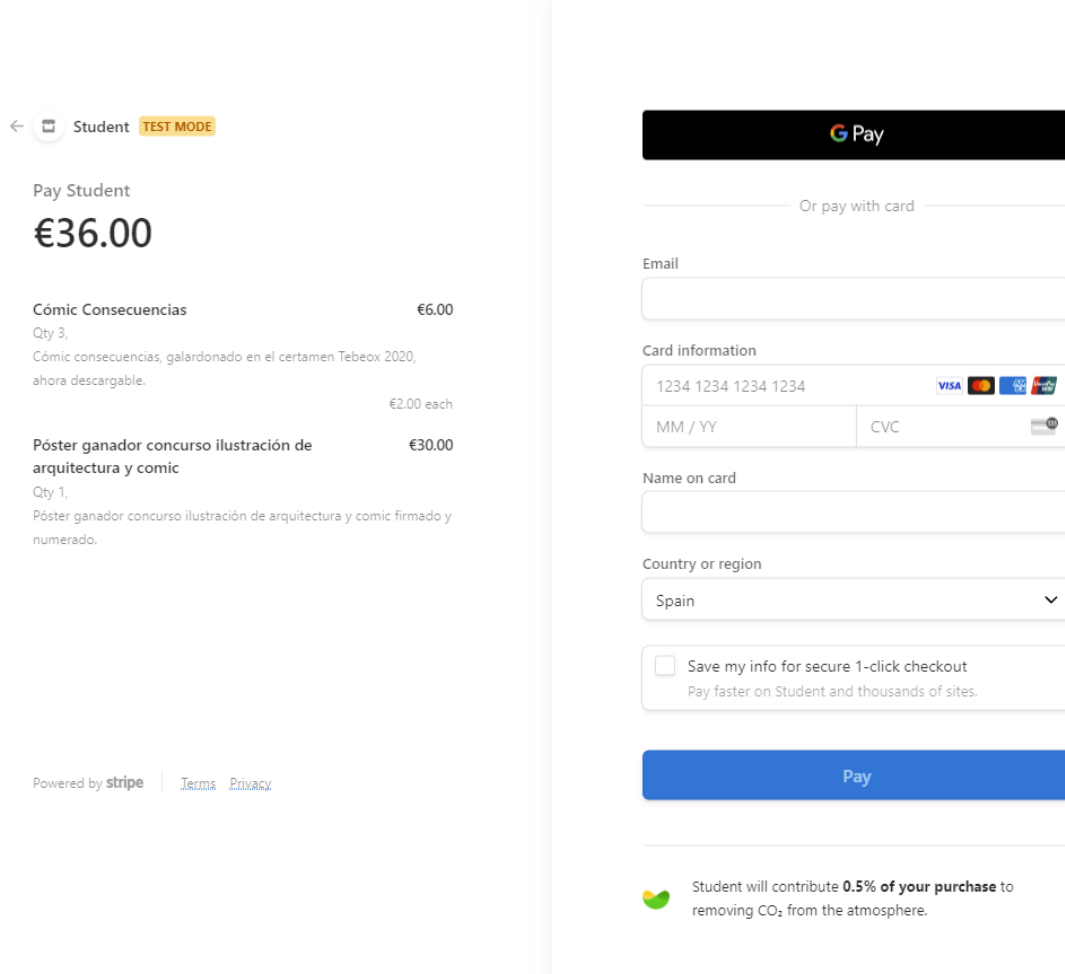


Ilustración 33: Pasarela de pago Stripe.

Anexo 4. Bibliografía

- Atlassian. (06 de 04 de 2021). *Gitflow Workflow | Atlassian Git Tutorial*. Obtenido de <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>
- Fernández, E. (2018). *Flujo de trabajo SCRUM - Platzi*. Obtenido de <https://platzi.com/clases/1307-scrum-2018/11889-flujo-de-trabajo-scrum/>
- Hall, L. (09 de junio de 2020). *Understanding folder / file structure in Nuxt*. Obtenido de <https://medium.com/@wearethreebears/understanding-folder-file-structure-in-nuxt-604ccc04a766>
- Kinsta. (06 de 2018). *Cuota de mercado de Wordpress*. Obtenido de <https://kinsta.com/es/cuota-de-mercado-de-wordpress/>
- We are social. (9 de febrero de 2022). *Digital Report España 2022*. Obtenido de 9 de cada 10 españoles usan las redes sociales y pasan casi 2 horas al día en ellas: <https://wearesocial.com/es/blog/2022/02/digital-report-espana-2022-nueve-de-cada-diez-espanoles-usan-las-redes-sociales-y-pasan-cerca-de-dos-horas-al-dia-en-ellas/>

Anexo 5. Vita

Actualmente

Desde 2018 trabajo para la empresa [Ticarum](#), dentro del grupo encargado del mantenimiento del Aula Virtual de la Universidad de Murcia, pasando a coordinarlo un poco más adelante. En un principio me encargaba de la actualización de la parte frontend de la plataforma, basada en JQuery, WebComponents y SCSS, aunque desde finales de 2019 me he encargado, como Tech Frontend Lead, de decidir la forma correcta de abordar la nueva infraestructura migrada desde los monolitos tradicionales de la empresa a una infraestructura de microservicios.

También formo parte del Core Team del proyecto LMS [Sakai](#), un proyecto de software libre con más de 10 años de recorrido.

Proyectos europeos

He trabajado en varios proyectos europeos durante mi trayectoria. Durante mi breve estancia en la empresa Answare Tech de Murcia colaboré en los proyectos europeos [iReact](#), [FloodServ](#) e [Itea 3](#).

Posteriormente, también he colaborado con el proyecto [Motiva](#) y, desde 2021, con "Advancing and Reassessing Oral Skills for English" o [Arose Project](#)

Desarrollador freelance

Durante casi 7 años, de 2012 a 2018, me dediqué a trabajar como trabajador freelance. Durante este tiempo colaboré con diversas empresas y clientes para llevar a cabo proyectos bastante diferentes.

Entre los que estoy más orgulloso puedo destacar el rediseño de los periódicos regionales de Vocento ([La Verdad](#) o [El correo](#) entre otros). Para este cliente también participé en los proyectos de Summer Beauty, Supersanos, el rediseño del [ABC móvil](#), nominada a los [Global Mobile Awards](#) o sus plataforma Guapabox, actualmente offline, y [Pidecita](#).

Otros de mis clientes a destacar podrían ser la Universidad Politécnica de Valencia, la Universitat de Lleida, el museo virtual de historia de la educación, Hefame o PCComponents.

Siéntete libre para preguntarme cualquier detalle sobre estos u otros proyectos de los que he formado parte.

Colaboraciones

En mis pocos ratos libres, además de dibujar e ilustrar cómics, me gusta colaborar con pequeños proyectos para ayudarles a ganar algo de visibilidad online. Las colaboraciones más recientes han sido con la [traductora y revisora Verónica Gil](#) y la web del videojuego indie [Pepo the cat](#).