UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO DE INGENIERO DE TELECOMUNICACIÓN

Análisis, diseño e implementación de un servicio de comercio electrónico en la nube

Adrián Blázquez León

2021

TRABAJO FIN DE MÁSTER

Título: Análisis, diseño e implementación de un servicio de comercio electrónico en la nube

Título (inglés): Analysis, design and implementation of an ecoomercer service in the cloud.

Autor: Adrián Blázquez León

Tutor: D. Santiago Pavón Gómez.

Departamento: Departamento de Ingeniería de Servicios Telemáticos

TRIBUNAL:

Presidente: D. el nombre del presidente

Vocal: D. el nombre del vocal

Secretario: D. el nombre del secretario

Suplente: D. el nombre del suplente

Fecha de lectura:

Calificación:

Resumen

En el contexto actual, se han identificado, por un lado, ciertas iniciativas que manifiestan que el software en el contexto empresarial no es accesible para las pequeñas empresas (debido a ello se plantean planes estatales de inversión como AceleraPyme). Por otro lado, que ciertas tecnologías, sobre todo el despliegue en la nube, permiten crear servicios baratos y eficientes. Así, dado una pyme, primero Breadfree (breadfree.es), una tienda de venta de pan sin gluten y luego Piedad León Art (piedadleon.art), una artista de arte hiperrealista a grafito, el objetivo de este proyecto es **analizar, diseñar e implementar un servicio de comercio electrónico en la nube**.

A continuación, se discuten tres puntos. Cada pregunta está relaciona con el método del "design thinking", una metodología para favorecer la innovación que consta de 4 fases.

La primera se llama **"what is?"** Y se elabora en la sección de **captura de requisitos y estado del arte**. Básicamente sobre la realidad, sobre el mercado, ¿qué usuario ha identificado qué necesidades? ¿Qué hay? Para ello primero se describen los procesos de negocio que se han identificado (caso de uso de tienda online y de gestor de inventarios) con el fin de plantear un marco de comparación de soluciones basado en métricas objetivas (una relacionada con cada caso de uso, una relacionada con la disponibilidad, otra con la seguridad y otra con el precio de la solución). A continuación, se describen las herramientas o soluciones existentes en el mercado en base a ese marco. Para entender las soluciones se realiza una taxonomía en función de los paradigmas de despliegue, desde on premise, IaaS, PaaS y SaaS. Se descarta "On premise" por precio, se identifican soluciones posibles como combinación de SaaS, por ejemplo, Shopify o Wix que permiten el caso de uso de tienda online, con Katana o Odoo que permiten la gestión de operaciones. Y se focaliza la investigación en despliegues IaaS o PaaS como "Container as a Service" que permiten desplegar un proyecto de código libre que permita ambos casos de uso.

En el siguiente apartado de **análisis y diseño**: se identifica el caso de estudio como resumen de la fase anterior, respondiendo así a la fase. **"what if?"**, y se plantea un diseño de posibles tecnologías. Hipotéticamente ¿qué es lo que busca la pyme? Una solución que permita estos dos casos de uso al menor precio posible. Se plantea un diagrama de casos de uso y varios diseños en cada fase de la sección posterior.

A continuación, en el apartado de **implementación** se desarrolla la evolución de la propuesta de soluciones, describiendo en cada artefacto el "what wows?" el qué de beneficio tiene cada artefacto en función de este marco de referencia propuesto.

- **Fase 0: comparativa despliegues: jenkinsX(eks, gke), cloudformation.** El objetivo de esta fase inicial no es implementar una solución entera, sino comprar en cuanto a la

parte de infraestructura las dos soluciones más extendidas IaaS, sobre todo para comparar precio: kubernetes y arquitecturas de cloud formation, resultando ser kubernetes la solución a elegir, si bien ambas son válidas y plantean un coste de 60€/mes, la gran ventaja de kubernetes frente a cloud formation es que no es una solución propietaria. Primero se realiza un prototipo de despliegue de servicio en alta disponibilidad, de un servicio con frontend y backend genérico en kubernetes, usando Jenkins X para implementar despliegue e integración continua sobre el repositorio de github. Por un lado, por otro lado, se plantea un despliegue de un sistema en cloud formation, definiendo las redes, los servidores y las bbdd.

- **Fase 1: FaaS shop:** en una segunda fase. Se diseña e implementa un artefacto lo más barato posible. Primero, se implementa un frontend básico en html y se encuentra lambda como el hosting más barato, pues el primer millón de peticiones se ofrece gratis. Luego, se plantea que para que el gestor de la tienda no sepa programar es necesario implementar más lógica. Para ello se implementa una demo de tienda serverless sobre lambda usando el stack serverless. Básicamente con la funcionalidad de un servicio de "todos" ampliado. Con una base de un frontend en angular y un backend en typescript también usando dynamo db. Este artefacto en fase Alpha no permite los casos de uso 1 y 2, pero cumple de por sí cumple la métrica de disponibilidad y la de precio, pero no la de seguridad.
- **Fase 2: saleor + gcp (cloud run + cloud sql):** en esta fase se desarrollan los distintos casos de uso que implementan esta "lógica" que se describe en el apartado anterior y se plantea un despliegue en un servicio Container as a Servicice, cloud run, en el que el primer millón de peticiones es gratis. Así, el coste del despliegue únicamente recae en la base de datos. Se despliega un proyecto de código libre llamado saleor, un frontend en react y un backend en Python. Se personaliza la tienda para breadfree usando figma como herramienta de prototipado. Se llega a la conclusión que es una solución que permite alta personalización en el frontend, pero le falta de integración de gestión de inventarios. Se lleva a fase beta, cumple 1 a medias porque no permite CMS pero no cumple el caso de uso 2, y no cumple las métricas de seguridad y disponibilidad para una métrica de precio aceptable.
- **Fase 3: odoo ce + aws (fargate + auroraserverless):** en una tercera fase, en cuanto a despliegue se encuentra una base de datos serverless en aws, pero resulta que los despliegues en aws a diferencia de gcp necesitan de un balanceador de carga 24/7 que ya por sí sólo consume 15€/mes. Así que, tras un planteamiento inicial, se replantea el despliegue en GCP. En cuanto al código se plantea modificar el proyecto de código libre más grande después de saleor, odoo, que no sólo incluye un módulo de tienda sino otro de gestor de inventarios. Este artefacto no pasa de la fase alpha.

- **Fase 4: odoo ce + gcp (cloud run + cloud sql) y vm:** en esta fase se despliega primero odoo ce en cloud run y cloud sql, de un coste de 10€/mes, pero se encuentra que odoo persiste en su memoria la sesión y cloud run son contenedores efímeros. De forma que se replantea el despliegue a un nodo vm para pasar a un futuro despliegue en kubernetes en alta disponibilidad. Se despliega, el código implementa los casos de uso, pero el presupuesto para un usuario es de 60€/mes, mayor que el coste de wix o Shopify. Es decir, en cuanto al marco, cumple con los casos de uso 1 y 2, y no cumple con las métricas de disponibilidad y seguridad para la métrica precio propuesta.
- **Fase 5: (wp + wc + plugins) + namecheap:** finalmente se plantea un despliegue de woocommerce en namecheap, buscando implementar los casos de uso que se han definido. Se llega a un coste de 4€/mes, es decir permite cumplir las 5 métricas.

Finalmente, en una fase de **conclusión** se desarrolla el "what works?" De los artefactos que se han elaborado qué funciona de cada uno. Se resume que de los artefactos que han llegado a la fase beta, los de la fase 2, 4 y 5, es el artefacto 5 el que mejor queda en la comparación de métricas: si bien un despliegue en kubernetes de 2 y 4 pueden presentar más grado de personalización que 5, 5 ofrece mejor resultado en el resto de las métricas a menor precio. 1 es una buena propuesta, pero desarrollarla implica elevado coste. Así, se destaca también que se han cumplido los objetivos se han evaluado 15 herramientas, discutido 15 stacks IaaS/PaaS distintos, trabajado con 4 proyectos de código libre, proponiendo 3 soluciones en fase beta.

Palabras clave

Nube, tienda online, Google cloud, Aws, woocommerce

Abstract

In the current context, certain initiatives have been identified that show that software in the business context is not accessible to small companies (due to this, state investment plans such as AceleraPyme are being proposed). On the other hand, that certain technologies, especially the deployment in the cloud, allow creating cheap and efficient services. Thus, given an SME, first Breadfree (breadfree.es), a shop selling gluten-free bread and then Piedad León Art (piedadleon.art), an artist of hyper-realistic graphite art, the objective of this project is to **analyze, design and implement an e-commerce service in the cloud**.

Three points are discussed below. Each question is related to the "design thinking" method, a methodology to promote innovation that consists of 4 phases.

The first is called **"what is?"** And it is elaborated in the section **of capture of requirements and state of the art**. Basicall, on the market, which user has identified what needs? To do this, the business processes that have been identified (use case of online store and inventory manager) are first described in order to propose a framework for comparison of solutions based on objective metrics (one related to each use case, one related to availability, another to security and another to the price of the solution). The tools or solutions on the market based on this framework are described below. To understand the solutions, a taxonomy is made based on the deployment paradigms, from on-premise, IaaS, PaaS and SaaS. On-premise is discarded by price, possible solutions are identified as a combination of SaaS, for example, Shopify or Wix that allow the use case of an online store, with Katana or Odoo that allow the management of operations. And the research is focused on IaaS or PaaS deployments such as “Container as a Service” that allow the deployment of an open-source project that allows both use cases.

In the following **analysis and design** section: the case study is identified as a summary of the previous phase, thus responding to the phase. "**What if**?", And a design of possible technologies is proposed. Hypothetically, what is the SME looking for? A solution that allows these two use cases at the lowest possible price. A use case diagram and various designs are discussed in each phase of the later section.

Next, in the **implementation** section, the evolution of the solution proposal is developed, describing the "what wows?" the benefit of each artifact based on this proposed frame of reference.

- **Phase 0: comparison of deployments: jenkinsX (eks, gke), cloudformation**. The objective of this initial phase is not to implement an entire solution, but to buy the two most widespread IaaS solutions in terms of infrastructure, especially to compare prices: kubernetes and cloud formation architectures, turning out to be kubernetes the solution to choose. Although both are

valid and cost € 60 / month, the great advantage of kubernetes over cloud formation is that it is not a proprietary solution. First, a high availability service deployment prototype is made, of a generic frontend and backend service in kubernetes, using Jenkins X to implement deployment and continuous integration on the github repository. On the one hand, on the other hand, a deployment of a system in cloud formation is proposed, defining the networks, servers and databases.

- Phase 1: FaaS shop: in a second phase. An artifact is designed and implemented as cheaply as possible. First, a basic frontend is implemented in html and lambda is found as the cheapest hosting, since the first million requests are offered for free. Then, it is suggested that for the store manager not to know how to program, it is necessary to implement more logic. To do this, a serverless store demo over lambda is implemented using the serverless stack. Basically, with the functionality of an extended "everyone" service. With a base of a frontend in angular and a backend in typescript also using dynamo db. This artifact in Alpha phase does not allow use cases 1 and 2, but it meets the availability and price metrics, but not the security metric.

- Phase 2: saleor + gcp (cloud run + cloud sql): in this phase the different use cases that implement this "logic" described in the previous section are developed and a deployment in a Container as a Servicice service is proposed, cloud run, in which the first million requests are free. Thus, the cost of deployment only falls on the database. An open-source project called saleor is deployed, a frontend in react and a backend in Python. The store is customized for breadfree using figma as a prototyping tool. It is concluded that it is a solution that allows high customization on the frontend but lacks inventory management integration. It is in beta, meets 1 half because it does not allow CMS but does not meet use case 2, and does not meet security and availability metrics for an acceptable price metric.

- Phase 3: odoo ce + aws (fargate + auroraserverless): in a third phase, in terms of deployment, there is a serverless database in aws, but it turns out that deployments in aws, unlike gcp, need a load balancer 24/7 that by itself only consumes € 15 / month. So, after some initial planning, the deployment on GCP is reconsidered. As for the code, it is proposed to modify the largest open source project after saleor, odoo, which not only includes a store module but also an inventory manager module. This artifact does not go beyond the alpha phase.

- Phase 4: odoo ce + gcp (cloud run + cloud sql) and vm: in this phase odoo ce is first deployed in cloud run and cloud sql, at a cost of € 10 / month, but it is found that odoo persists in its memory session and cloud run are ephemeral containers. So, the deployment to a vm node is reconsidered to move to a future deployment in high availability kubernetes. It is deployed, the code implements the use cases, but the budget for a user is € 60 / month, higher than the cost

of wix or Shopify. That is, in terms of the framework, it meets use cases 1 and 2, and does not meet the availability and security metrics for the proposed price metric.

- Phase 5: (wp + wc + plugins) + namecheap: finally, a woocommerce deployment is proposed in namecheap, seeking to implement the use cases that have been defined. It comes at a cost of € 4 / month, that is, it allows you to meet the 5 metrics.

Finally, in a conclusion phase, the **"what works?"** Of the artifacts that have been made, what works for each one. It is summarized that of the artifacts that have reached the beta phase, those of phase 2, 4 and 5, it is the artifact 5 that best remains in the metric comparison: although a deployment in kubernetes of 2 and 4 may present more degree of customization than 5, 5 offers a better result in the rest of the metrics at a lower price. 1 is a good proposal, but developing it involves high cost. Thus, it is also highlighted that the objectives have been met, 15 tools have been evaluated, 15 different IaaS / PaaS stacks have been discussed, 4 open-source projects have been worked on, and 3 solutions are proposed in the beta phase.

Keywords

Cloud, shop, Google cloud, Aws, woocommerce

Agradecimientos

La elaboración de este proyecto y sus artefactos no habrían sido posibles sin el apoyo de mi tutor Santiago Pavón, de Daniel Gómez-Bravo (de Breadfree), ni de Piedad León (de Piedad León Art) así como de mi familia y amigos. Quiero agradecérselo a ellos.

CONTENIDO

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

En el contexto actual, se han identificado, por un lado, ciertas iniciativas que manifiestan que el software en el contexto empresarial no es accesible para las pequeñas empresas (debido a ello se plantean planes estatales de inversión como AceleraPyme). Por otro lado, que ciertas tecnologías, sobre todo el despliegue en la nube, permiten crear servicios baratos y eficientes, más accesibles. Así, dado una pyme, el objetivo de este proyecto es **analizar, diseñar e implementar un servicio de comercio electrónico en la nube**.

A continuación, desarrollo estas dos motivaciones por las que tiene sentido elaborar este proyecto que he mencionado brevemente antes: la primera, parte de un análisis del entorno externo, qué software necesitan las empresas y qué segmento de estas se puede identificar. La segunda motivación, parte de un análisis de mi entorno interno, en relación con **desarrollar habilidades** que todo egresado del máster de Ingeniero de Telecomunicación especializado en el itinerario de Telemática ha aprendido.

Las empresas son entes que proveen de un producto o un servicio a sus clientes. Para ello, necesitan realizar una serie de tareas o actividades que se pueden modelar como procesos de negocio. Los sistemas de información pueden recoger y automatizar la información relativa a estos procesos de negocio. Estos procesos de negocio se pueden agrupar en distintas funciones (producción y logística, finanzas y contabilidad, recursos humanos o ventas y marketing) y distintos niveles. Por ello, se pueden clasificar los sistemas de información en el contexto empresarial, como “Enterprise Resource Planning” (ERP), o “Customer Resource Management” (CRM), o “Supply Chain Management” (SCM) o “Knowledge Management System” (KMS); como se observa en la siguiente figura obtenida de la documentación de la asignatura Sistemas de Información para la Gestión Empresarial. [1]

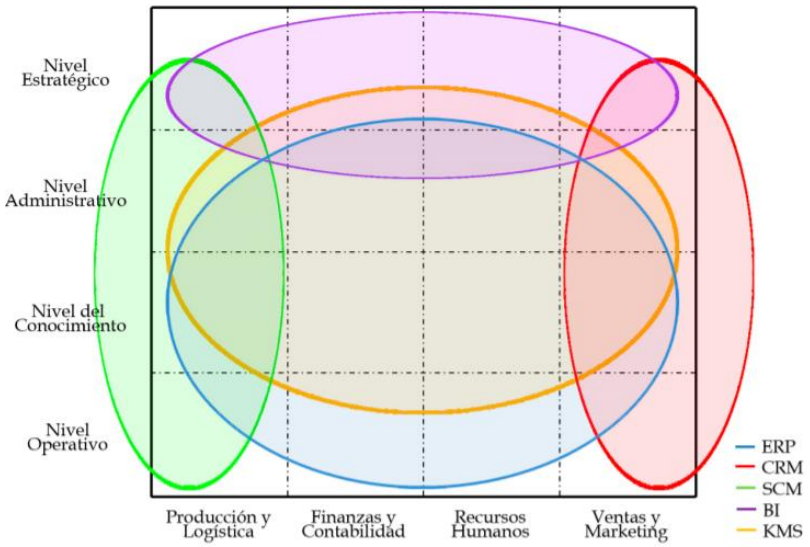


Ilustración 1. Sistema de Información para la Gestión Empresarial

Ahora bien, si se desea desarrollar un proyecto software para resolver las necesidades de una empresa, se sabe que se pueden proporcionar distintos sistemas de información, pero qué empresa se debe elegir. Se identifica que existen muchas empresas de tamaños y sectores distintos, pero que son las pequeñas las que o no cuentan con sistema de información o los que poseen son poco adecuados e ineficiente pero o carecen de recursos suficientes para plantearse este software o como estos sistema de información no son baratos, nunca han recibido una prueba de cómo pueden mejorar su negocio estos. Por ejemplo, este estudio en el que se entrevistan a 159 empresas del barrio de Vista Alegre muestra que sólo 34 de ellas poseen tienda online. [2] Pero, es que, además, este hecho no es local, sino el propio plan Acelera Pyme financiado con fondos europeos, plantea ofrecer recuros y oficina de digitalización a las PYMEs para poder realizar su transformación digital. [3]

La segunda motivación, se relaciona con la primera: poder ofrecer sistema de información a una empresa puede ser en este proceso de digitalización una habilidad demandada. En el máster, en concreto en el itinerario de telemática, se han aprendido ciertas técnicas que se pueden aplicar y desarrollar: se plantea implementar artefactos relacionados con metodologías de despliegue en la nube del estado del arte, para ampliar los conocimientos de nube iniciados en las asignaturas de Big Data fundamentos e infraestructuras o Blockchain fundamentos y arquitecturas, como son dockerizar servicios, desplegar en kubernetes o en Function as a Service. Por otro lado, desplegar un proyecto de código libre que emplee react como frontend y proponer cambios en él, para ampliar conocimientos de react aprendidos en Blockchain Desarrollo de Aplicaciones.

Estas dos motivaciones, hacen razonable plantear este proyecto como solución para dos pymes en concreto, una startup de venta de pan, breadfree, y una artista de dibujos hiperrealistas a grafito, Piedad León Art.

1.2 OBJETIVOS

A continuación, se plantean tres objetivos:

- El primero, analizar los requisitos de la pymes y comparar las soluciones y tecnologías del estado del arte. Es decir, analizar los procesos de negocio que necesita la pyme, y después plantear un marco de referencia basado en 5 métricas objetivas para comparar soluciones. Así, investigar soluciones y tecnologías y comrpararlas.
- El segundo objetivo es **diseñar** soluciones posibles en base a estas herramientas y arquitecturas, proponiendo diseños de soluciones posibles y piezas que las compondrían.
- El último objetivo es **implementar** artefactos que resuelvan el problema de estas pymes y luego discutir de qué manera son buenas soluciones.

2 ANÁLISIS

Hasta ahora, se ha descrito cuál es la motivación de este proyecto y sus objetivos. En esta sección, se describe cuál es la necesidad del cliente para articularla en requisitos. Para ello, primero se describen **los procesos de negocio** que se plantean para el cliente Piedad León Art, luego se recogen estos en **un diagrama de casos de uso** y finalmente se propone un **marco de comparación** de soluciones, basado métricas objetivas relacionadas con requisitos funcionales y no funcionales.

2.1 PROCESOS DE NEGOCIO:

Se listan las actividades que componen los procesos de negocio de este servicio minorista de venta:

I. Se crea arte

II. Se inventaria: asigna un código y precio y se publica en la web (1)

III. Un usuario realiza una compra

a. Comprueba qué producto quiere, con qué precio y para cuándo, y acepta tasa de envío

b. Realiza el pago y el trámite de compra rellenando la dirección (le llegan correos confirmando el pedido)

c. Aparece un pedido con una dirección.

d. Dependiendo de si es original o impreso.

IV. Cíclicamente se atienden pedidos: se comprueba el correo para encargos y los pedidos para impresos y originales

- Si llega un pedido de un impreso o se compra un original, se escanea este.
- Una vez se ha obtenido el lienzo se componen el producto con sus componentes y se cambia el estado ha completado para que llegue la notificación que avisa que va a ser enviado a Correos.

2.2 DIAGRAMA DE CASOS DE USO:

En el último sprint del proyecto, los casos de uso que se han analizado se podrían proponer como el siguiente diagrama de casos de uso. Dos casos de uso reflejados como epic, **tienda online** y **gestor de inventarios**. Dos grandes actores, el comprador y el gestor de la tienda, y casos de uso secundarios derivados de los casos de uso principales: de tienda online se identifica uno relacionado con que el comprador visualiza los productos del catálogo y pueda realizar el pago bajo la introducción de su domicilio. Además, se necesita que lleguen correos que informen al usuario del

seguimiento del pedido. Sobre este caso de uso principal el gestor de la tienda puede construir la web, es decir añadir o quitar uno u otro producto añadir algún mensaje de oferta. Respecto al caso de uso de gestor de operaciones se resalta el poder visualizar el stock. Esto es muy importante en el caso de la tienda online de la artista, en el que se prevé gran rotación de stock de embalaje y poco espacio de almacén, así que se propone como forma de automatizar la detección de falta de stock. Por último, el caso de uso de calculador de coste, en cuanto a que se necesita que en los pedidos se refleje tanto el coste del propio producto para realizar una estimación del coste, como del propio coste del envío en función del peso de los productos.

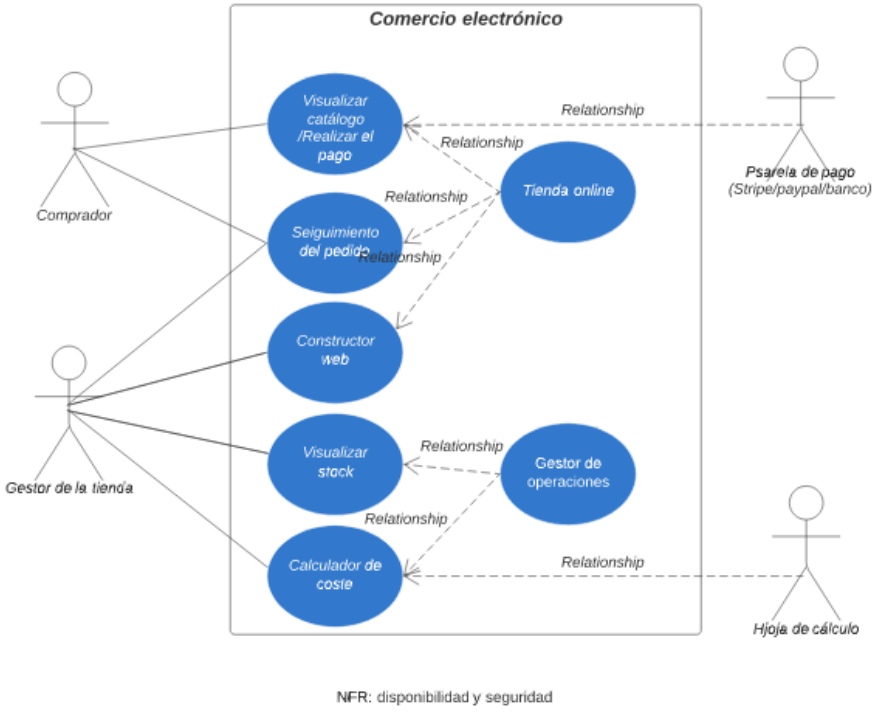


Ilustración 2. Diagrama de casos de uso

2.3 MARCO DE COMPARACIÓN:

Con el fin de comparar los distintos artefactos que se han desarrollado se plantean 5 métricas objetivas:

- Caso de uso de tienda online: número de subcasos de uso que plantea la solución. 0-ninguno, a 3.
- Caso de uso de gestor de operaciones: número de subcasos de uso que plantea la solución, desglosando el subcaso de calculadora de coste, como calculadora de coste de manufactura de productos y calculadora de coste de envío en función del peso.

- Usabilidad: en función de la métrica de performance de lighthouse sobre móvil en base a objetivo de 17 de la tienda online Juliane Berge(0-5, 0, 5-15, 1, 15-30 2 y en adelante un 3)
- -Seguridad: en función de la implementación sistemática L3 de distintos controles de seguridad inspirados en Magerit y Pilar.
    - Backup diaria de la información de negocio: base de datos e imágenes.
    - Creación de roles
    - Alerta sobre el control de acceso de administrador y 2FA
- Precio: relacionado con el precio de la solución, 0 si más de 60€/mes, 1 si menos de 40€/mes, 2 si menos de 20€/mes y 3 si menos de 10€/mes

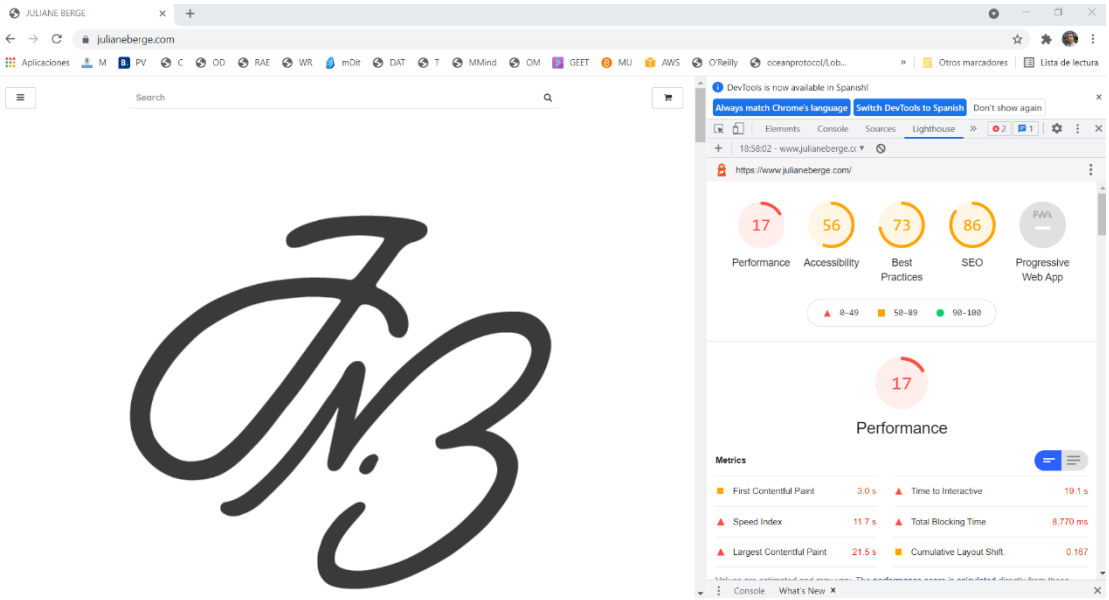


Ilustración 3. Test de ligthhouse sobre la tienda online de Juliane Berge, competidor de referencia del servicio

3 ESTADO DEL ARTE DE LAS SOLUCIONES Y LA TECNOLOGÍA

A continuación, se describe **el estado del arte, en cuanto a las herramientas o soluciones existentes en el mercado**. Para entender las soluciones primero se plantea que toda solución se compone de un **software** ejecutándose en una **infraestructura**. Así, existen tres posibilidades de plantear el proyecto software: desplegándolo desde 0, usar un proyecto de código libre o usar un software propietario en una infraestructura propietaria (lo que se conoce como Software como Servicio, SaaS). Por tanto, existirán dos tipos de soluciones las SaaS y los servicios "in house" o aquellos que se han desarrollado o usado código y se han desplegado sobre una infraestructura.

Entonces, en este apartado primero se comparan en base a este marco comparativo las soluciones SaaS que aparecen como más comunes en blogs y en comunidades de software para pymes, y a continuación se presentan las soluciones de código libre más comunes, comparándolas en este marco de referencia. Después, se realiza una taxonomía de los paradigmas de despliegue de infraestucutra, desde on premise, IaaS, PaaS y SaaS para comparar propuestas de stacks en la nube sobre el que desplegar el servicio.

3.1 SERVICIO SOFTWARE

3.1.1 Introducción

En esta sección se comparan soluciones en el mercado, soluciones de código propietario y libre desplegadas como SaaS sobre la infraestructura que se describirá en el apartado siguiente. Finalmente, se compran en tablas las soluciones bajo el marco comparativo para testear la validez de cada una.

3.1.2 Soluciones

3.1.2.1 Shopify

Es una plataforma que se define en su web como "the platform commerce is built on" (la plataforma de la que está hecha el comercio) [1] Se explora las funcionalidades planteadas en el marco comparativo y cumple el caso de uso de "tienda online" [2] [3], y es líder en muchos mercados por este servicio, como se observará en la ilustración posterior referente a Google Trends. En cuanto al caso de uso de gestor de operaciones, también lo cumple si bien Shopify permite visualizar stock [4] y calcular el precio del envío en función del peso [5] pero no permite crear productos compuestos

para gestionar el inventario o calcular el coste del producto en función de sus componentes; aunque existen plugins de pago que si pueden proveer de este servicio [6]

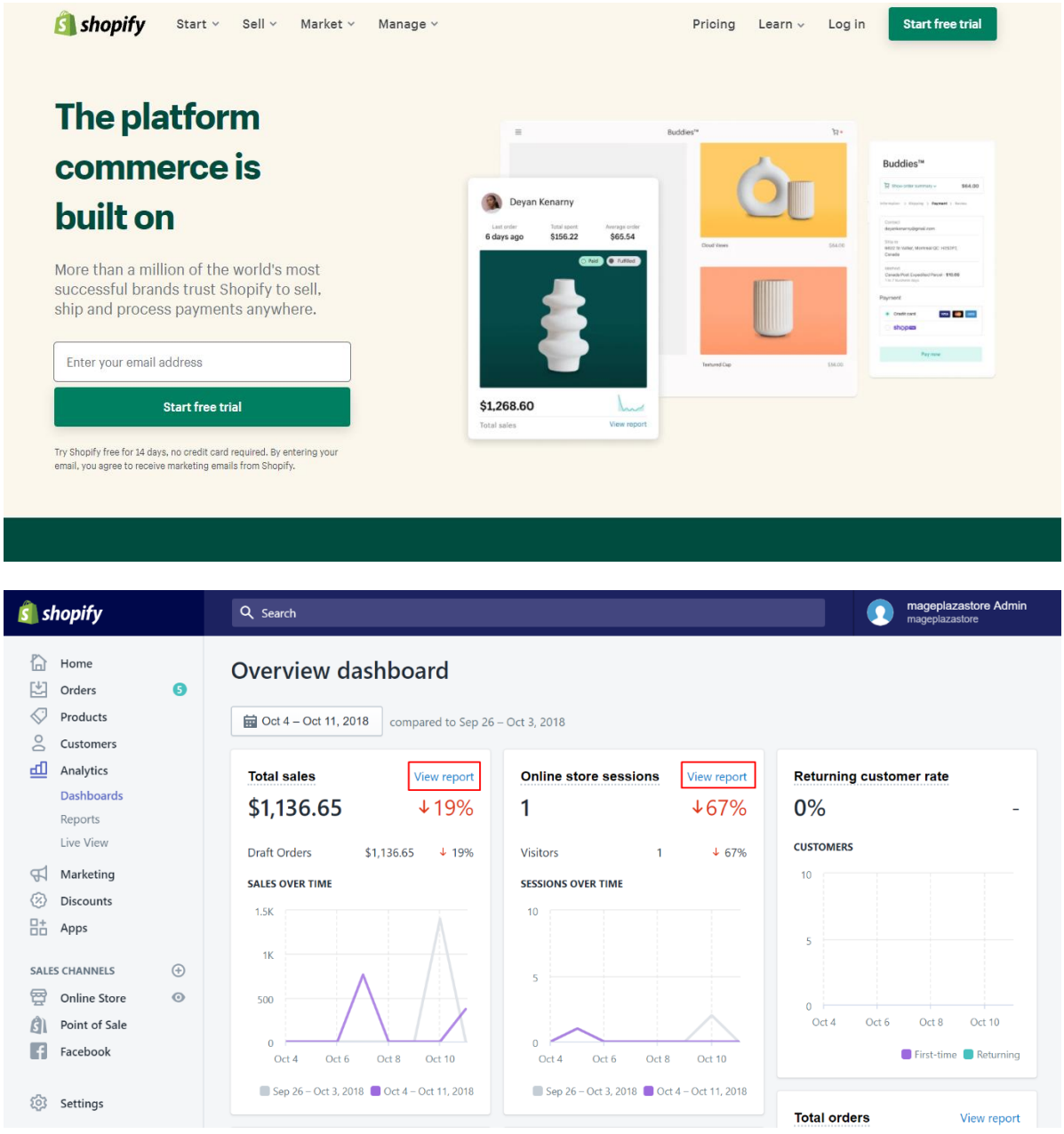


Ilustración 4 landing page y dashboard de Shopify

3.1.2.2 Wix

Otra plataforma de generación de ecommerce. Parecida en funcionalidad a Shopify pero con un precio menor de18€/mes. [7]

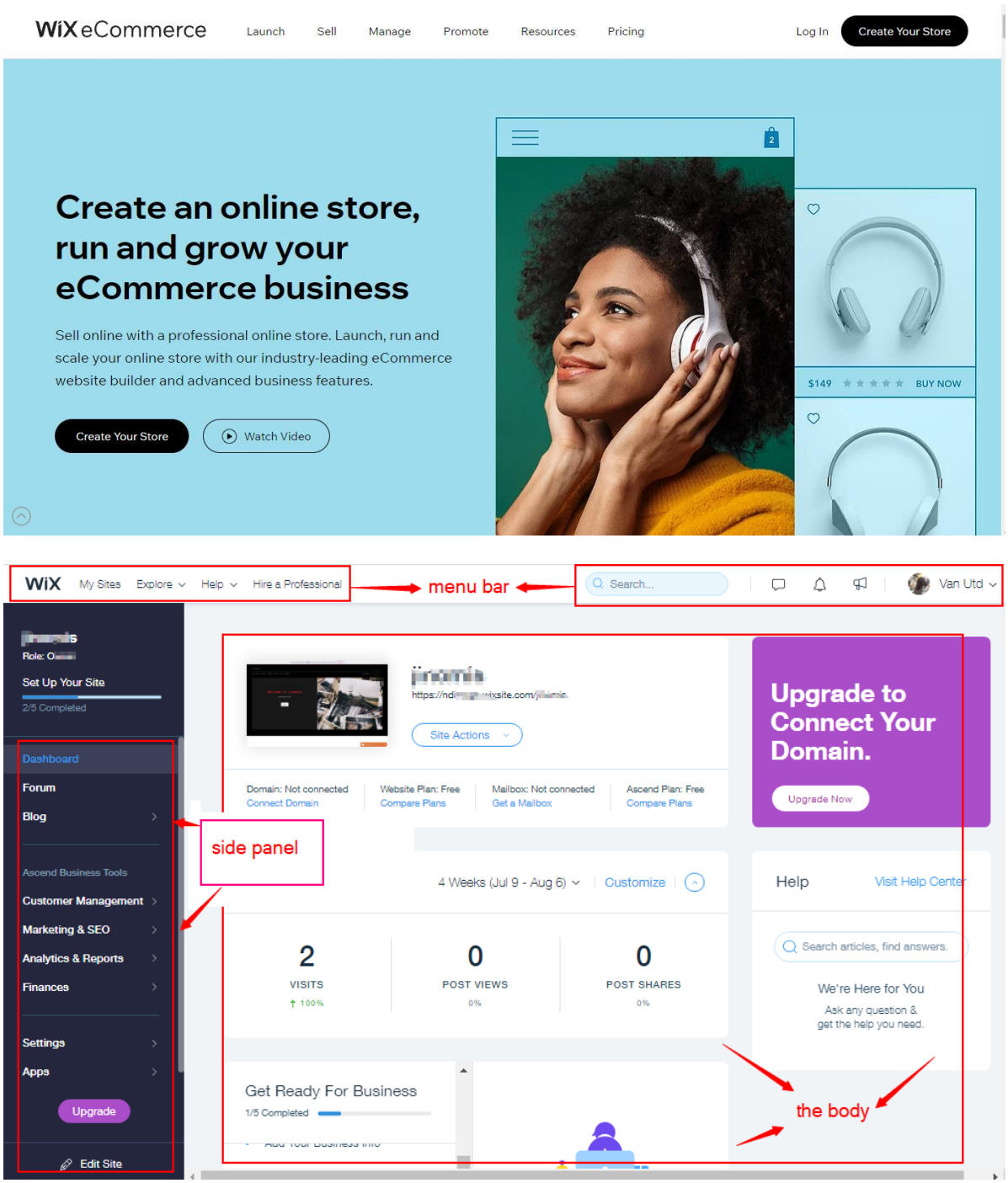


Ilustración 5 landing page y dashboard de wix ecommerce

3.1.2.3 Ecosistema Wordpress

"Wordpress is open source software you can use to create a beautiful website, blog or app" (wordpress es un software de código libre que puedes usar para crear una bonita página, blog o app). El ecosistema wordpress, primeramente centrado en creación de contenido, tras la publicación del plugin woocommerce, pudo ser usado para proveer de un canal online de ventas. Existen distintas

formas de desplegar wordpress, por ejemplo se presenta en la tabla wordpress.com que no permite plugins, y luego otros hosts como siteground, webempresa o namecheap que sí permiten. Como opinión personal se comenta que los plugins del ecosistema wordpress son muy profesionales, reusleven muchos problemas puede ser por la adopción en masa de estos, por un lado. Por otro lado, los hosting de wordpress son muy baratos. Por ejemplo, si se compara el ecosistema wordpress con el ecosistema odoo; si bien es cierto que odoo se publicita como una solución más profesional, es cierto que es una solución más integrada, en la que no hay que ir construyéndola plugin a plugin, pero, los PaaS de odoo son bastante más caros (véase apartados posteriores, de precio de namecheap 5€/mes a 40€/mes pro un despliegue en una máquina virtual). [8] [9]

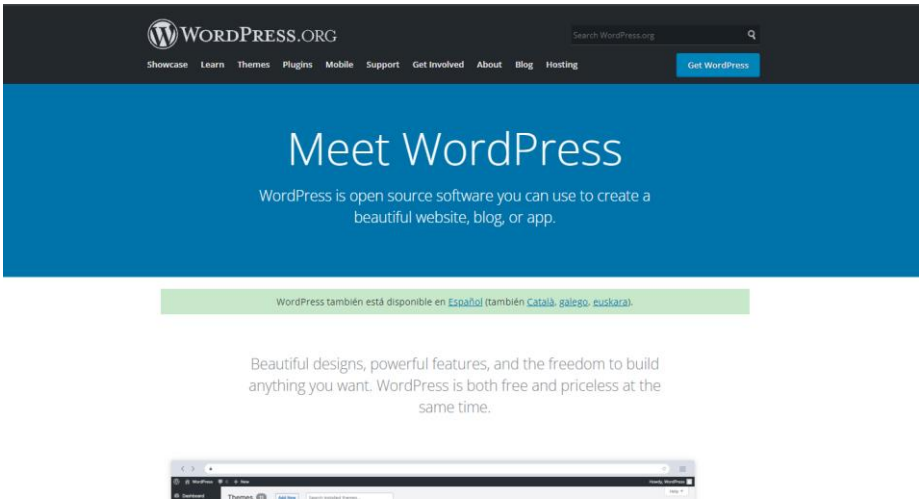


Ilustración 6 landing page de wordpress.org

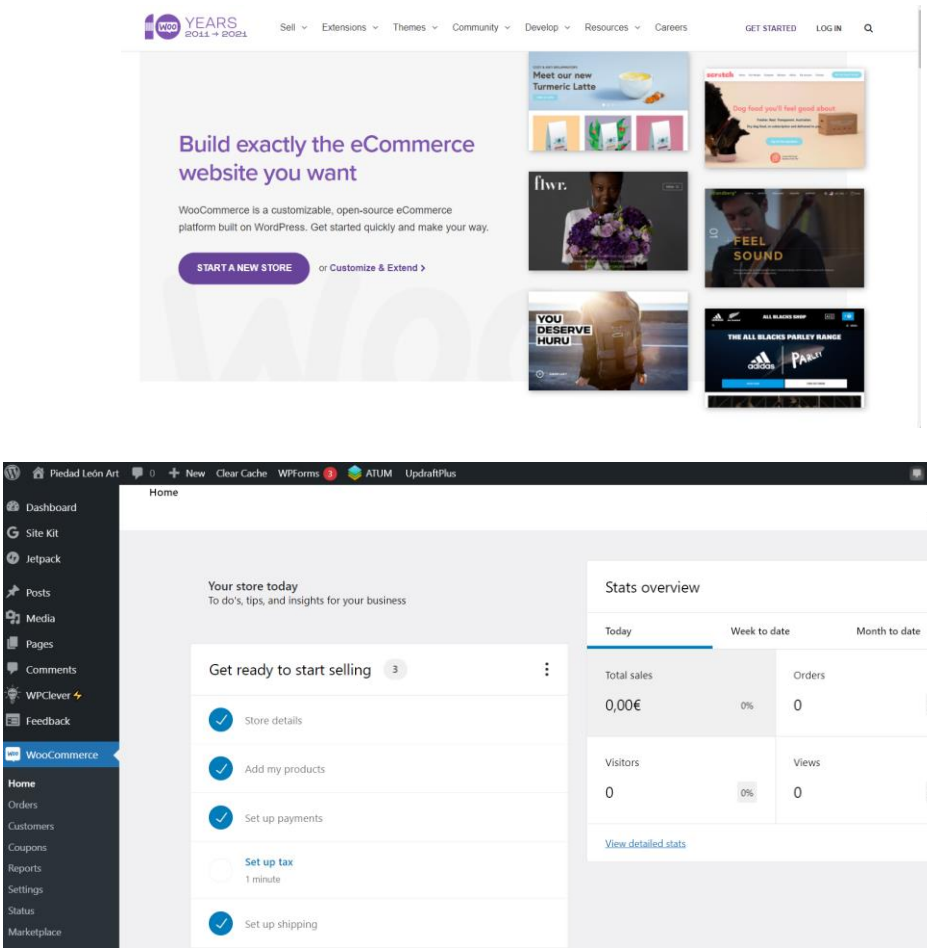


Ilustración 7 landing page y dashboard de woocommerce

3.1.2.4 Servicios para grandes empresas: Magento, Salesforce o Sap.

También en el mercado existen soluciones propietarias con destacado carácter profesional: son de elevado precio, pero presentan numerosas funcionalidades específicas y es capaz de proveer de servicio a gran cantidad de usuario. Este es el caso de Magento [10], Salesforce [11] o Sap [12].

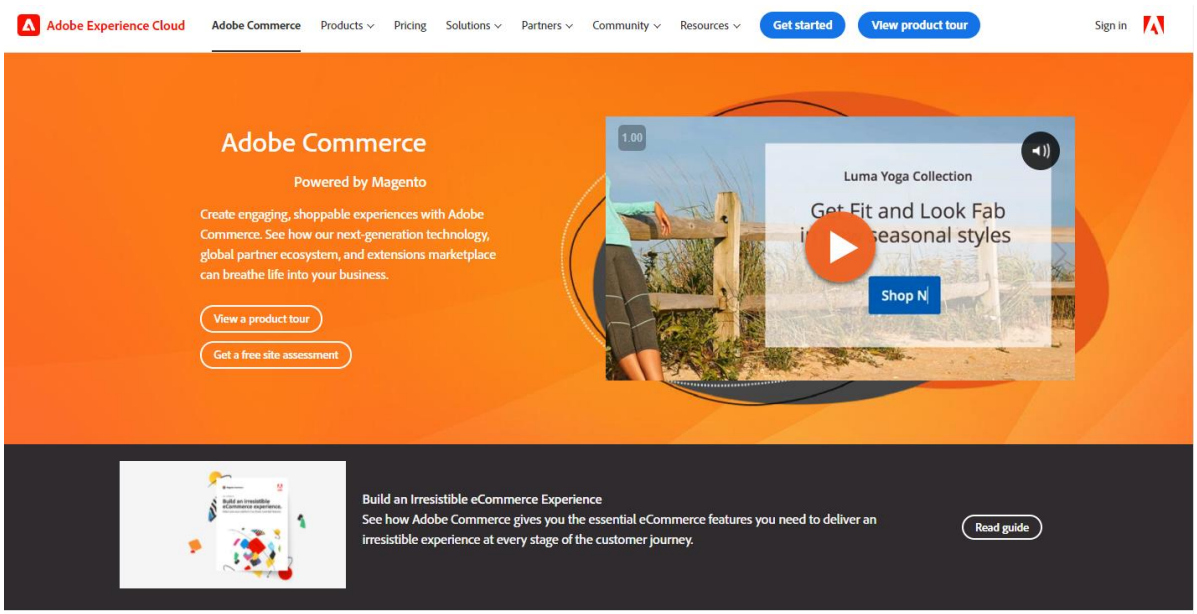


Ilustración 8 landing page de magento

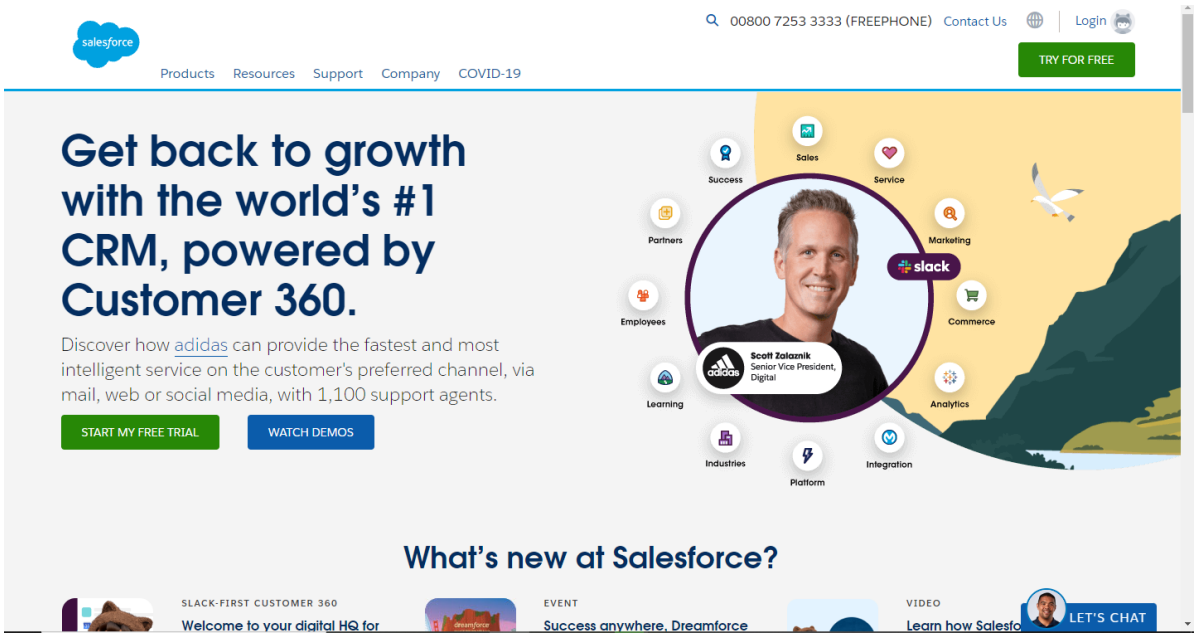


Ilustración 9 landing page de salesforce

Ilustración 10 landing page de SAP de servicios para PYMEs

3.1.2.5 Etsy

"Find things you'll love. Support independent sellers. Only on Etsy" [13](Encuentras cosas que amarás. Ayuda a vendedores independientes. Sólo en Etsy) Etsy es otra plataforma de creación de tiendas, como lo son servicios anteriores comparados, pero Etsy tiene la especialidad de ser un marketplace en el que puedes encontrar productos de más de un vendedor, por ejemplo, como amazon. Los vendedores deben pagar unos céntimos al mes por listar el anuncio de un producto en el marketplace. En relación con este marco de referencia para comprar, permite el caso de uso de tienda online pero no el de gestor de operaciones. Además, con un problema respecto al subcaso de visualizar catálogo, puesto que Etsy no permite visualizar per se el catálogo entero de un vendedor, sino que al comprador le aparecen productos de forma que los distintos vendedores compiten entre sí. [13] [14]

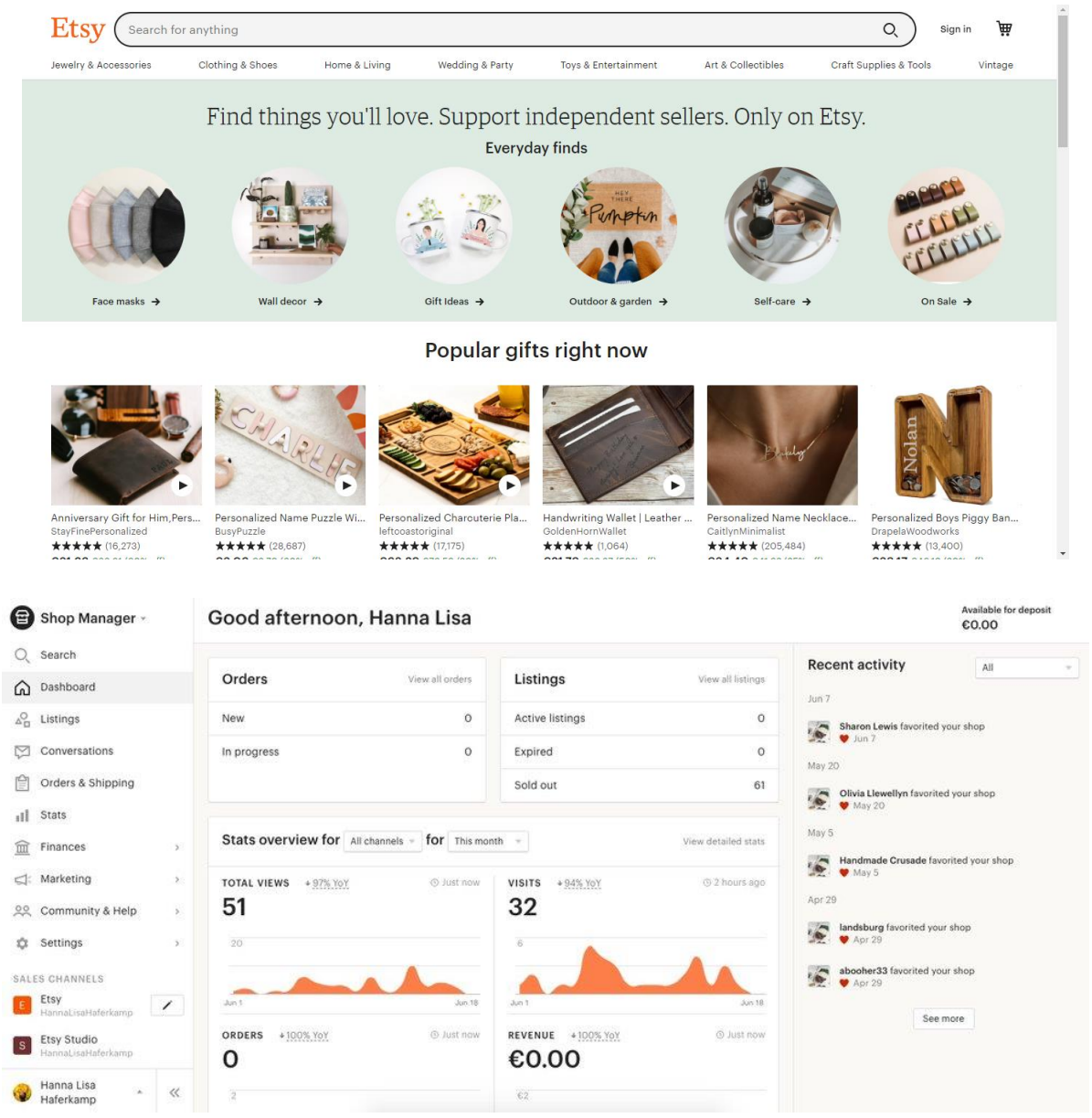


Ilustración 11 landing page y dashboard de Etsy

3.1.2.6 Ecosistema Odoo

"La única plataforma que necesitarás para manejar tu negocio: aplicaciones integradas, sencillas y adoradas por millones de usuarios felices". Se comparan distintas soluciones de código libre [15], las más desarrolladas parecen ser wordpress + woocommerce y odoo. Odoo se plantea como un servicio core más módulos pensados para ofrecer todos los servicios que necesita una empresa, entre ellos un módulo que cumple el caso de uso de tienda online y otro de gestor de inventario. Existe la versión SaaS con un precio entorno a los 50€/mes y la versión community (Odoo Community Edition o odoo CE), que se puede descargar en github.

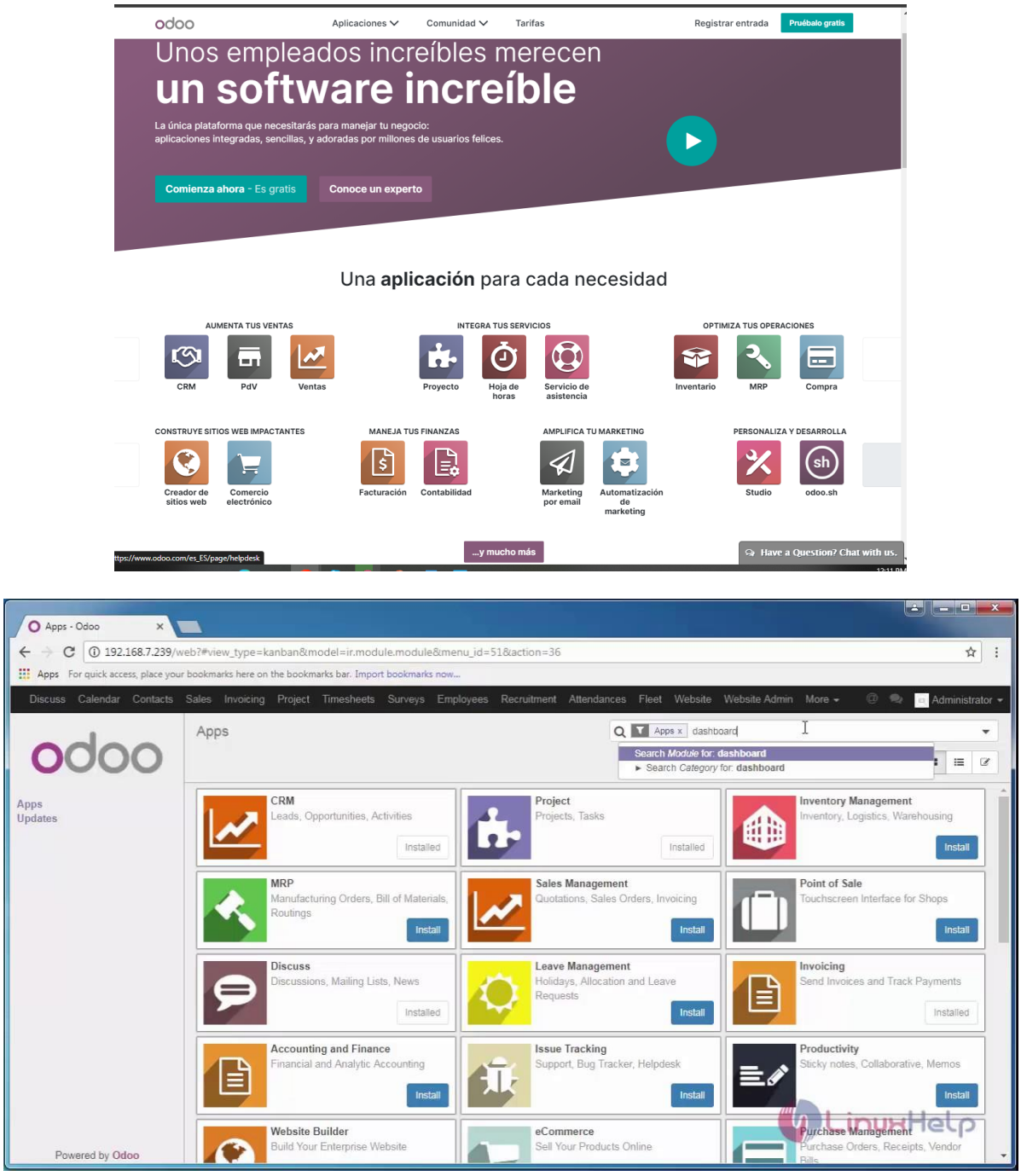


Ilustración 12. Landing page y dashboard de odoo

3.1.2.7 Saleor

Como software de código libre se ha introducido antes wordpress, basado en php. Existen otros proyectos que permiten tiendas online, se destacará el ecosistema odoo y en este apartado Saleor, un proyecto open source “headless” es decir, con un servicio dividido en tres microservicios, un dashboard en react, un storefront en react y un microservicio core en Python. [16]

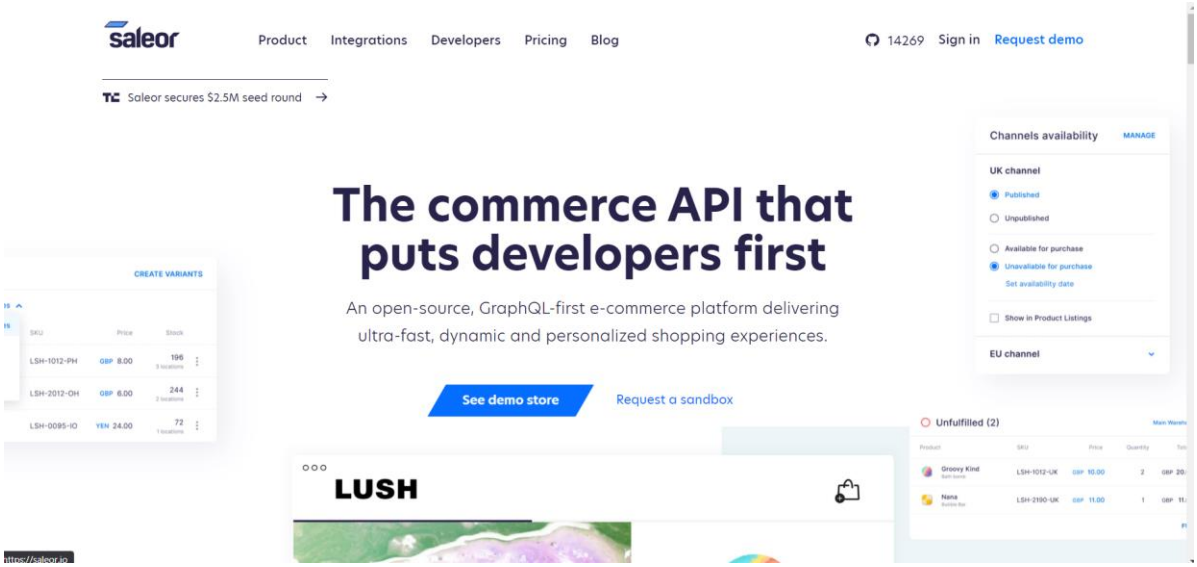
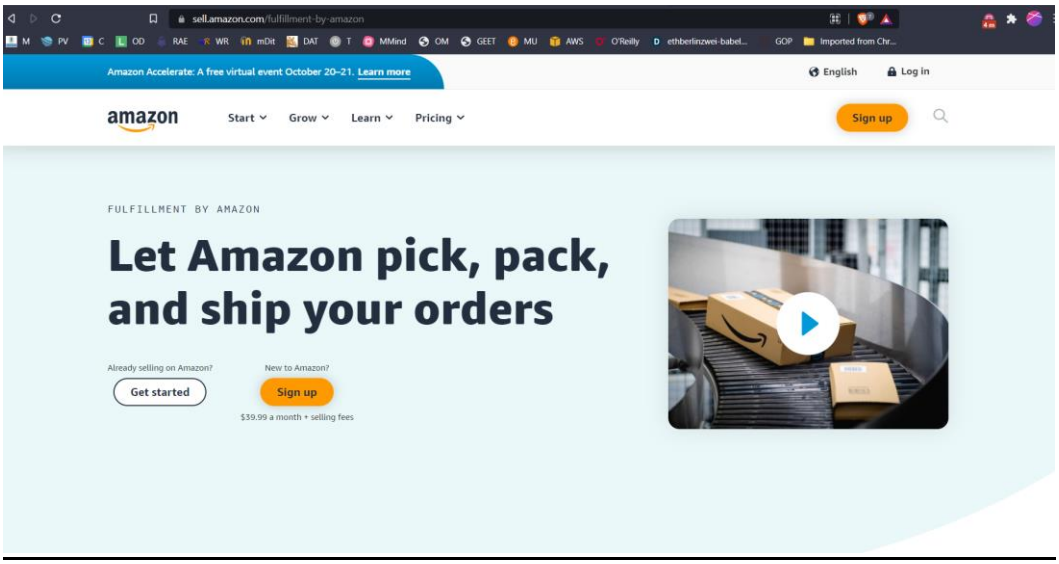


Ilustración 13 web de aterrizaje de saleor

3.1.2.8 Amazon FBA y Glovo

Ambos servicios son llamados de "fullfilment" o envío; permiten crear un catálogo de productos en las respectivas plataformas y además recogen el pedido, lo empaquetan y lo envían. En Amazon [17] la política de precios consiste en cobrar desde 1€ por producto vendido, hasta llegar al 50%. En Glovo [18] ocurre algo parecido, aunque los precios a priori aparecen como desconocidos, pues hace falta registrase como vendedor. Desde el punto de vista de estrategia de negocio, se destaca un hecho parecido a lo que ocurre con Etsy, son servicios que permiten ambos casos de uso desde un punto de vista funcional, pero "atan" la empresa a estas plataformas.



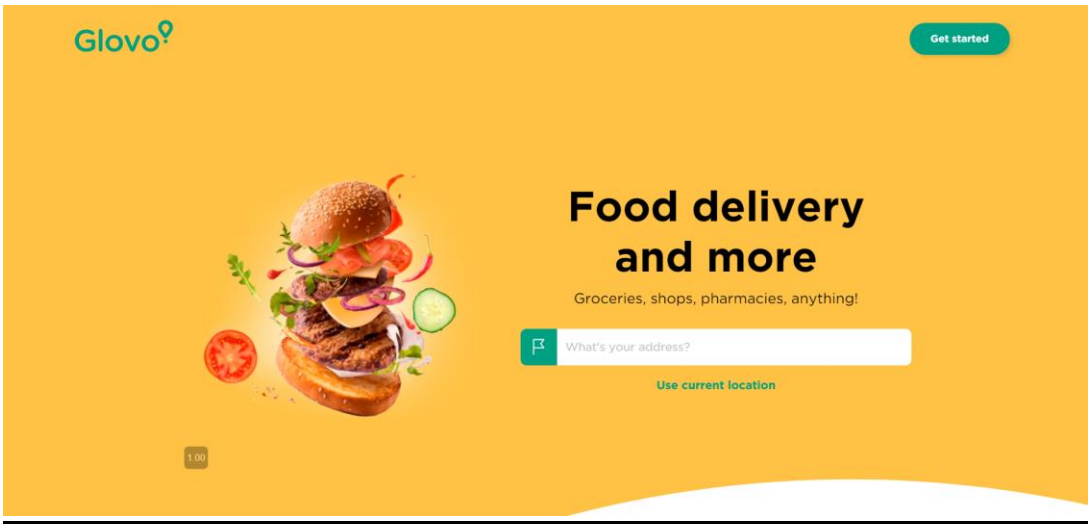
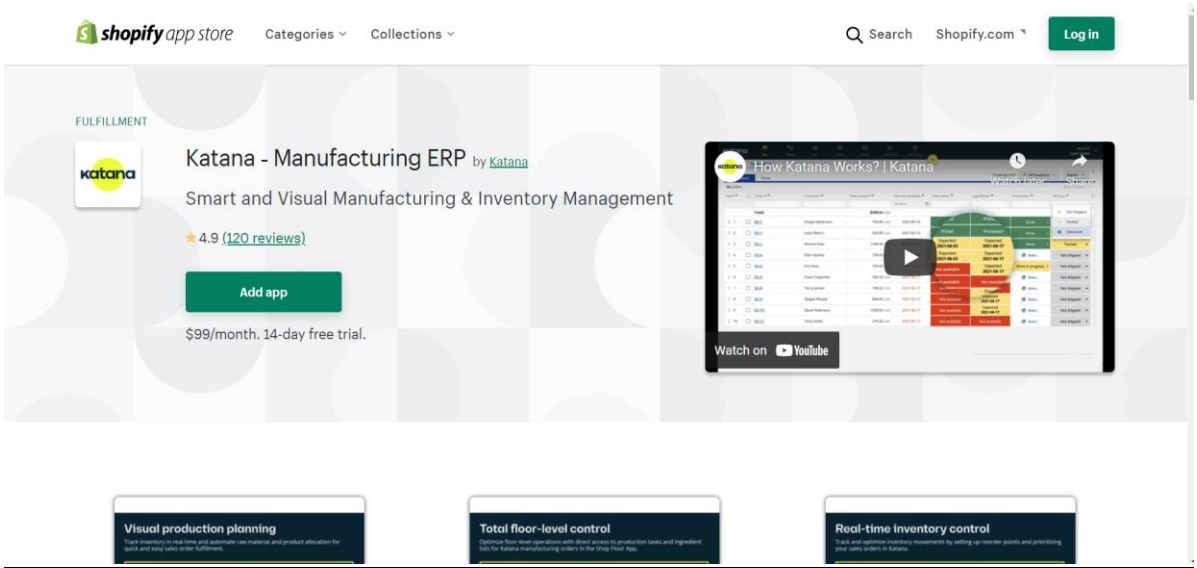


Ilustración 14 páginas de aterrizaje de amazon fba y glovo

3.1.2.9 Google sheets y katana

En este último apartado se describen dos herramientas. La primera usar una hoja de cálculo, Google sheets o Excel, que permita implementar el caso de uso "gestor de operaciones", para de forma manual, registrar los pedidos y el estado de estos para controlar el stock. Por otro lado, katana, es un servicio de gestor de inventarios que se integra con Shopify. [19]



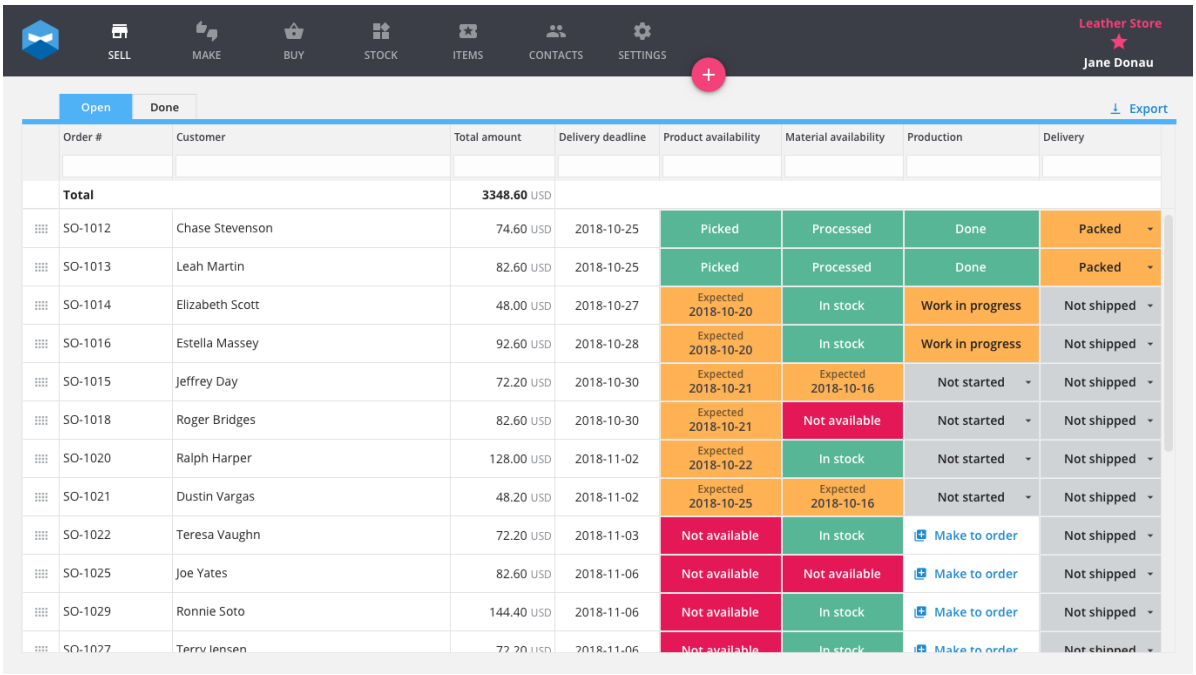


Ilustración 15 landing page y dashboard de katana

3.1.3 Comparativa

Finalmente, en la siguiente tabla, se presenta, en el eje horizontal las herramientas (Shopify, Wix, Wordpress, Magento, Salesforce, Sap, Odoo Saas, Etsy) y en vertical los procesos de negocio, identificados casos de uso y subcasos de uso que cumple, llamando 1, al caso de uso de tienda online y 2 al gestor de operaciones. De esta primera tabla se concluye que el espectro de soluciones es ancho. Junto con el siguiente gráfico de Google Trends se identifica que Shopify y Wix son las herramientas más usadas. Además, se presenta una tercera tabla en la que se compran otros servicios SaaS más ligado con el caso de uso 2 (Amazon, Glovo, Google sheet, Katana) de gestor de operaciones y con un llamado caso de uso 3 ligado a la gestión del marketing.

Por otro lado, se comparan herramientas de código libre. En general el coste directo de una solución con ambos casos de uso es menor, por ejemplo, de esos 50€/mes de odoo SaaS a40€/mes de odoo CE en compute engine de GCP. Después de discutir estas tablas se describen brevemente ciertas soluciones: Shopify, Wix, el ecosistema wordpress, Magento, Salesforce, Sap, Etsy, el ecosistema Odoo, Amazon, Glovo, Google Sheet y Katana.

Tabla 1. Comparativa SaaS [20]

	1 Shopify	Wix	Wordpress.com	Magento	Gumroad	Salesforce	Sap	Odoo Saas	Etsy 1
1.1.	x	x		x	x	x	x	x	x
1.2.	x(aftership)	x		x	x	x	x	x	
1.3.	x	x	x	x	x	x	x	x	x
2.1.		x		x		x	x	x	
2.2.				x		x	x		
€/mes	30	17.5	8	2000	10	25	16.000€	100	0.2€/producto/mes.1

	1 Saleor(crun+csql)	OddoCE(VM+csql)	WP+WC+NC+plugins 1
1.1.	x	x	x
1.2.	x	x	x
1.3.	x	x	x
2.1.		x	x
2.2.		x	x
2.3.		x	x
€/mes	15€/mes	40$/mes => 480$	60$ anuales

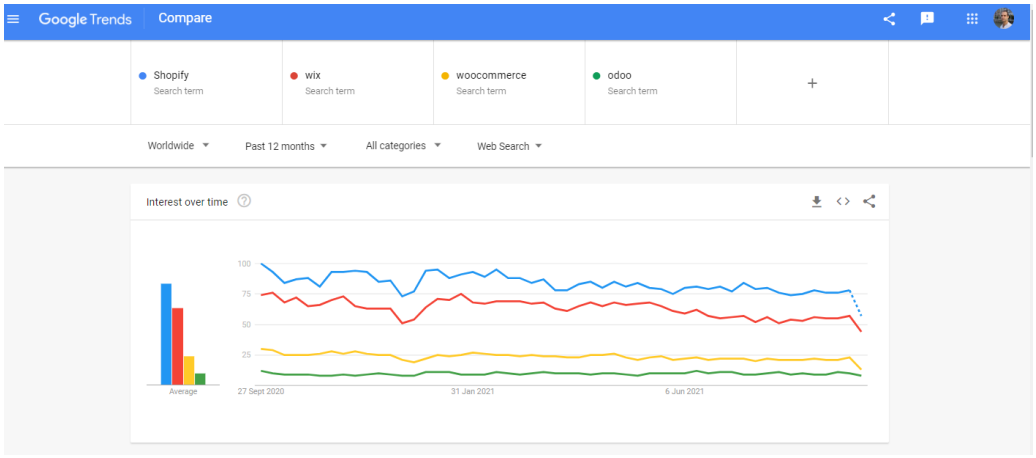


Ilustración 16. Comparativa Google Trends Shopify, Wix, Woocommerce y Odoo

Tabla 2. Comparativa SaaS de casos de uso 2 y 3

	2 Amazon	Glovo	G sheet	Katana 2	3 ConvertKit	MailChimp	Klaviyo	Hootsuite 3
1.1.	x				x	x		
1.2.		x				x	x	
2.1.	x							
2.2.				x			x	x
2.3.				x			x	x
3.1.				x			x	x
3.2.				x			x	x
Precio del servicio	2. 0 hasta 40 artículos 40€/mes	200€/mes	0€/mes	100€/mes .2	3. 30€/mes	15€/mes	30€/mes	40€/mes .3
Comentario	2. Completo, alto precio a largo	Completo, alto precio a corto	2	2 .2	3. Falta 2	3	3	3 .3

3.2 INFRAESTUCTURA

3.2.1 Introducción

En esta sección se discute cuál es el coste de desplegar una solución en función del tipo de despliegue. Para ello, primero se introducen qué tipos de despliegues existen, y se introducen y describen los nombres de las tecnologías del estado del arte clasificándolas en esta, para, a continuación, que se comparen tipos de despliegue genéricos y después ciertos stacks IaaS y PaaS.

Primero se introduce al lector qué tipos de despliegue se contemplan presentándole una taxonomía clásica del entorno de los despliegues en la nube. Como se observa en la figura siguiente. se realiza la clasificación entre tipos de despliegue: On premise, IaaS (Infraestructura como Servicio), PaaS (Plataforma como Servicio) y SaaS (Software como Servicio).

Ilustración 17 comparativa de tipos de despligue

3.2.2 Soluciones

3.2.2.1 On premise

"On premise" implica que todos los elementos del servicio estén gestionados por la empresa. En el caso de este servicio de tiendas online se puede plantear desplegar en una máquina local siempre encendida, con la ventaja que ofrece el control sobre toda la estructura, pero se resaltan los inconvenientes de falta de escalabilidad del servicio que queda ligado a la capacidad del ordenador donde está desplegado el servicio. Además de falta de seguridad, puesto que, a diferencia de un servicio en la nube, un posible agujero de seguridad comprometería la red local. Finalmente, se encuentra el inconveniente del coste, puesto que se un ordenador siempre debería estar encendido consumiendo electricidad. Por estas razones se descarta plantear este tipo de despliegue, si bien se conocen formas de hacer despliegues "on premise" comerciales, como usar "ngork" [21] para hacer un túnel a localhost.

3.2.2.2 IaaS (Infraestructura como servicio) y PaaS (Plataforma como servicio)

Se clasifican como servicios "IaaS" a aquellos en los que el proveedor cede los servidores, la red, la virtualización y el almacenaje y el cliente sólo debe proveer el software hacia "arriba" en la pila desde el sistema operativo. Por otro lado, se denomina a un servicio PaaS si el cliente se le proporciona además un sistema operativo, middleware y un runtime.

A continuación, se introducen distintas tecnologías IaaS y PaaS del estado del arte. Nótese que como la diferencia entre IaaS y PaaS es muy sutil se incluyen estas tecnologías en este apartado.

Primero, se describe quiénes son los actores en el mercado de estas tecnologías. AWS fue la primera que triunfó al empezar a ofrecer estos servicios IaaS/PaaS de forma masiva. Se aprovecha este apartado para ilustrar brevemente cuál es el panorama respecto a proveedores de nube, líderes en servicios IaaS/PaaS. Se observará en la gráfica como los proveedores que más recaudan son AWS y Azure, pero también se contemplan otros proveedores. Se adjunta otra imagen para describir que esto es debido a que estos proveedores de nube no sólo ofrecen servicios de IaaS/PaaS, sino que consiguen beneficios ofreciendo otros tipos de servicios.

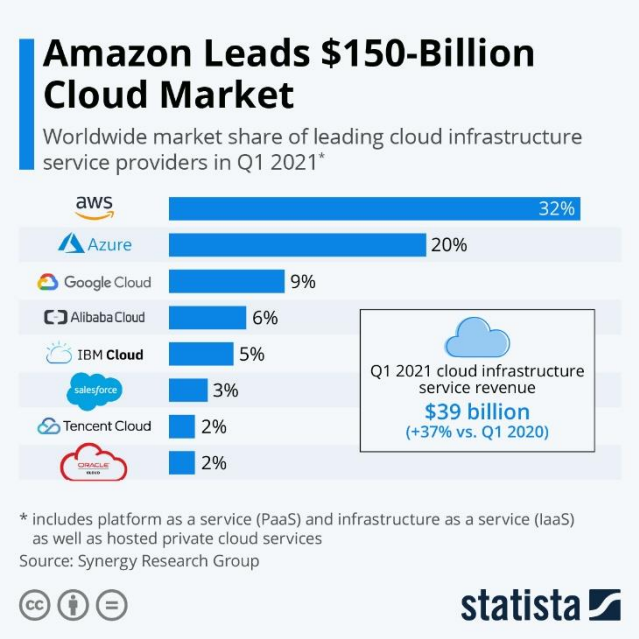


Ilustración 18 comparación de proveedores de cloud públicas por IaaS y PaaS [22]

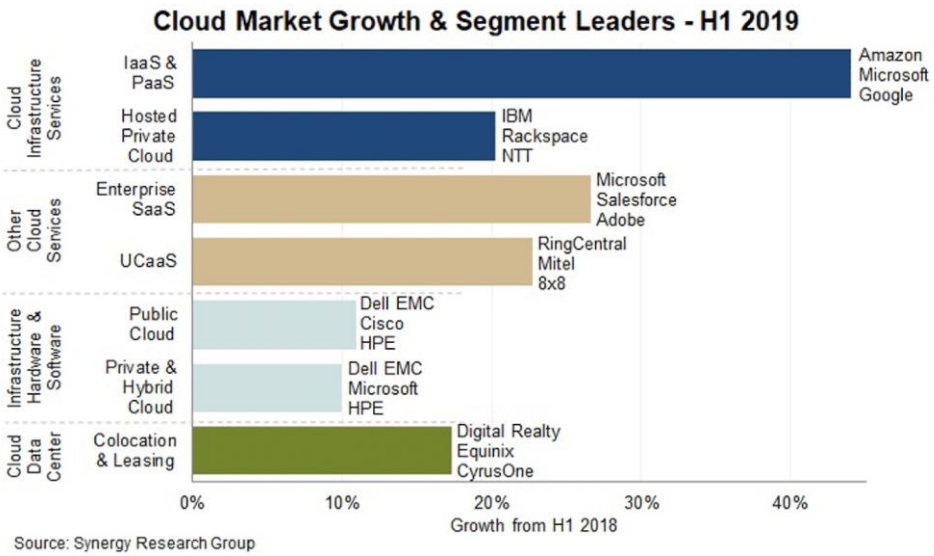


Ilustración 19 crecimiento del mercado y líderes del segmento h1 2019 [23]

En esta sección, se ha introducido el concepto de IaaS y PaaS se ha descrito brevemente la situación de los proveedores de cloud pública que ofrecen IaaS. Finalmente se van a introducir distintas tecnologías de despliegue IaaS/PaaS para que más adelante en la comparación, el lector las conozca.

- Máquinas virtuales en la nube: (en AWS el servicio se llama EC2 y en GCP compute engine) ofrecen distintos tipos de máquinas virtuales en función de su potencia de procesador y tamaño de la ram por ejemplo. Es el servicio más puro IaaS teóricamente.

- Servicios de automatización de despliegue: el único servicio interoperable entre todos los grandes proveedores de nube es terraform. En AWS se usa cloud formation, que permite mediante un yaml, describir servicios de máquinas virtuales (ec2), red (vpcs) y almacenamiento, para desplegar una infraestructura.

- Kubernetes: es el servicio de orquestación de contendores en alta disponibilidad standard de facto del mercado. Cada proveedor de nube ofrece su propio servicio en base a este, GCP(GKE), AWS(EKS) o Azure (AKS); servicios a los que a veces se les denomina Kubernetes as a service (o kubernetes como servicio).

- Container as a Service (contenedores como servicio): otra tecnología clave, muy de moda ahora; se le llama a un servicio IaaS de CaaS o contenedor como servicio si el usuario no tiene que desplegar ningún orquestador de contenedores, solamente elegir el sistema operativo y el código y desplegarlo en un contenedor, que sólo gastará créditos si es usado. A continuación, en la comparación se observa como este servicio es interesante para las Pymes clientes y se estudian si son servicios CaaS distintos servicios de distintos proveedores. Desde AWS (fargate), GCP(cloud run) o Azure (ACI).

- Bases de datos: otra tecnología o servicio importante son los servicios de base de datos como servicio de los proveedores de nube, que ofrecen servicios de base de datos ya instalados y con dashboard para su fácil mantenimiento. Servicios como son Cloud Sql de GCP o Aurora DB de AWS. Estos son servicios relacionales, también existen servicios de otras bases de datos no relacionales como pueda ser Dynamo DB de AWS o Cloud Spanner de GCP.

- Function as a Service: es una tecnología bastante novedosa que consiste en ofrecer al cliente un runtime especial para el que si el cliente implementa su código el proveedor de nube sólo le facturará los milisegundos de invocación de la función. El primer servicio que apareció en el mercado fue lambda de AWS pero el resto de principales hiperescalares también ofrecen ahora FaaS. Nótese, que con el find de organizar los servicios serverless se usa el proyecto

"serverless" que permite añadir una capa extra de software para estandarizar el código escrito del servicio sobre el que se ejecuta en nubes distintas.

- Hosting: el último tipo de tecnología que se desea describir son los hosting, los que mejor se clasificarían como PaaS. Existen muy diversos tipos de hosting y se ha trabajado en este proyecto con varios de ellos: desde algunos que admiten código genérico, como son Netlify o Heroku, o beanstalk de AWS, o hosting sobre todo enfocados por ejemplo a wordpress, como son sitegroud, webempresa o namecheap easy wp.

- De almacenaje: los sistemas gestores de bases de datos, sobre todo los relacionales, están pensados para almacenar información estructurada. Se plantea otro tipo de servicio sobre todo orientado a almacenar una información no tan estructurada y de mayor tamaño que la que se suele almacenar en las bases de datos. Por ejemplo, información relativa a backups de la base de datos a imágenes o a contenidos de software como plugins. Hay muchos servicios de este tipo en el mercado, cada hiperescalar cuenta con el suyo, por ejemplo el de Amazon Web Services (AWS) Simple Storage Service (S3). También hay servicios de almacenaje más específicos, por ejemplo el de Cloudinary para imágenes.

3.2.2.3 SaaS

El último tipo de despliegue se le llama software como servicio y se le clasifica a un servicio como tal, cuando el proveedor del servicio lo único que tiene es introducir los datos, pues el resto de la pila se ofrece. Todas las herramientas propietarias comparadas en el ejemplo anterior son SaaS.

3.2.3 Comparativa

Finalmente, se comparan distintos stacks de soluciones en la nube (IaaS/PaaS) con la premisa de poder encontrar una solución con un coste en la nube más barato que los 20€/mes que de media cuesta en 2021 un servicio SaaS. Básicamente las soluciones Caas, sobre los proveedores de los que se disponen créditos (Azure, GCP y AWS) puesto que las soluciones basadas en kubernetes exigen de un número elevado de usuarios para que empiecen a ser rentables. Se busca servicio de Container as a Service con el objetivo de pagar 10€ por cada servicio compartiendo base de datos.

Tabla 3. Comparativa CaaS

CaaSstack/prop	Compatibilidad	Cost (€/mes)
Azure:ACI+AzPGSQL	No psql svless	?
AWS:Fargate+AuroraSvl	Y	min 23
GCP:CloudRun+CloudSql	Y	min 10
Alicloud:ECI+AsparaDB	Y	?
IBM	No psql svless	?
CRun+AuroraSvl	Y	?
Clouding	N	?
Digital Ocean	Y	min 50
Linode	N, sólo k8s	?
AWS:AppRunner+AuroraSvl	Y	min 56 (15 lb + una isntacia de fargate y de db siempre corriendo)

Una interesante reflexión personal es presentar los créditos que por un alumno puede conseguir a junio de 2021, pues en parte a esto dependerá la elección del servicio de despliegue. Tras este estudio se concluye que son AWS y GCP los que a priori ofrece mayor tiempo de prueba

Azure => 100$/12 months

AWS => créditos de alumno upm 50$ 4/30/2022 => 5 meses

Google=> 40$ 15 Oct 21, 50$ 1 Oct 21 => 5 meses

Alibaba => ?

A continuación, se comparan distintas tecnologías entre estos tipos de despliegue sobre el marco de referencia propuesto en función a las necesidades de los clientes.

De la comparativa de herramientas anterior, se concluye que el coste de un SaaS que casa con los procesos de negocio puede ser de 20€/mes/cliente sólo del caso de uso 1 y como 100e/m del caso de uso 2. Además, si se desea ofrecer este servicio a más de un cliente no es escalable, que es lo que se compara en la tabla. La opción de encontrar un software y desplegarlo parece que puede ser más barata en este sentido. A continuación, se comparan las soluciones que se englobarían en el rango. Básicamente la conclusión de la siguiente comparativa, es que son las soluciones Kubernetes as a Service (básicamente servicios nube de este software de gestión de microservicios de alta disponibilidad) o CaaS (Contenedores como servicio), las que ofrecen mayor escalabilidad a un coste elevado de 60€/mes mínimo, las VM (máquinas virtuales o nodos, en la terminología de elementos de la nube) no son una alternativa por si solas, porque no escalan a no ser que se plantee un “scale group” (grupo de escalado, que define qué máquinas crear si se supera un límite en el rendimiento de determinados nodos)), que es lo que se plantea. Las FaaS (function as a service) son las soluciones más baratas en coste, sólo consumen por el tiempo de procesamiento, pero en el

mercado no se encuentra una solución “open-source” software que ofrezca implemente los casos de uso así que se debería desarrollar una solución particular.

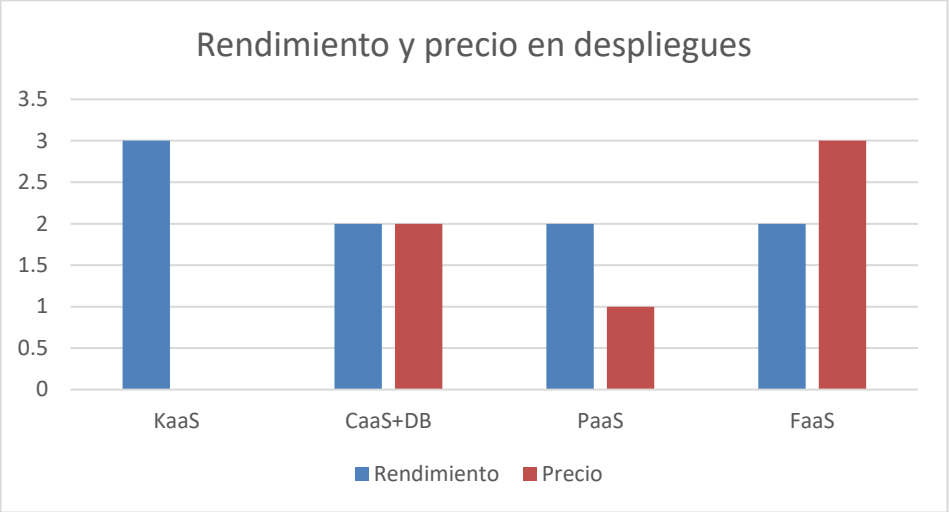


Ilustración 20. Gráfica comparativa de rendimiento y precio.

Tabla 4. Comparativa paradigmas de despliegue

type/prop	Scalability	Cost	Comment
Vm	Not scalable	Depends	
KaaS	Y	?	hay que estudar qué opción pemirte que se apague durante la noche, más coste de operación
CaaS	Y	?	depende de qué opción, cloud run, fargate o eci están bien, el problema está con la bbdd
FaaS	Y	0	El método más barato para adaptar el coste a la demanda, pero ncesita de reescribir el código enteor para usar las funciones
PaaS	N	0	PaaS como DonDominio o Heroku son ampliamente usadas pero restringen a la plataforma: Heroku unos costes altos y DonDominioo usar un sistema en PHP

4 DISEÑO.

En las secciones anteriores se ha descrito la parte del “what is” del proyecto. Qué procesos de negocio se han identificado y que herramientas y tecnologías existen en el mercado. En el siguiente apartado de **diseño**, se responde a **“what if?”**, y se plantean diagramas de arquitectura de distintos artefactos detallado cómo se combinan las distintas tecnologías..

Hipotéticamente ¿qué es lo que busca la pyme? Una solución que permita estos dos casos de uso al menor precio posible. Básicamente, como ventajas hipotéticas se ha planteado una solución más barata que las SaaS, una copia de Shopify y wix que sólo ofrecen ese caso de uso de tienda, con gestor de inventarios.

En cuanto al diseño en cada una de las fases se platea un diseño en cuanto al diagrama de arquitectura. Brevemente se recoge cada diseño a continuación, para luego describir en cada artefacto cómo se ha implementado cada diseño.

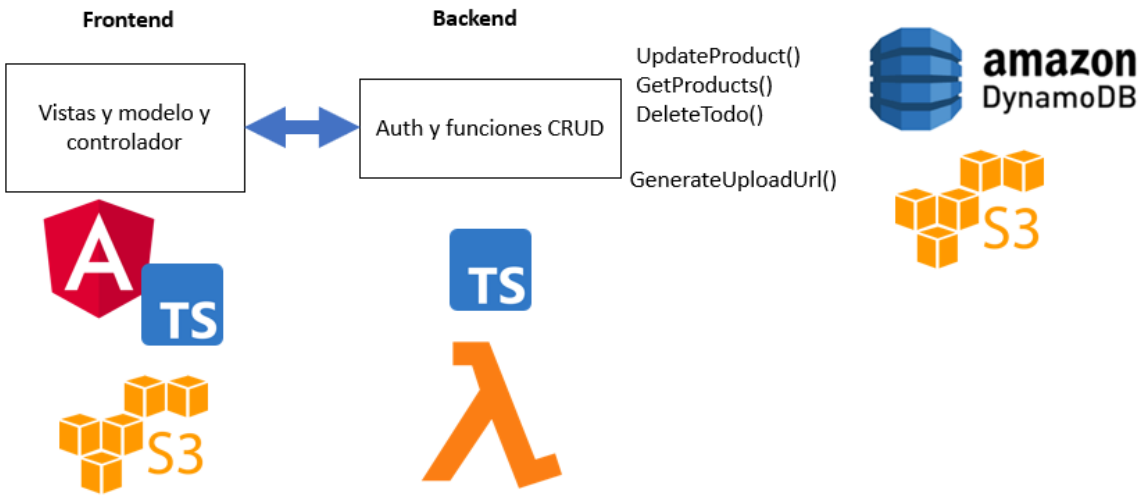


Ilustración 21. Arquitectura de artefacto 1, shop serverless. Desloges del backend y del frontend

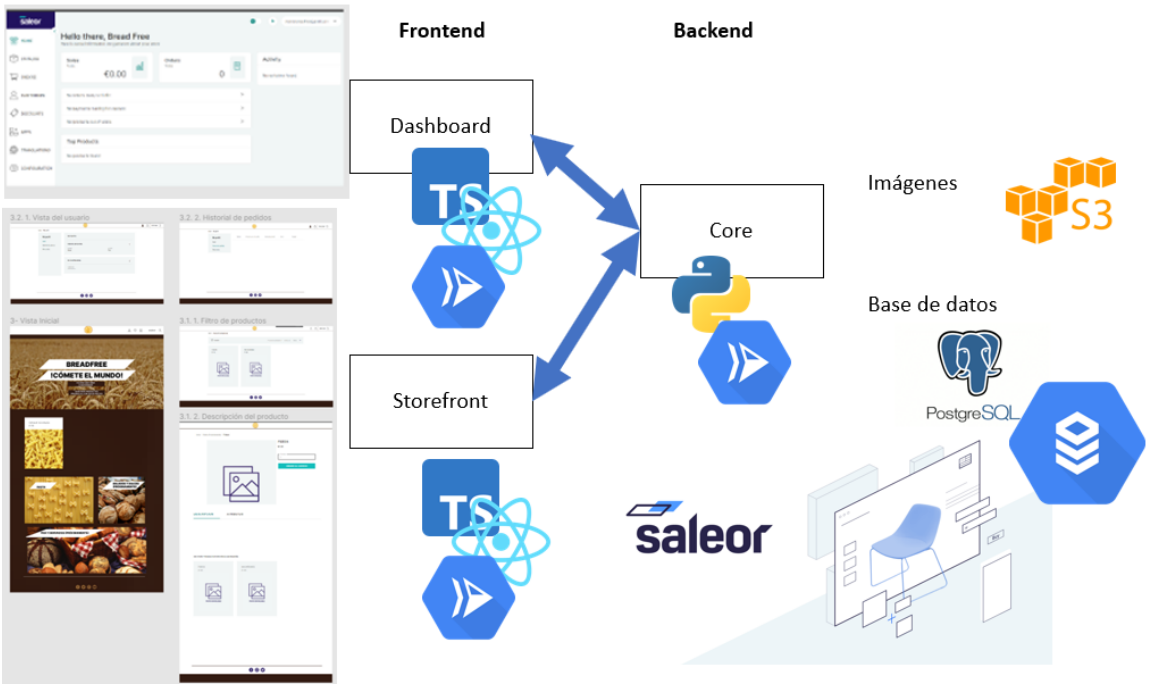


Ilustración 22. Arquitectura del despliegue de saleor, artefacto 2

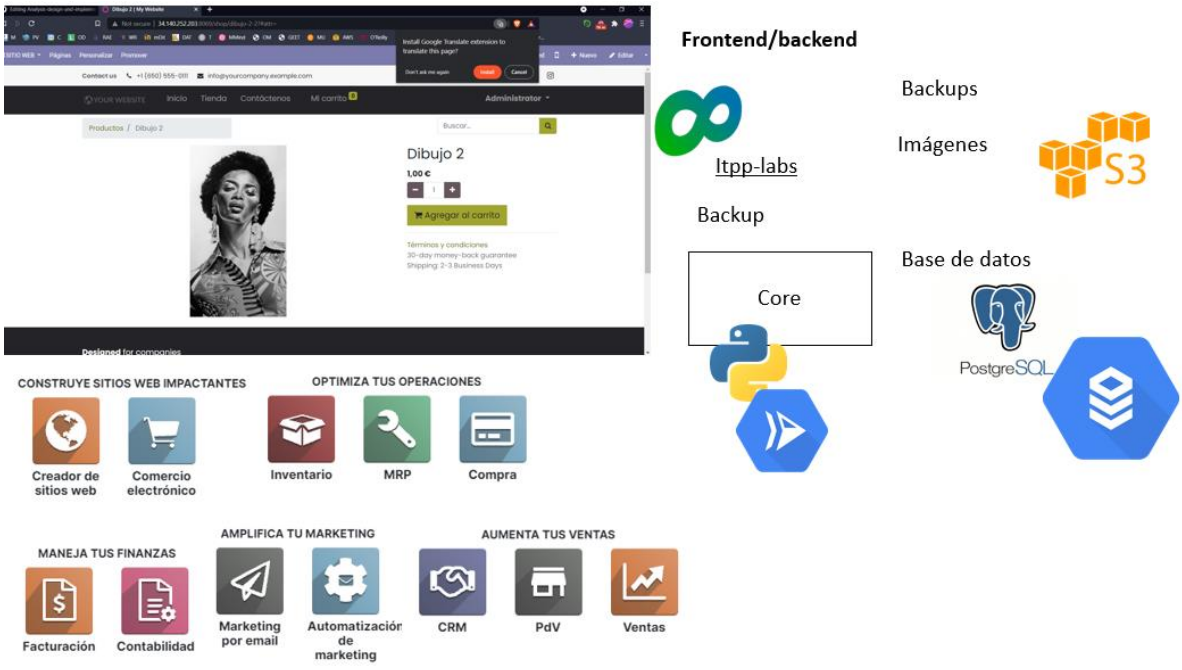


Ilustración 23. Arquitectura del artefacto 3 con cloud run

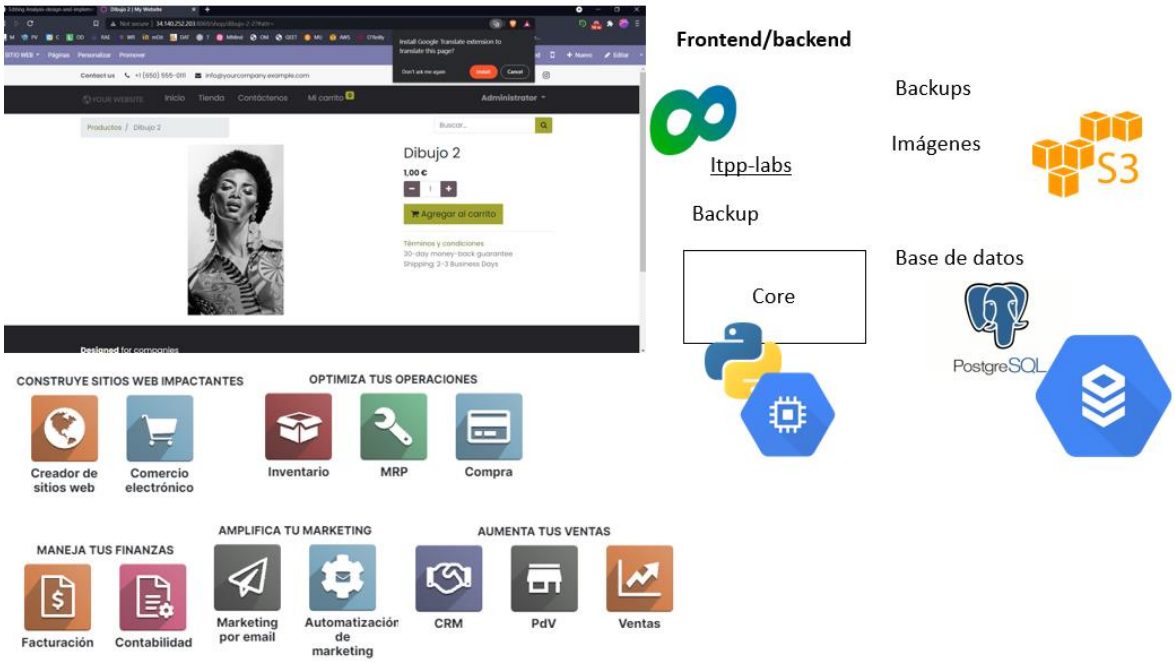


Ilustración 24. Arquitectura del artefacto 4 con compute engine

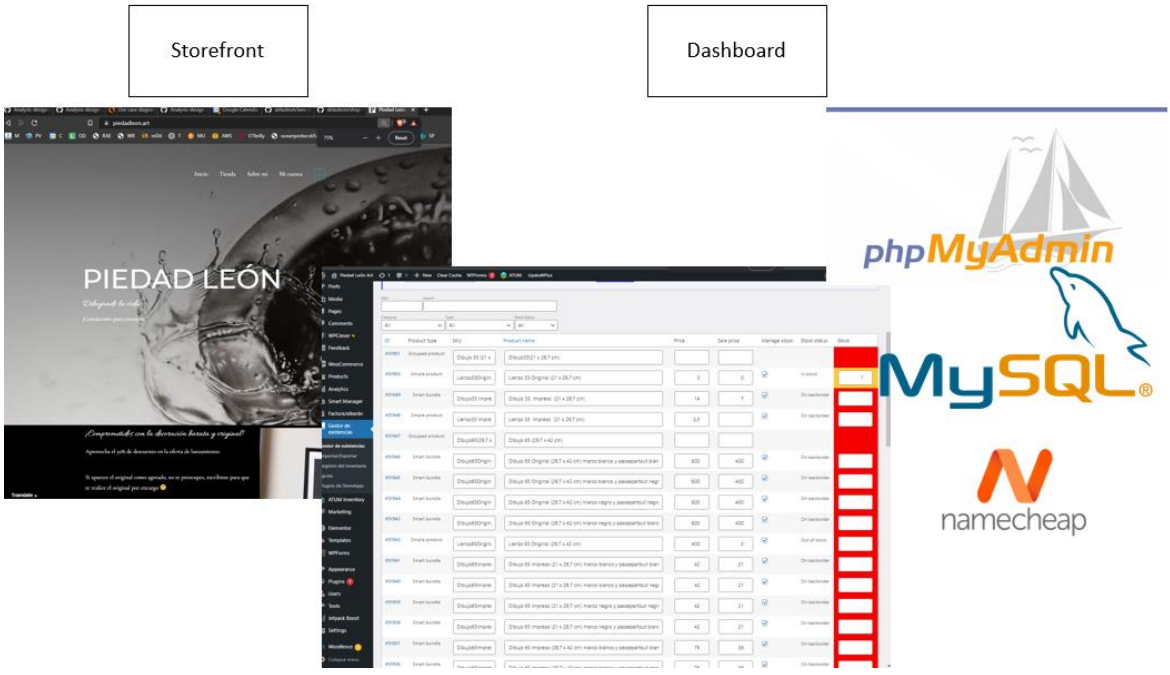


Ilustración 25. Arquitectura del artefacto 5.

5 IMPLEMENTACIÓN

En este proyecto de 850 horas se han desarrollado 6 artefactos. Como eje temporal:

En el verano del año pasado realicé la fase 0, y la fase 1

La fase 2en otoño del año pasado

La fase 3 desde invierno hasta junio

La fase 4 también y la fase 5 en agosto y en septiembre de este año

A continuación, describo cada uno de los artefactos.

- **Fase 0: comparativa despliegues: jenkinsX(eks, gke), cloudformation**
- **Fase 1: FaaS shop**
- **Fase 2: saleor + gcp (cloud run + cloud sql)**
- **Fase 3: odoo ce + aws (fargate + auroraserverless)**
- **Fase 4: odoo ce + gcp (cloud run + cloud sql) y vm**
- **Fase 5: (wp + wc + plugins) + namecheap**

En cada fase se realiza un artefacto, y en cada sección se discuten de él: primero una sección de análisis en la que se relacionan los requisitos capturados con las razones a nivel personal de por qué se toma la decisión de diseñar ese artefacto del conjunto de tecnologías presentadas. A continuación, en una fase de diseño se describen las partes de las que consta el artefacto. Después, en una sección de guía de instalación se describen qué pasos se han seguidos para elaborar ese artefacto. En una fase final de evaluación se evalúa en el merco comparativo propuesto de qué manera es ese artefacto útil para el cliente.

5.1 FASE 0: COMPARATIVA DESPLIEGUES: JENKINSX(EKS, GKE), CLOUDFORMATION

5.1.1 Análisis

El objetivo de esta fase inicial no es implementar una solución entera, sino comparar en cuanto a la parte de infraestructura las dos soluciones más extendidas IaaS, sobre todo para comprar precio: kubernetes y arquitecturas de cloud formation, resultando ser kubernetes la solución a elegir, si bien ambas son válidas y plantean un coste de 60€/mes, la gran ventaja de kubernetes frente a cloud formation es que no es una solución propietaria.

5.1.2 Diseño

En la fase 0 investigo sobre formas de desplegar en la nube un entorno de alta disponibilidad. Por un lado despliego un clúster en kubernetes al que aplico la herramienta Jenkins X para automatizar los pipelines desde git.

Ilustración 26. Partes del buildpack de jenkins X [24]

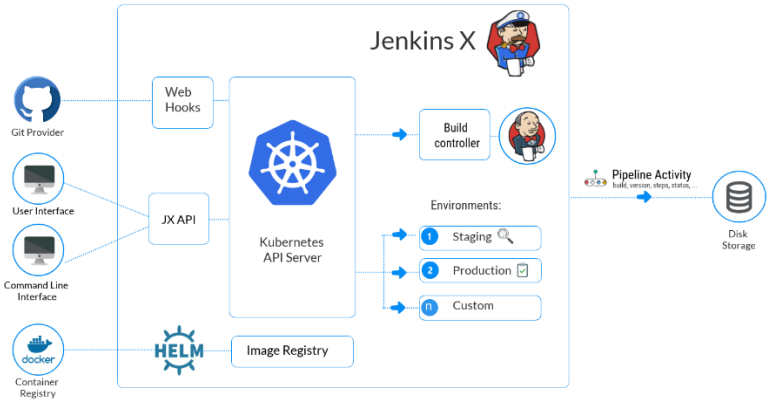
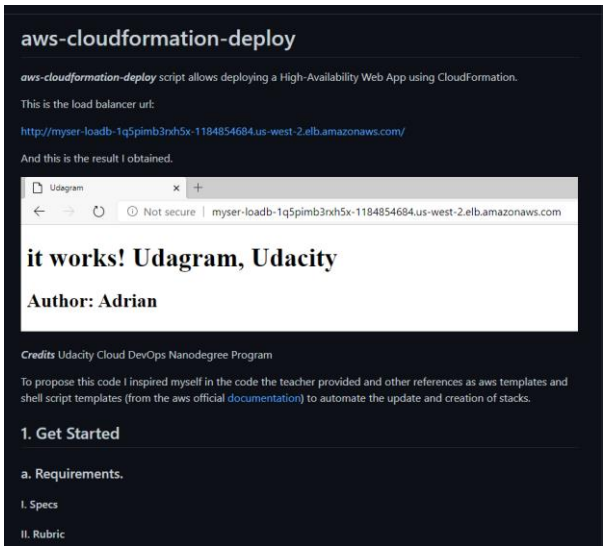


Ilustración 27. Arquitectura de Jenkins X

Y por otro lado, aprendo cloud formation para desplegar un entorno en AWS. Para ello implemento en script bash y valores yml de distintos ficheros como server.yml



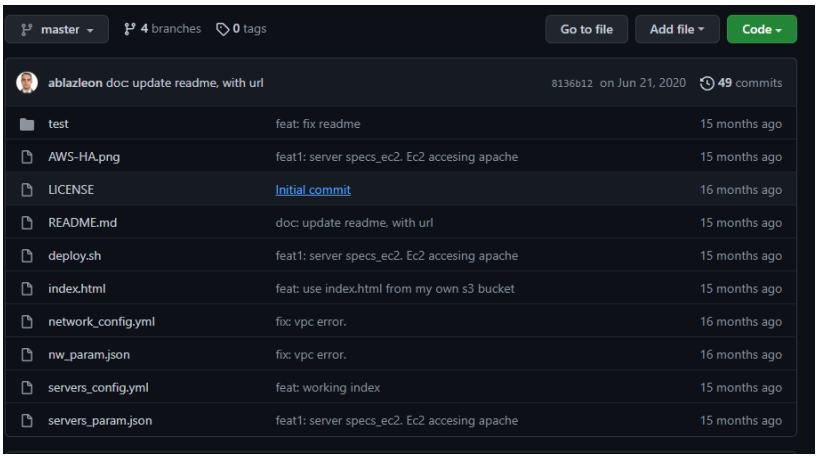


Ilustración 28. Captura de inicio del servicio levantado en cloud formation y estructura del proyecto [25]

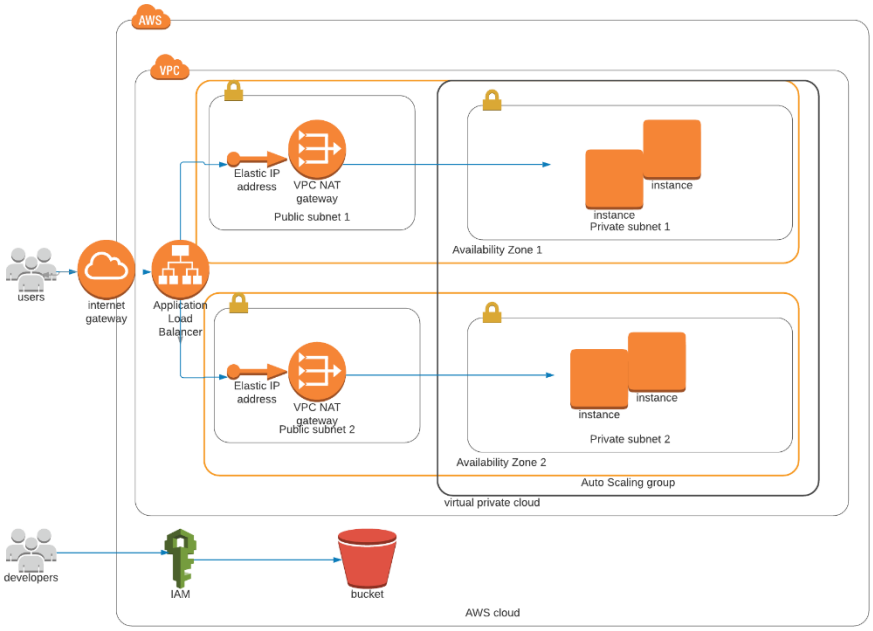


Ilustración 29. Arquitectura del despliegue en cloudformation, según las instrucciones del curso de Udacity

5.1.3 Guía de instalación

Con respecto al despliegue usando Jenkins X:

Paso 1: Proponer y definir el alcance del proyecto: planifique cómo se será el pipeline.

Install -> lint -> test -> Rolling deployments (on certain environments)

Decide qué opciones incluirás en tu fase de Integración Continua.Utilice Jenkins.

```
buildPack: python
pipelineConfig:
 pipelines:
  release:
   build:
    steps:
    - sh: pip install --upgrade pip
      name: upgrade
    - sh: pip install -r requirements.txt
      name: install
    - sh: pylint --disable=R,C,W1203 app.py
      name: lint
```

Elija un tipo de implementación: implementación continua o implementación azul / verde.

Para la aplicación Docker, se puede usar una aplicación propia, o usar una aplicación de código abierto extraída de Internet, o, puede usar un Nginx "Hola mundo, mi nombre es (nombre del estudiante)".

Paso 2: usa Jenkins e implementa la implementación azul / verde o continua.

Cree su master box de Jenkins con Jenkins e instale los complementos que necesitará. Configure su entorno en el que implementará el código.

Se aprovechan las actualizaciones continuas [26] nativas de los deployments de kubernetes, para implementar un deployment progresivo. Se comprueba que las versiones del pod, estén ejecutándose cuando se promueva un cambio. Entonces, primero, se promueve una versión 1, y luego se asegura que es un cambio progresivo entre ellos.

Paso 3: elija AWS Kubernetes como servicio o cree su propio clúster de Kubernetes.

Utilice Ansible o CloudFormation para construir su "infraestructura"; es decir, el clúster de Kubernetes. Primero, se prueba con este clúster [27]. Tenga en cuenta que, para administrar eks form cloud9, se deben seguir estos pasos [28] . Básicamente agregando el rol de acceso de administración en lugar de los relacionados con el ec2

Debe crear las instancias EC2 (si está construyendo las suyas propias), establecer la configuración de red correcta e implementar el software en estas instancias.

Como paso final, será necesario inicializar el clúster de Kubernetes. La inicialización del clúster de Kubernetes se puede realizar a mano o con Ansible / Cloudformation.

Paso 4: Construye tu pipeline

Construya su pipeline en su repositorio de GitHub. Configure todos los pasos que incluirá su canalización. Configure una implementación del pipeline. Incluya su Dockerfile / código fuente en el repositorio de Git. Después de crear la aplicación quickstart se copia y modifica, y luego se aplica:

```
jx import
```

Paso 5: prueba tu pipeline

Realice compilaciones en su pipeline. Verifique que su pipeline funcione como lo diseñó. Luego, usando el “helm chart” se implementa el servicio

En cuanto se refiere al despliegue con cloud formation:

1. Configurar aws cli y usuario
    1. Cree un usuario para probar esta implementación
    2. Instale aws cli y ejecútelo en powershell en esta carpeta "aws --version"
    3. aws configure, con la clave secreta del usuario creado
    4. aws s3 ls, para comprobar la configuración
2. Escriba un .sh que ejecute el yml. Como esta referencia [29]
3. Intente ejecutar un stack de cloudformation sencillo

Solo un ec2 y ejecute el código: verifique que se ejecute correctamente

4. Realice las especificaciones del servidor

Crea una red

Cree un grupo de ajuste de escala automático. Y compruebe la accesibilidad.

Como el grupo de ajuste de escala automático por sí solo no funciona, se busca una instancia de diseñador.

5. Hacer los grupos y roles de seguridad

6. Haz las otras consideraciones

En cuanto al uso, este script en Shell describe la infraestructura descrita:

```
./deploy.sh networkstack network_config.yml nw_param.json
./deploy.sh serversstack servers_config.yml servers_param.json
```

Para depurar:

```
bash -x ./deploy.sh networkstack network_config.yml nw_param.json
bash -x ./deploy.sh serversstack servers_config.yml servers_param.json
```

5.1.4 Evaluación

Finalmente, si se tuviera que describir lo bueno de la solución en función del marco de métricas objetivas en todas obtendría un 0, puesto que como se ha comentado esta solución despliega la arquitectura, no un software que permita ofrecer los casos de uso.

Ilustración 30. Fase 0 en el marco de referencia para comparar

5.2 FASE 1: FAAS SHOP

5.2.1 Análisis

En esta segunda fase, el objetivo del artefacto es poder proveer el servicio de manera lo más barata posible para que pueda seguir en producción, a diferencia de las estructuras levantadas previamente con gke, eks o cloudformation, de mayor precio. Primero se implementa un frontend básico en html y se encuentra en lambda el hosting más barato, pues el primer millón de peticiones se ofrece gratis [30] . Luego, se plantea que para que el gestor de la tienda no sepa programar es necesario implementar más lógica. Entonces para ello se implementa una demo de tienda serverless sobre lambda usando el stack serverless [**Error! Reference source not found.Error! Reference source not found.**]

5.2.2 Diseño

Básicamente con la funcionalidad de un servicio de “todos” un poco ampliado basado en el servicio de todos del curso cloud devops de Udacity. Con una base de un frontend en angular y un backend en typescript también usando dynamo db.

Esta aplicación permitirá crear / eliminar / actualizar / recuperar elementos. Cada elemento puede tener opcionalmente una imagen adjunta. Cada usuario solo tiene acceso a los elementos que ha creado.

 # Artículos TODO

La aplicación debe almacenar elementos y cada elemento contiene los siguientes campos:

* `itemId` (cadena) - una identificación única para un artículo

* `createdAt` (cadena): fecha y hora en que se creó un elemento

* `name` (cadena): nombre de un artículo (por ejemplo," Zanahorias 100g ")

* `price` (cadena) - precio del artículo

* `attachUrl` (cadena) (opcional): una URL que apunta a una imagen adjunta a un elemento

También puede almacenar una identificación de un usuario que creó un artículo.

 # Funciones a implementar

Para implementar este proyecto, necesita implementar las siguientes funciones y configurarlas en el archivo `serverless.yml`:

* `Auth`: esta función debe implementar un autorizador personalizado para API Gateway que debe agregarse a todas las demás funciones.

* `GetItems`: debe devolver todos los elementos de un usuario actual. Se puede extraer una identificación de usuario de un token JWT enviado por la interfaz

Debería devolver datos que se vean así:

json

```
{
  "elementos": [
    {
      "itemId": "123",
      "createdAt": "2019-07-27T20: 01: 45.424Z",
      "name": "Zanahorias 100g",
```

```
      "precio": "0.5",
      "attachUrl": "http://example.com/image.png"
    },
    {
      "todoId": "456",
      "createdAt": "2019-07-27T20: 01: 45.424Z",
      "name": "Filetes de pollo 200g",
      "precio": "2",
      "attachUrl": "http://example.com/image.png"
    }
  ]
}
''
```

* `CreateItem` - debe crear un nuevo elemento para un usuario actual. Una forma de datos enviados por una aplicación cliente a esta función se puede encontrar en el archivo `CreateItemRequest.ts`

Recibe un nuevo elemento para ser creado en formato JSON que se ve así:

json

```
{
  "createdAt": "2019-07-27T20: 01: 45.424Z",
  "name": "Compra leche",
  "dueDate": "2019-07-29T20: 01: 45.424Z",
  "hecho": falso,
  "attachUrl": "http://example.com/image.png"
}
''
```

Debería devolver un nuevo elemento TODO que se ve así:

json

```
{
  "todoId": "456",
  "createdAt": "2019-07-27T20: 01: 45.424Z",
  "name": "Filetes de pollo 200g",
  "precio": "2",
  "attachUrl": "http://example.com/image.png"
}
```

* `UpdateItem`: debe actualizar un elemento creado por un usuario actual. Una forma de datos enviados por una aplicación cliente a esta función se puede encontrar en el archivo `UpdateItemRequest.ts`

Recibe un objeto que contiene tres campos que se pueden actualizar en un elemento TODO:

json

```
{
  "nombre": "Pan",
  "dueDate": "2019-07-29T20: 01: 45.424Z",
  "precio": "1"
}
''
```

La identificación de un elemento que debe actualizarse se pasa como un parámetro de URL.

Debería devolver un cuerpo vacío.

* `DeleteItem`: debe eliminar un elemento creado por un usuario actual. Espera una identificación de un artículo para eliminar.

Debería devolver un cuerpo vacío.

* `GenerateUploadUrl`: devuelve una URL firmada previamente que se puede usar para cargar un archivo adjunto para un elemento.

Debería devolver un objeto JSON que se parece a esto:

json

```
{
  "uploadUrl": "https://s3-bucket-name.s3.eu-west-2.amazonaws.com/image.png"
}
"
```

Todas las funciones ya están conectadas a los eventos apropiados de API Gateway.La identificación de un usuario se puede extraer de un token JWT pasado por un cliente.También debe agregar los recursos necesarios a la sección `resources` del archivo` serverless.yml`, como la tabla DynamoDB y el depósito S3.

 # Interfaz

La carpeta `client` contiene una aplicación web que puede usar la API que se debe desarrollar en el proyecto.

Esta interfaz debería funcionar con su aplicación sin servidor una vez desarrollada, no necesita realizar ningún cambio en el código. El único archivo que necesita editar es el archivo `config.ts` en la carpeta` client`. Este archivo configura su aplicación cliente y contiene un punto final de API y una configuración de Auth0:

ts

```
const apiId = '...' ID de puerta de enlace API

export const apiEndpoint = `https: // $ {apiId} .execute-api.us-east-1.amazonaws.com / dev`


export const authConfig = {
  dominio: '...', // Dominio de Auth0
  clientId: '...', // ID de cliente de una aplicación Auth0
  callbackUrl: 'http: // localhost: 3000 / callback'
}
''
```

Autenticación

Para implementar la autenticación en su aplicación, tendría que crear una aplicación Auth0 y copiar "dominio" y "ID de cliente" al archivo `config.ts` en la carpeta` cliente`. Recomendamos el uso de tokens JWT cifrados asimétricamente.

Inicio sesión

El código de inicio viene con un registrador Winston [31] configurado que crea JSON formateado [32] declaraciones de registro. Puede usarlo para escribir mensajes de registro como este:

ts

```
importar {createLogger} desde '../../utils/logger'

const logger = createLogger ('auth')
```

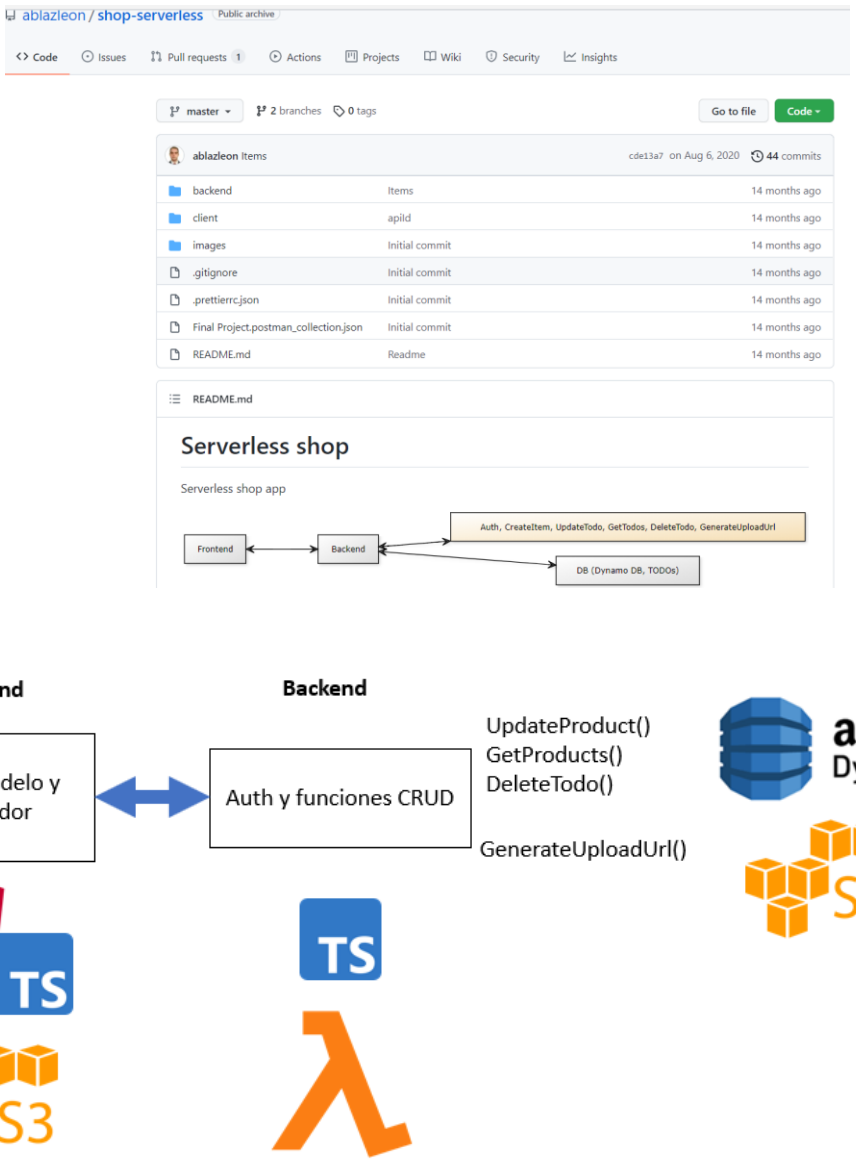
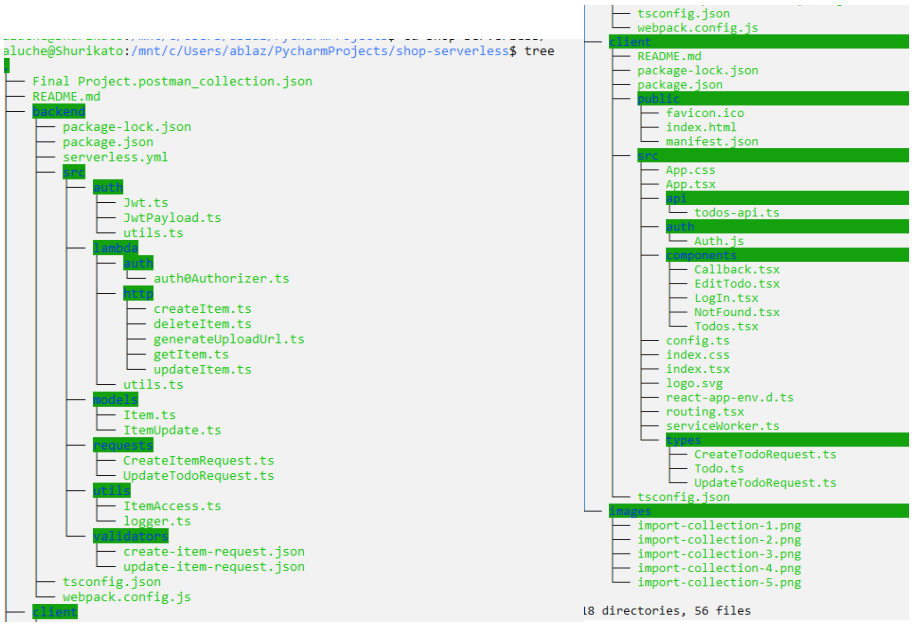


Ilustración 31. Arquitectura del shop serverless. Desloges del backend y del frontend



5.2.3 Guía de instalación

Backend

Para implementar una aplicación, ejecute los siguientes comandos:

```
backend de cd

npm install

sls deploy -v
```

Interfaz

Para ejecutar una aplicación cliente, primero edite el archivo `client / src / config.ts` para establecer los parámetros correctos. Y luego ejecute los siguientes comandos:

```
cliente cd

npm install

npm run start
```

Esto debería iniciar un servidor de desarrollo con la aplicación React que interactuará con la aplicación TODO sin servidor.

Después se ha testeado esto se procede a:

1. Ejecutar frontend: está marcado que es necesario para implementar el inicio de sesión

2. Configurar el marco sin servidor

A. npm install -g sin servidor

B. Configure un nuevo usuario en IAM llamado "sin servidor" y guarde la clave de acceso y la clave secreta

C. sls config credentials --provider aws --key YOUR_ACCESS_KEY --secret YOUR_SECRET_KEY --profile serverless

3. Configure un autorizador personalizado

A. Cree una aplicación en autho 0.

B. Coloque el certificado en el jwturl del controlador implementado

4. Configure los recursos serverless.yml

5. Cree las funciones CRUD

5.2.4 Evaluación

Finalmente, en cuanto al marco comparativo de referencia, quedaría como en la ilustración siguiente. Este artefacto permite visualizar el catálogo, pero no permite ni seguir el pedido ni el constructor web. Ni el caso de uso de gestor de operaciones. En cuanto a usabilidad, no se aplicó lighthouse porque no se entró en fase beta, pero se plantea un score razonable de 2. En cuanto a seguridad se plantea backup pues aws s3 y dynamo db lo permiten, pero no se plantea ni creación de roles ni alerta ni 2FA porque no proporciona nativo el servicio. Por último, en cuanto al precio, se plantea para un número de visitas de una pyme razonable, de menos de 100k al mes, un gasto de menos de 10€/mes.

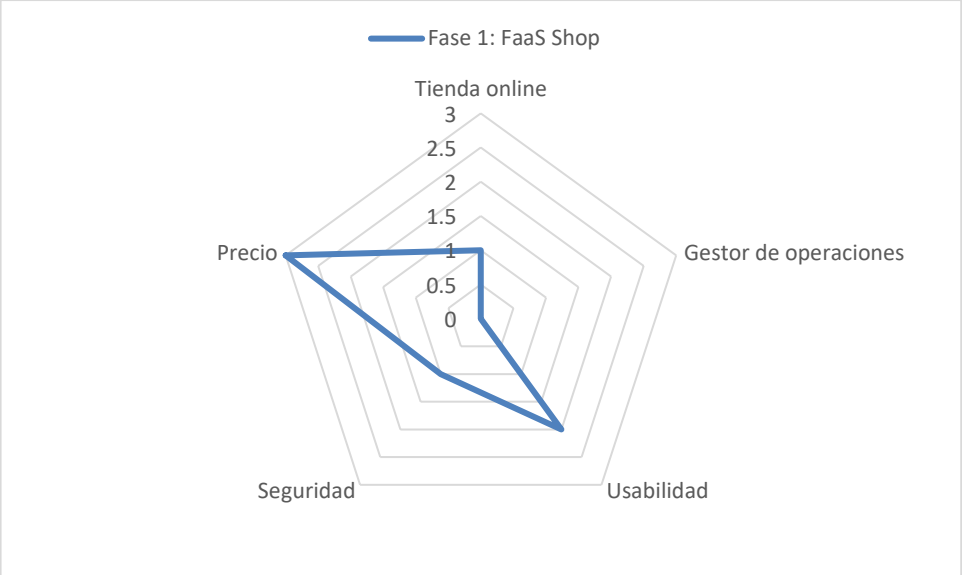


Ilustración 32. Fase 1 bajo el marco de referencia

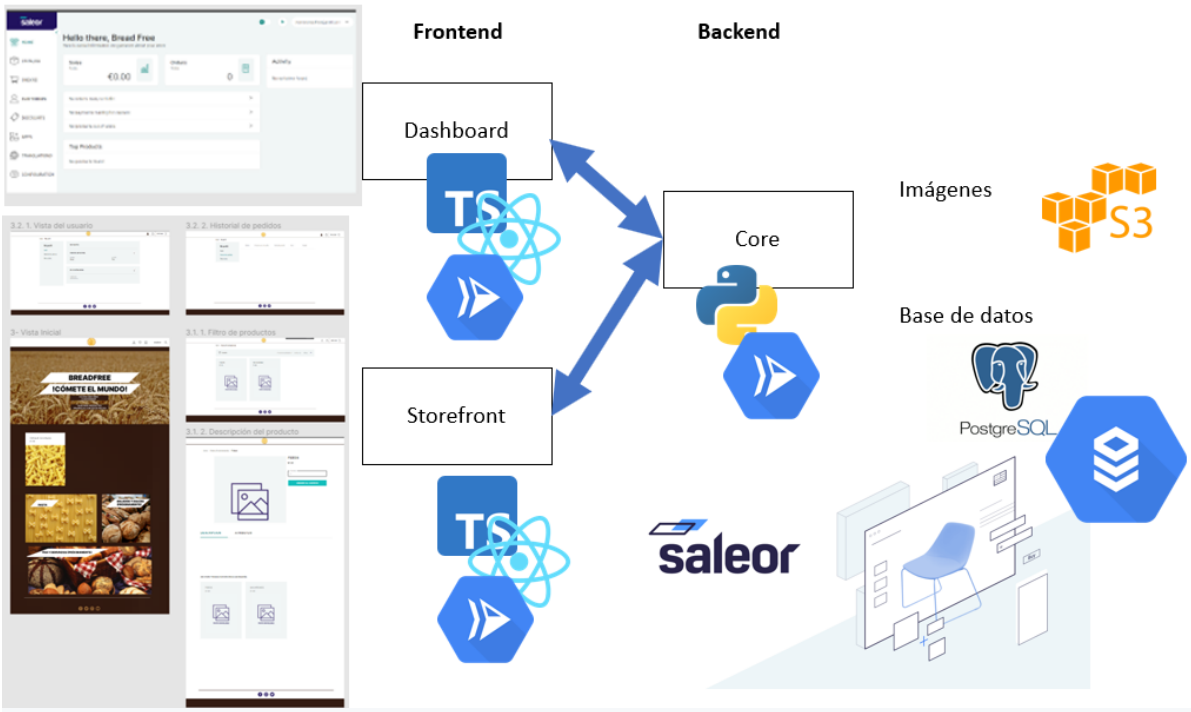
5.3 FASE 2: SALEOR + GCP (CLOUD RUN + CLOUD SQL)

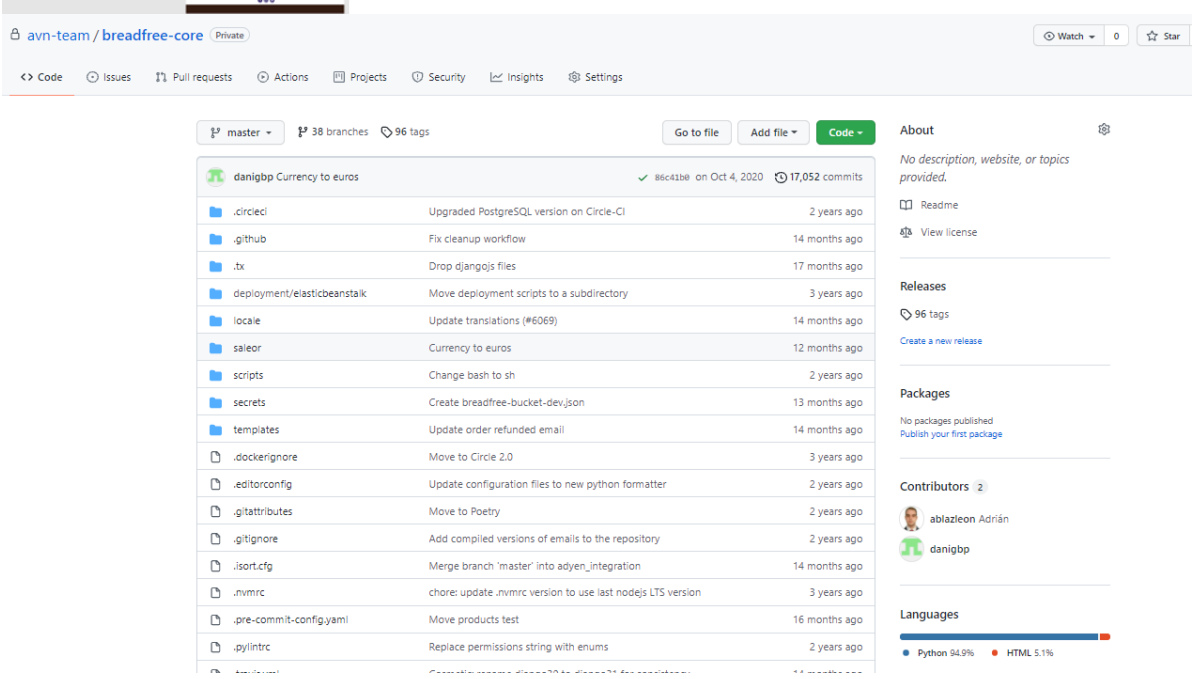
5.3.1 Análisis

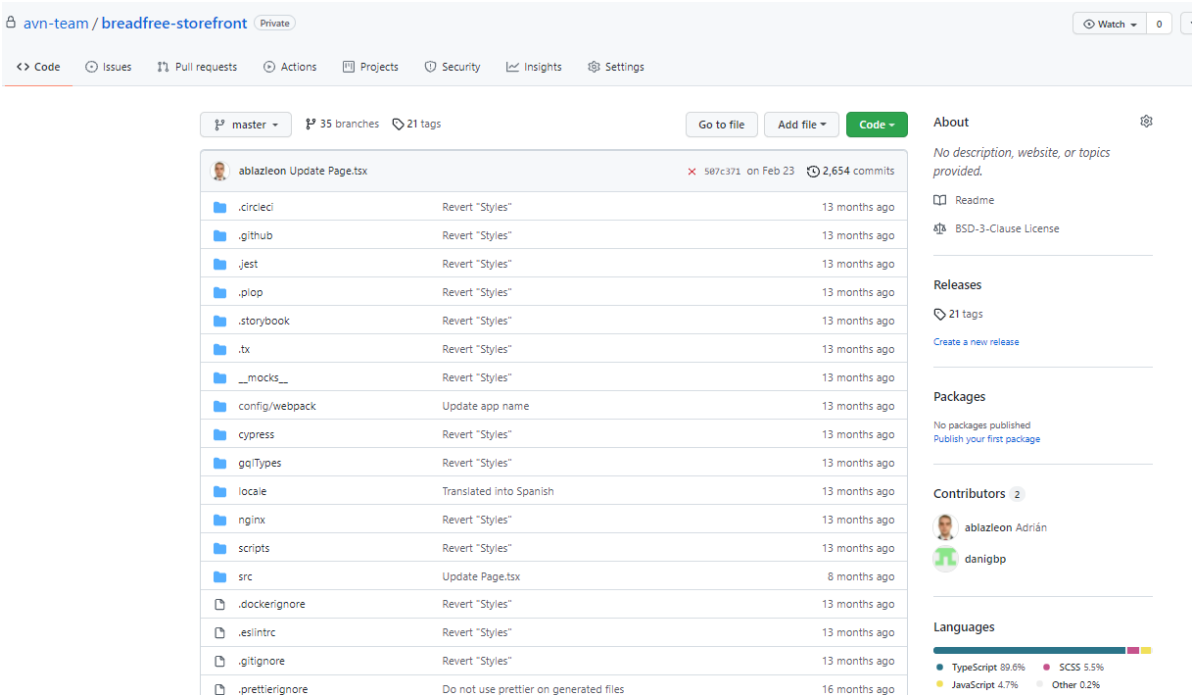
Se plantea desplegar Saleor [**Error! Reference source not found.**], un servicio que cumple el caso de uso 1 en Google cloud run, un servicio de contenedores as a servicio cuyo primer millón de peticiones es gratis. De forma que lo único que se pagaría sería el postgres db en la nube. Se encuentra saleor un proyecto de código libre de tienda headless con frontend con graphql para acceder al modelo y django como backend. Se encuentra que saleor, es muy personalizable en cuanto al frontend. Se puede implementar código para que se ajuste al diseño del cliente.

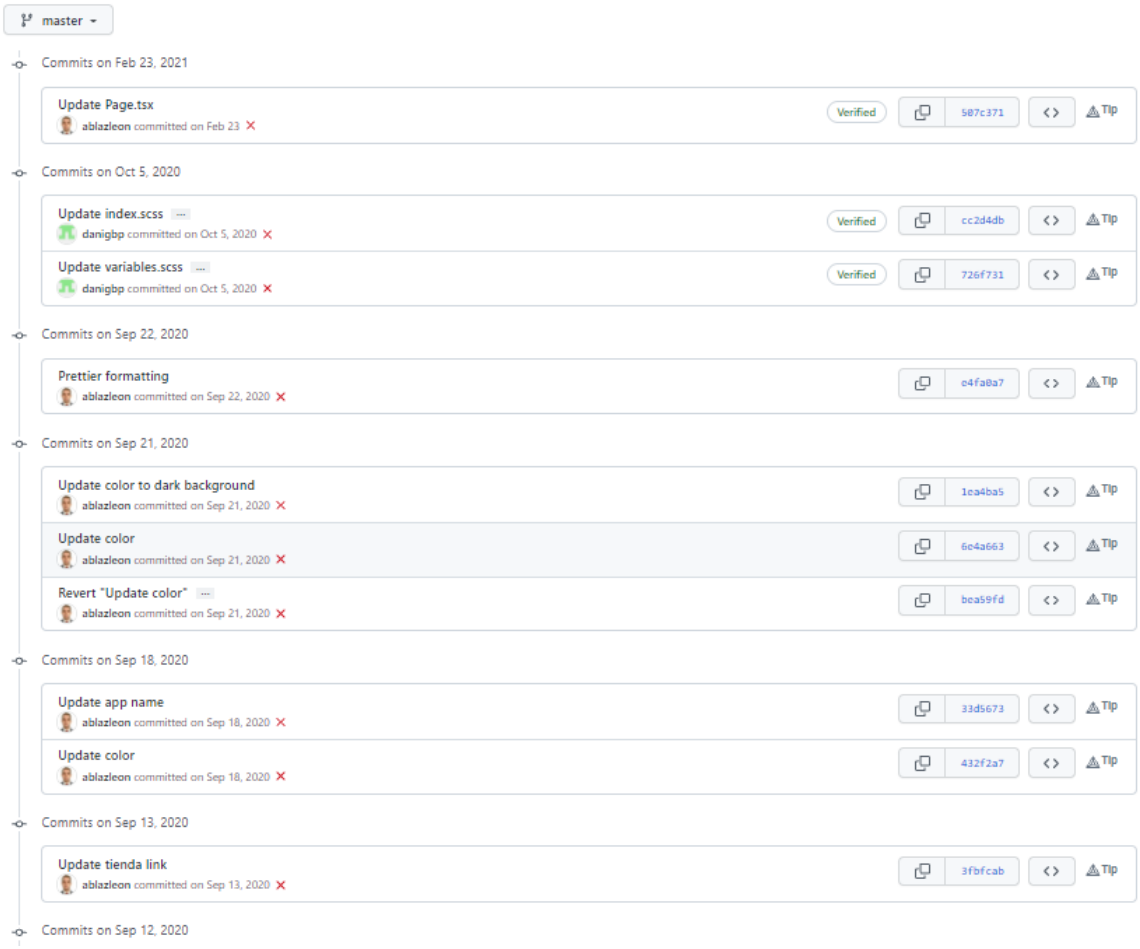
5.3.2 Diseño

A continuación, muestro captura del código de este tercer producto. Lo componen 3 microservicios saleor core, saleor dashboard y saleor storefront. Estos dos últimos en typescript. Se resaltan también ciertos cambios que se implementaron en el servicio de stroefront, primero el entender el proyecto y cambiar sobre cada repo los link de los servicios para ligar los microservicios. Otro cambio importante fue configurar cloudbuild.yaml para que se creara el contenedor. Y otro fue añadir un botón “pincha para conocernos” que linkaba a un host donde estaba la página de descripción del equipo.


Frontend
Backend
Dashboard
Storefront
Core
Imágenes
S3
Base de datos
PostgreSQL
saleor


avn-team / breadfree-core
Private
Watch
Star
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
master
38 branches
96 tags
Go to file
Add file
Code
danigbp Currency to euros
86c41b0 on Oct 4, 2020
17,052 commits
.circleci Upgraded PostgreSQL version on Circle-CI 2 years ago
.github Fix cleanup workflow 14 months ago
.tx Drop djangojs files 17 months ago
deployment/elasticbeanstalk Move deployment scripts to a subdirectory 3 years ago
locale Update translations (#6069) 14 months ago
saleor Currency to euros 12 months ago
scripts Change bash to sh 2 years ago
secrets Create breadfree-bucket-dev.json 13 months ago
templates Update order refunded email 14 months ago
.dockerignore Move to Circle 2.0 3 years ago
.editorconfig Update configuration files to new python formatter 2 years ago
.gitattributes Move to Poetry 2 years ago
.gitignore Add compiled versions of emails to the repository 2 years ago
.isort.cfg Merge branch 'master' into adyen_integration 14 months ago
.nvmrc chore: update .nvmrc version to use last nodejs LTS version 3 years ago
.pre-commit-config.yaml Move products test 16 months ago
.pylintrc Replace permissions string with enums 2 years ago
About
No description, website, or topics provided.
Readme
View license
Releases
96 tags
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
ablazleon Adrián
danigbp
Languages
Python 94.9%
HTML 5.1%


avn-team / breadfree-storefront
Private
Watch 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
master
35 branches
21 tags
Go to file
Add file
Code
ablazleon Update Page.tsx
507c371 on Feb 23
2,654 commits
.circleci Revert "Styles" 13 months ago
.github Revert "Styles" 13 months ago
.jest Revert "Styles" 13 months ago
.plop Revert "Styles" 13 months ago
.storybook Revert "Styles" 13 months ago
.tx Revert "Styles" 13 months ago
__mocks__ Revert "Styles" 13 months ago
config/webpack Update app name 13 months ago
cypress Revert "Styles" 13 months ago
gqlTypes Revert "Styles" 13 months ago
locale Translated into Spanish 13 months ago
nginx Revert "Styles" 13 months ago
scripts Revert "Styles" 13 months ago
src Update Page.tsx 8 months ago
.dockerignore Revert "Styles" 13 months ago
.eslintrc Revert "Styles" 13 months ago
.gitignore Revert "Styles" 13 months ago
.prettierignore Do not use prettier on generated files 16 months ago
About
No description, website, or topics provided.
Readme
BSD-3-Clause License
Releases
21 tags
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
ablazleon Adrián
danigbp
Languages
TypeScript 89.6%
SCSS 5.5%
JavaScript 4.7%
Other 0.2%


master
Commits on Feb 23, 2021
Update Page.tsx
ablazleon committed on Feb 23
Verified
507c371
Commits on Oct 5, 2020
Update index.scss
danigbp committed on Oct 5, 2020
Verified
cc2d4db
Update variables.scss
danigbp committed on Oct 5, 2020
Verified
726f731
Commits on Sep 22, 2020
Prettier formatting
ablazleon committed on Sep 22, 2020
e4fa0a7
Commits on Sep 21, 2020
Update color to dark background
ablazleon committed on Sep 21, 2020
1ea4ba5
Update color
ablazleon committed on Sep 21, 2020
6e4a663
Revert "Update color"
ablazleon committed on Sep 21, 2020
bea59fd
Commits on Sep 18, 2020
Update app name
ablazleon committed on Sep 18, 2020
33d5673
Update color
ablazleon committed on Sep 18, 2020
432f2a7
Commits on Sep 13, 2020
Update tienda link
ablazleon committed on Sep 13, 2020
3fbfcab
Commits on Sep 12, 2020

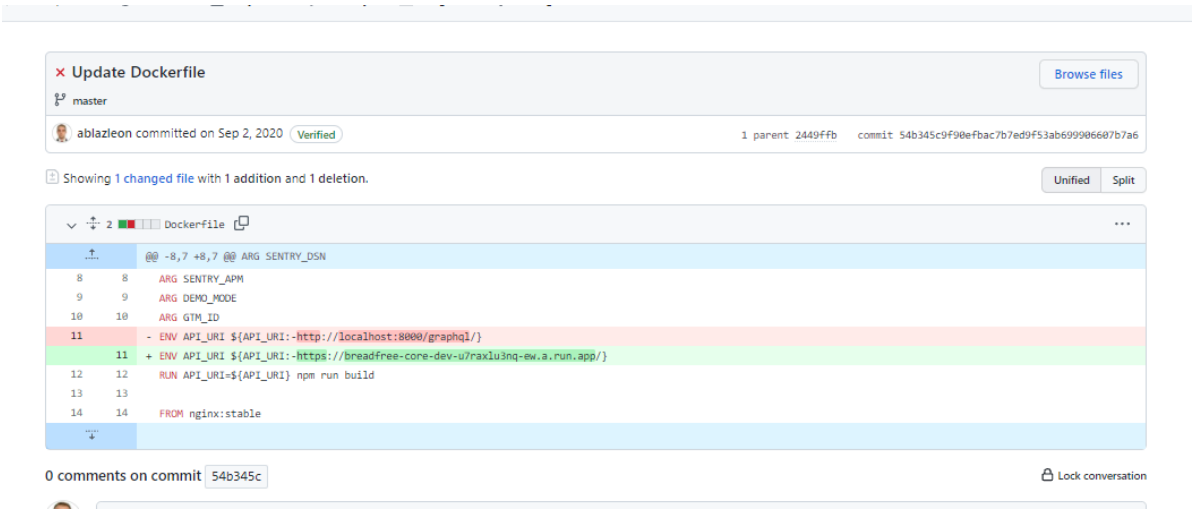


Ilustración 33. Vistas del código de los proyectos de saleor, sólo dashboard y core, faltaría mostrar storefront. Y cambios sobre breaafree sotrefront (https://github.com/avn-team/breadfree-storefront/commits/master)

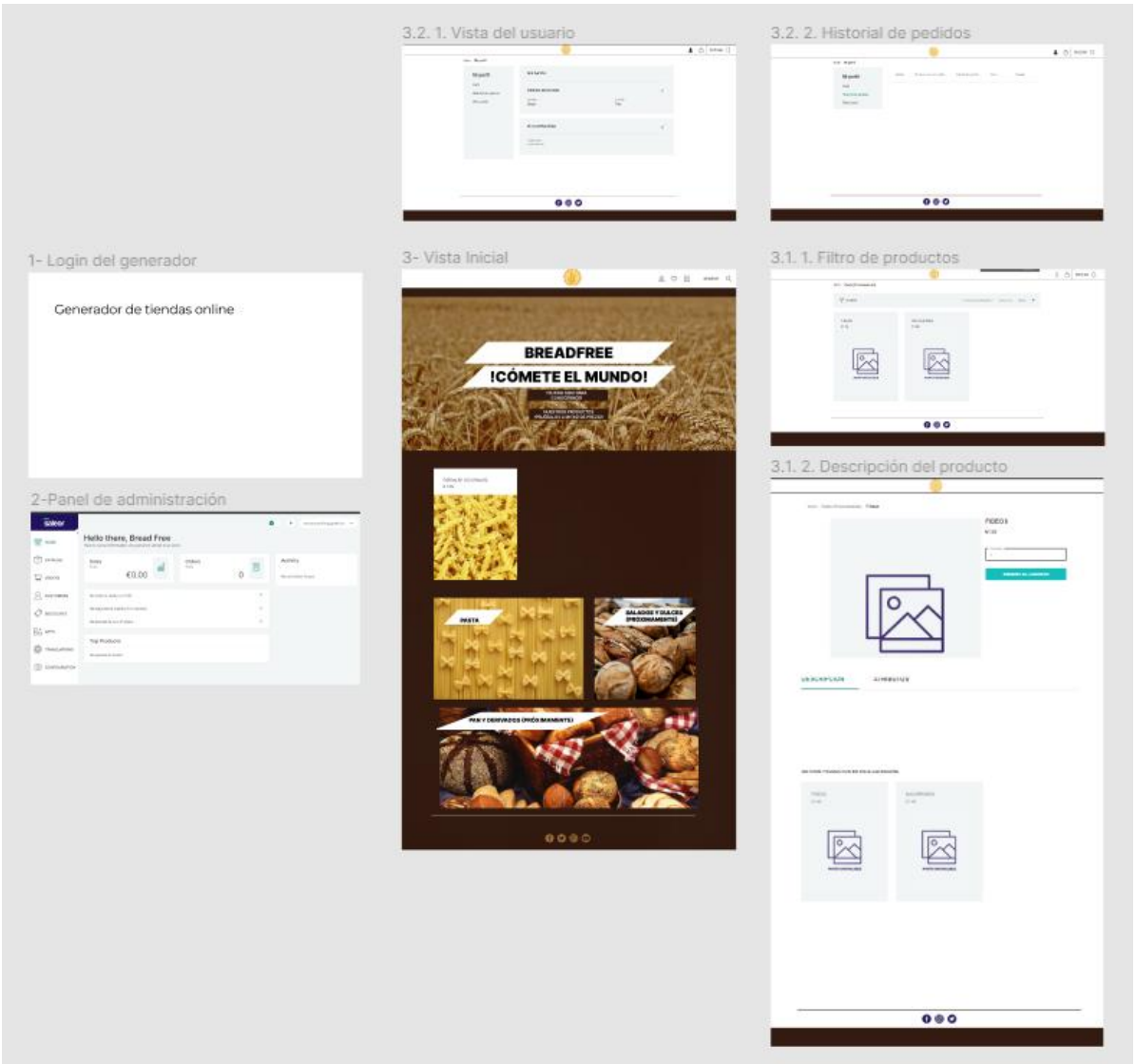


Ilustración 34. Vistas configuradas del storefront de saleor en figma

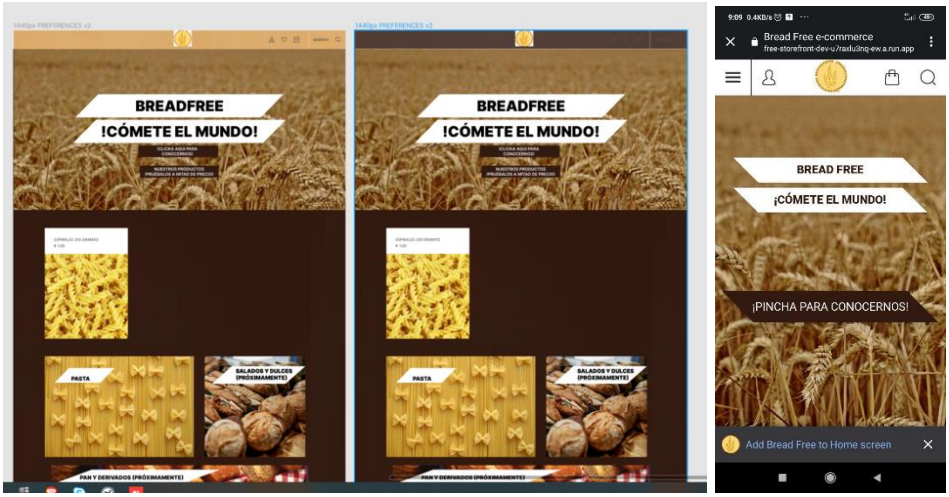


Ilustración 35. Propuesta de preferencias de color a cliente y Webapp de saleor levantada

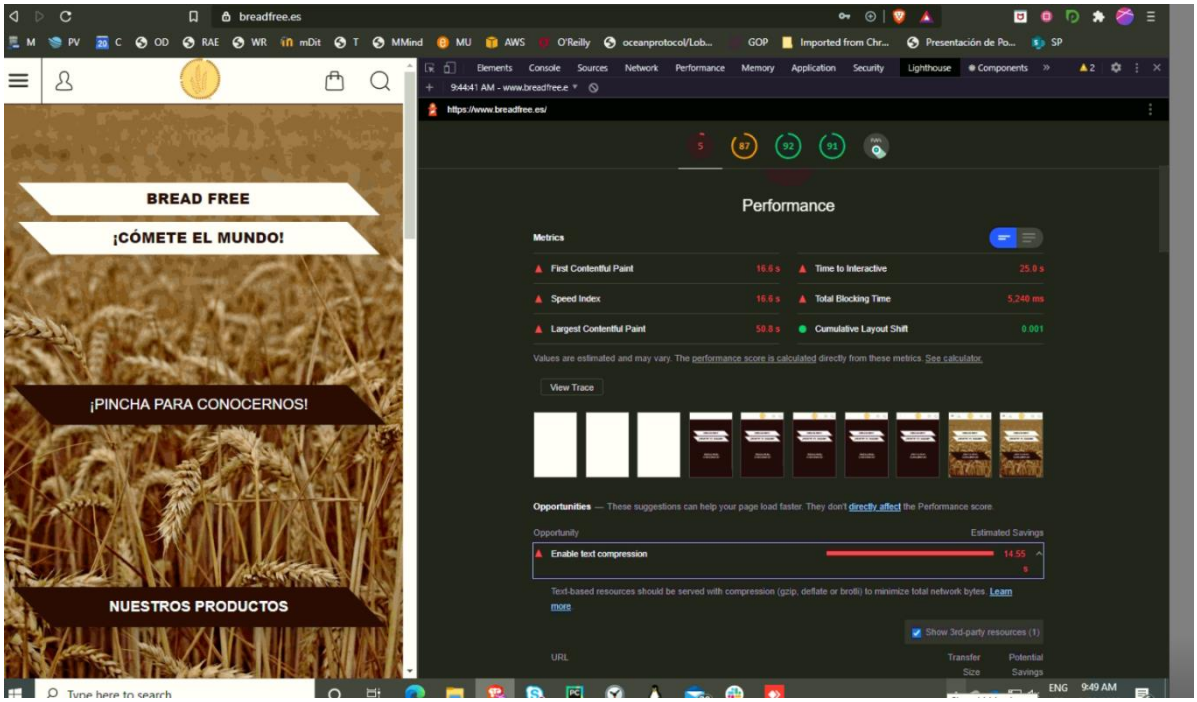


Ilustración 36.Test de ligthhouse sobre saleor en web. Se observa el bajo rendimiento

5.3.3 Guía de instalación

En cuanto a la instalación, este artefacto se ha realizado mediante la UI de GCP, creando los servicios.

Si se desea cambiar la interfaz, se propone en figma y según las preferencias del cliente, se cambia en el propio código de react, los colores de estilo, los mensajes, o el nombre de la webapp. Como se ha propuesto para breadfree. Además, se integra con circle ci y cypress de forma que se pueden implementar tests de integración para que en cada cambio que se ejecute, circle ci levante un

contenedor y así se sabe que en cada cambio que se realiza hace que el sistema siga funcionando. En este apartado se ha usado bastante el chat de saleor porque muchas configuraciones no están en los tutoriales.

Se adjuntan dos preguntas realizadas en el chat de spectrum para indicar al lector cómo cambiar aspectos de saleor:

Q: ¿Cómo configurar una imagen de fondo, un color bg y el nombre de la aplicación web en la pantalla de inicio?

¡Hola! Estoy siguiendo una de las publicaciones para personalizar el storefront (enlace al final de la publicación), es muy útil pero tenemos tres pequeñas dudas:

La primera es, ¿Saleor admite actualmente una forma de agregar una imagen de fondo personalizada? Encontré en src / views / Home / Page.tsx que parece ser una entrada en la api de graphql para agregar una imagen de fondo (ProductsList_shop_homepageCollection_backgroundImage). Pero, ¿cómo puedo? ¿Usando cartero? Puedo agregar manualmente una pestaña img, luego arreglarlo con css al tamaño, pero prefiero pedir tu opinión primero, ya que realmente no sé cómo debo administrar estos css allí.

```
import "./scss/index.scss";

import classNames from "classnames"; import * as React from "react"; import { FormattedMessage, useIntl } from "react-intl"; import { Link } from "react-router-dom";

import { Button, Loader, ProductsFeatured } from "../../components"; import { generateCategoryUrl } from "../../core/utils";

import { ProductsList_categories, ProductsList_shop, ProductsList_shop_homepageCollection_backgroundImage, } from "./gqlTypes/ProductsList";

import { structuredData } from "../../core/SEO/Homepage/structuredData";
```

Parte del color de fondo y del color de la pestaña es blanco. ¿Cómo cambiarlo a un color personalizado? Cambio todos los colores en src / globalStyles / scss / variables.scss pero ninguno lo cambia. . .

Finalmente, cuando agrego la aplicación web a mi pantalla de inicio, dice Saleor. Cambié el título index.html a mi personalizado y funciona en la pestaña de título pero no en esta "aplicación" en el nombre de la pantalla de inicio. ¿Puede darme alguna pista sobre cómo cambiar el nombre de Saleor?

¡Muchas gracias de antemano!

A: Hola Adrian. No hay un solo lugar donde pueda cambiar el fondo globalmente. Deberá establecer las variables en los estilos globales y hacer referencia a ellas en los componentes de las páginas: vistas / inicio /.../*. Scss

¿Le gustaría cambiar la imagen de la colección que aparece en la página principal? Si es así, puede hacerlo en el panel de control.

E intente cambiar el título aquí: config / webpack / config.base.js

Encontrado en [33]

Q: ¿Alguien ha logrado mostrar el storefront en otro idioma?

¡Hola! En primer lugar, ¡gracias por este increíble producto! Estoy configurando el escaparate para un amigo y solo entienden español. Como leí que la función de selector de idioma no estará disponible pronto, mientras tanto, esta función se publica, me preguntaba: ¿alguien ha podido cambiar el idioma del escaparate al menos manualmente? (Me refiero a las strings "iniciar sesión", "registrarse"...)Como vi eso en ambas carpetas @next/pages y vistas de las que se extrae el texto@temp/ intl. Pensé que estas cadenas provenían de la carpeta de configuración regional, así que copié el contenido de es.json en defaultMessage.json. Pero las cuerdas no cambiaron. Después, como configuré un depósito GS como un contenedor de archivos estáticos, busqué este contenido allí, pero no hay archivos estáticos. Por cierto, ¿cómo es posible que no haya archivos estáticos en el depósito? Necesitaba configurar 1 GB de memoria en la nube para el saleor-core porque de lo contrario no funciona. Tal vez tuve que hacerlo, porque los archivos estáticos no se almacenan en el bucket.

A: sí, puede, pero esta solución no recomendada:en `/src/commponets/Locale/Locale.jsx` cambiar esta línea de códigoconst defaultLocale = Locale.EN; const locale = Locale.EN;for saleor core: en settings.py cambiar `LANGUAGE_CODE` Sugiero ejecutar saleor core en 2GB de ram para comenzar. y la tienda frontal y el tablero en 1 GB de RAM para comenzar. y agrega espacio de swap

Obtenido de [34]

Q: ¿Se puede habilitar la compresión de texto en la venta o en el storefront?

¡Hola! Estoy configurando el escaparate y tengo una duda. El demo.saleor.io tarda 4 segundos según faro y lo que he desplegado en la nube, 20 s en carga (www.breadfree.es). El análisis dice que la compresión de texto debería ayudar a ahorrar 12 segundos. Si acabo de agregar imágenes y elegir el texto en español por defecto, ¿por qué mi implementación tiene más texto? Y luego, ¿Saleor admite

la compresión de texto de forma nativa? Como no veo el plugin de paquete web de compresión, no es así, ¿no es así? Así que creo que debería instalarlo y configurar el nodo para que sirva este gzip como la publicación de stack overflow. ¿Cuál es tu consejo?

A: no hubo ninguna respuesta, pero se sobreentendió que iba a ser muy difícil optimizar saleor para que cargara rápido.

5.3.4 Evaluación

Finalmente, en cuanto a la comparación bajo el marco de referencia. Este artefacto permite visualizar catálogo, pero ni seguimiento de pedido ni constructor web, ni gestión de operaciones. En cuanto a disponibilidad, se obtiene menos de 5 de score para unos valores de coste de la bbdd de 15€/mes, así que una puntuación de 0. De seguridad sí que se plantean backups y creación de roles pero no alertas ni 2fa. En cuanto a precio, se encuentra en el rango 2 de 15€/mes, si bien mejoraría la usabilidad. El inconveniente que resalto de esta solución es que requiere además de expertize para la configuración del frontend, y sobre todo la falta de integraciones nativas, como con un servicio de gestión de inventarios que avise cuando el stock esté bajo o envío mensaje para que el cliente sepa el estado del pedido.

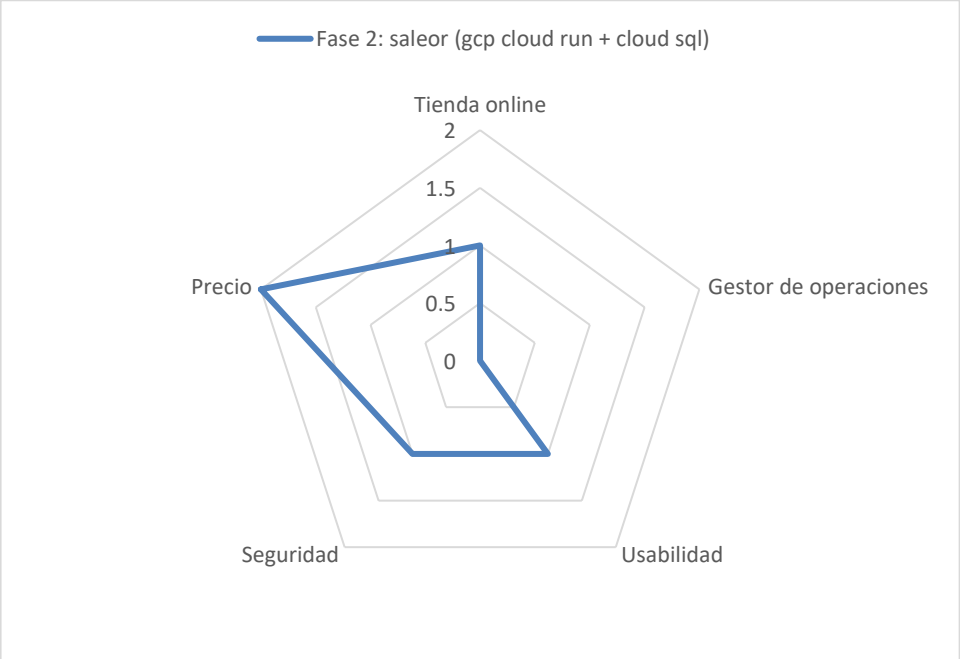


Ilustración 37. Marco de referencia para comprar la solución artefacto 2

5.4 FASE 3: ODOO CE + AWS (FARGATE + AURORASERVERLESS)

5.4.1 Análisis

En esta tercera fase, se plantea una solución que resuelve más casos de uso que saleor, ya que en el mercado, se aprecia que las pequeñas empresas empiezan usando un storefront en algún SaaS, Shopify, wix, o con woocomerce o con saleor, y luego le integran un gestor de inventarios. La apuesta fue desplegar directamente **odoo**, un proyecto de código libre con distintos módulos [**Error! Reference source not found.**]. Uno de ellos que hace las funciones de tienda online y otro de gestor de inventarios.

5.4.2 Diseño

En la tabla de comparación de servicios de Contenedor como Servicio, se identifica que aws posee este servicio fargate [**Error! Reference source not found.**], y además en ese momento se encontró que sacó un base de datos llamada aurora serverless que permite sólo pagar por su uso, de forma que permitiría en vez de gastar 10€/mes como con gcp como se estaba haciendo, como 1€/mes en función del servicio. Sin embargo, tras desplegarlo resulta que, primero, que existe un servicio de aws que se llama copilot que permite automatizar la creación del entorno, crea el escenario con fargate el contenedor as a service, la db pero un balanceador de carga que consume 15€/mes. Y además, resulta que aurora serverless v1 tiene un cold start de 5s, pero se ha sacado un aurora serverless v2 con un tiempo de arranque menor pero que en mayo cuando se prototipó esto no estaba en producción todavía.

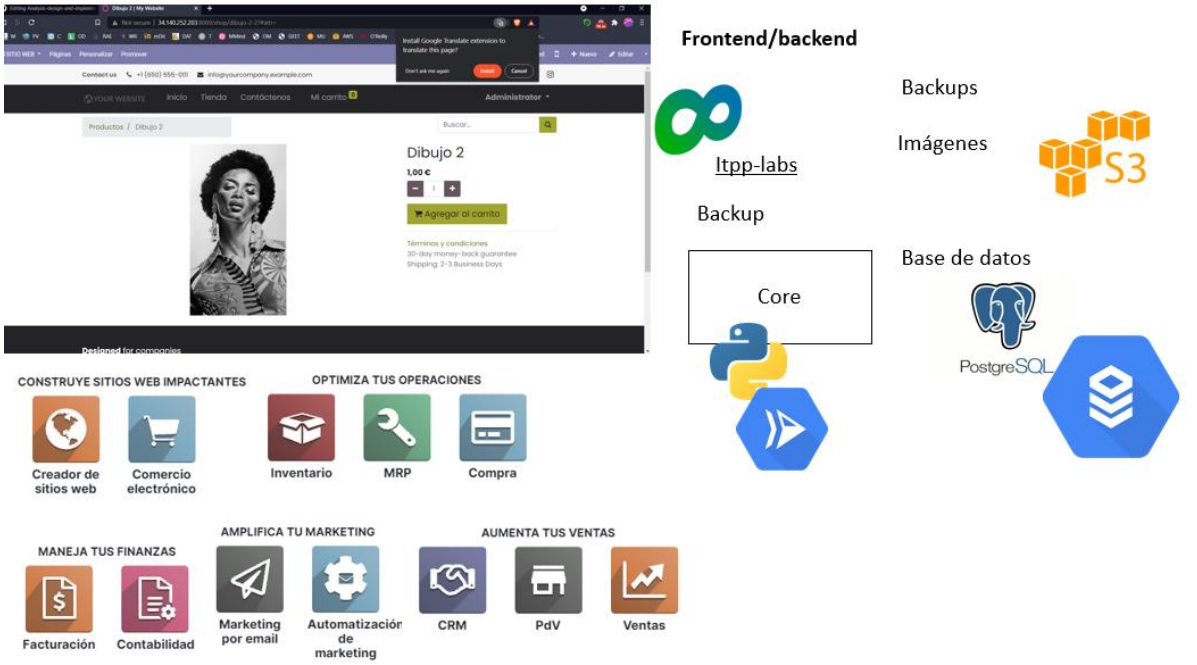
5.5 FASE 4: ODOO CE + GCP (CLOUD RUN + CLOUD SQL) Y VM

5.5.1 Análisis

Los requisitos del apartado anterior plantean desplegar odoo en Google cloud en el contendor de cloud run y la bbdd.

5.5.2 Diseño

Aquí por ejemplo se muestra la arquitectura y una captura pantalla sobre los logs sobre el contendor.



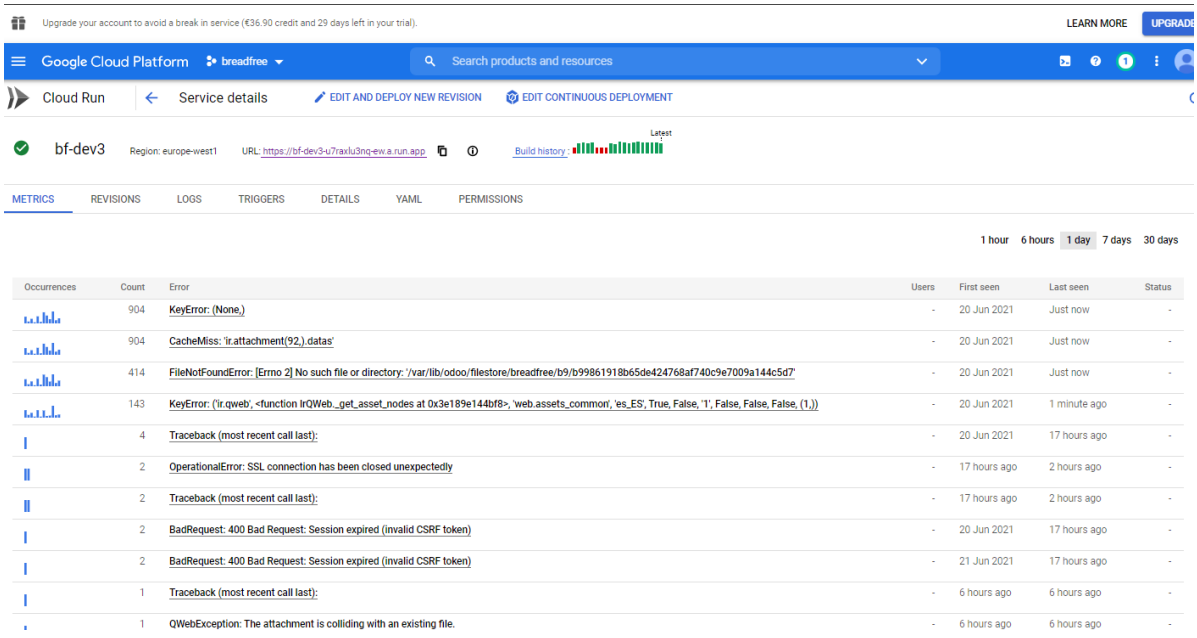


Ilustración 38. Arquitectura de la fase 4 con cloud run y logs del container con gcrun

Se puede ver en esta captura el aspecto de un producto en odoo.

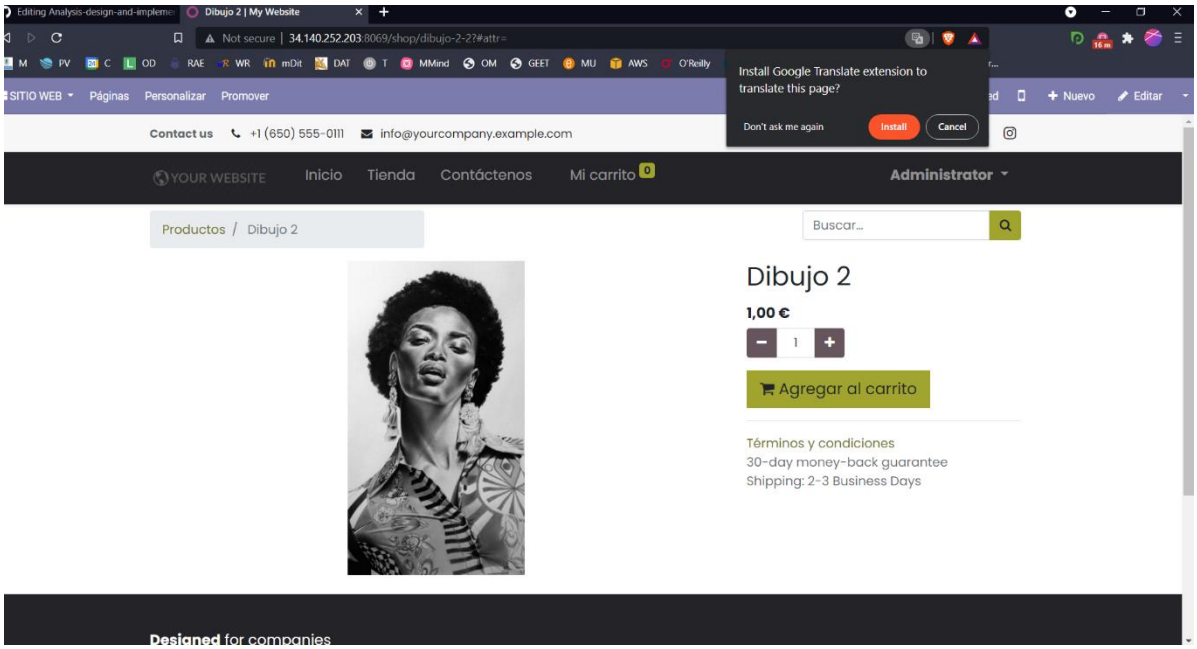
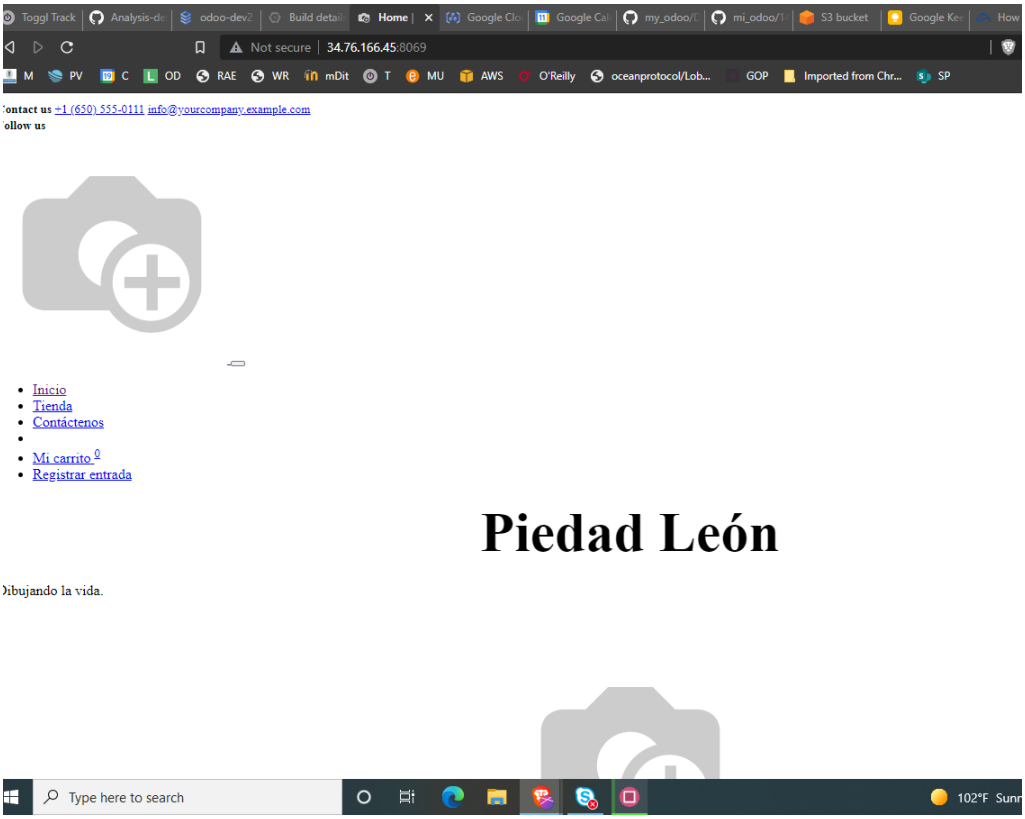


Ilustración 39. Producto en odoo



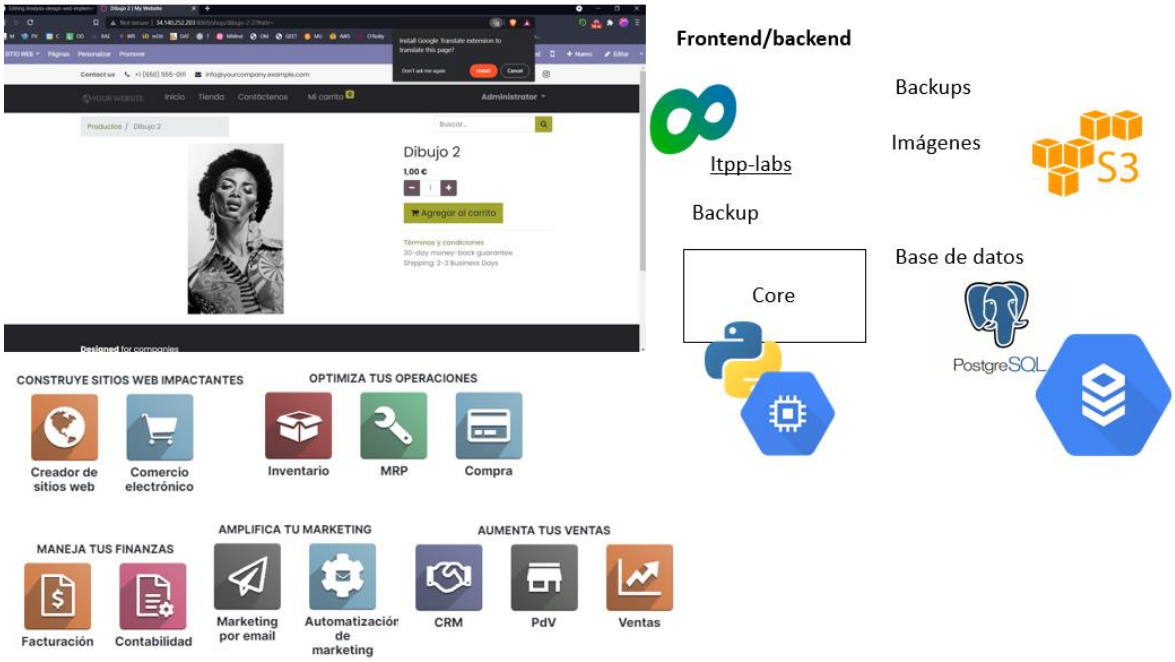


Ilustración 40. Problema de memoria efímera en cloud run y arquitectura con compute engine

Como se observa en la captura anterior, como odoo ce es sólo un microservicio y almacena en memoria la sesión, como Google cloud run es efímero, a los 5 min se pierda la sesión. Se añadió un addon que permitía guardar la sesión en la bbdd pero no se obtuvo los resultados esperados. Y por otro lado, a diferencia que en aws, cloud run no soporta nativo un sistema para almacenar la memoria sino que sólo posee datastore, que necesita implementar cada método de guardado de sesión.

Como solución se plantea desplegar odoo en uno nodo vm como paso previo a desplegarlo en alta disponibilidad configurando un scale group o desplegarlo en kubernetes. Como se observa en la foto de costes si en tres días consume 7 dólares va a consumir 70 el vm y la bbdd, lo cual sobrepasa el presupuesto de las pymes.

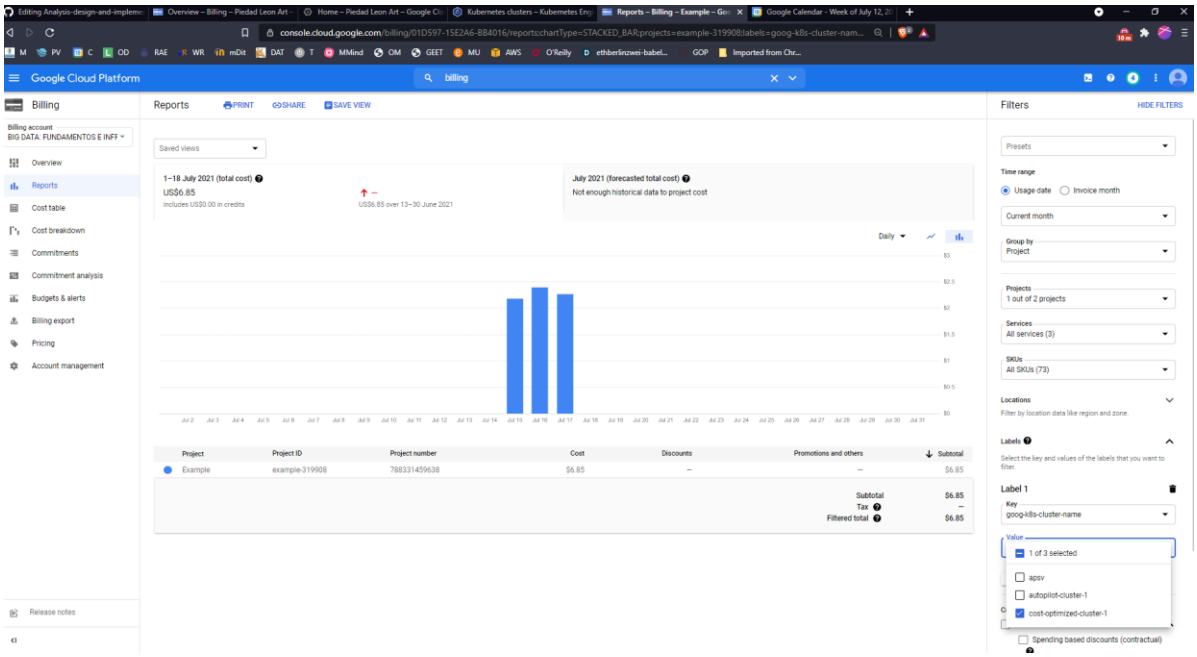


Ilustración 41. Costes de VM en GCP

Además de que se realiza un análisis de carga con blazemeter y se observa como a partir de 25 usuarios concurrentes la vm deja de funcionar y responde 404. En cambio, en name cehap un hosting de wordpress sigue y no sólo, sino que deja de atender a peticiones de probador de carga porque lo identifica como un denial of service

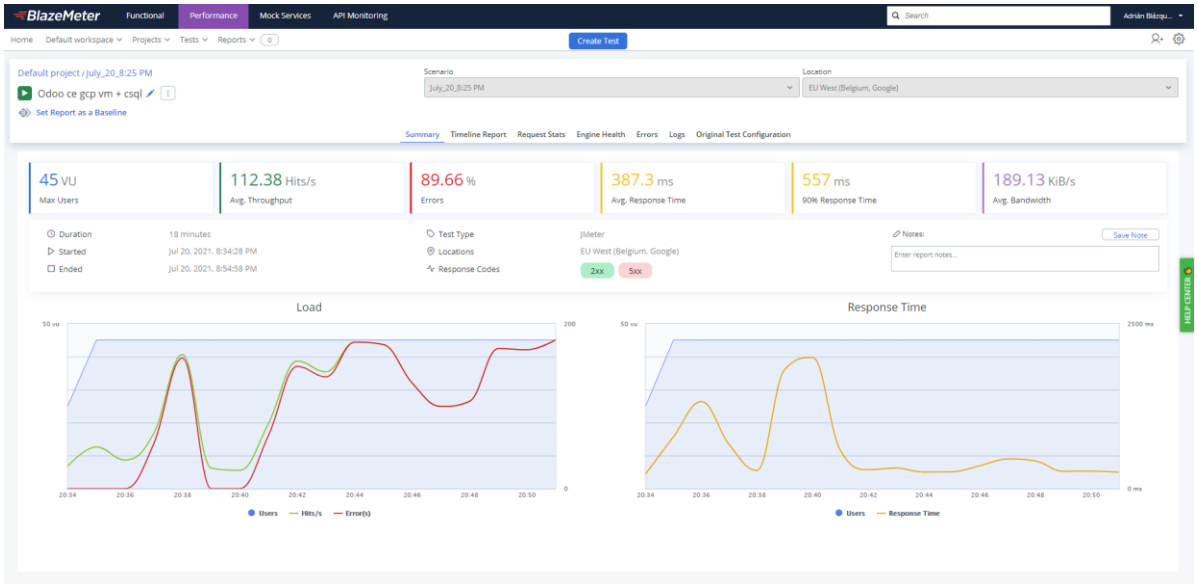


Ilustración 42. Análisis de carga para vm + cloud sql odoo ce

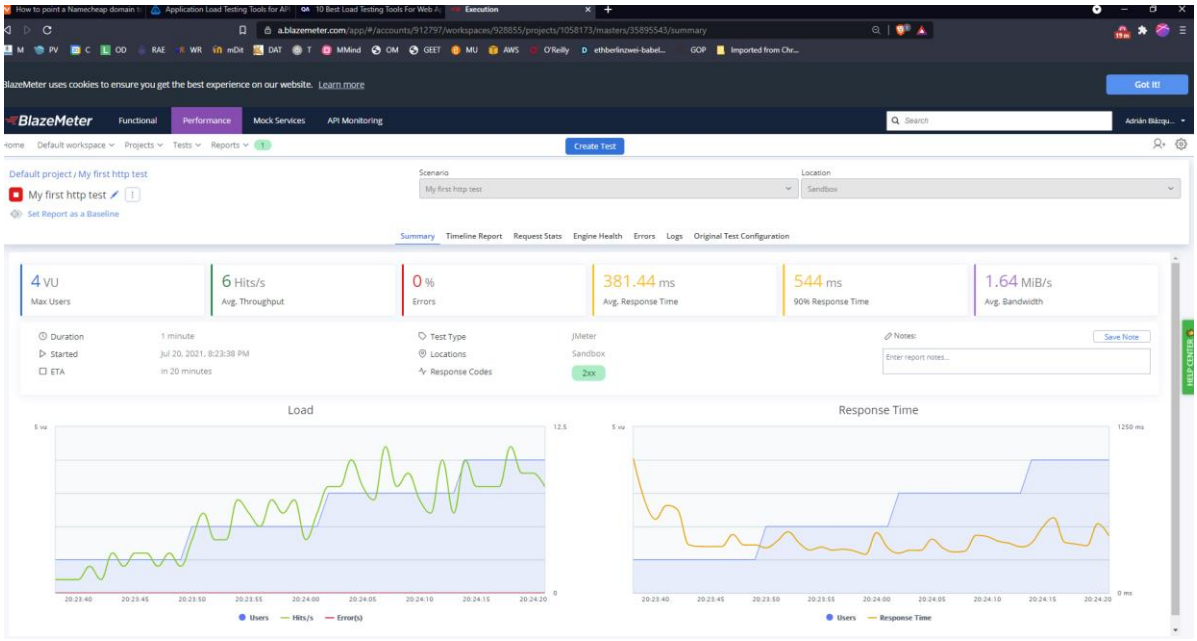


Ilustración 43. Análisis de carga para woocommerce en namecheap

Demo: https://www.youtube.com/watch?v=TR54jjVPmIY

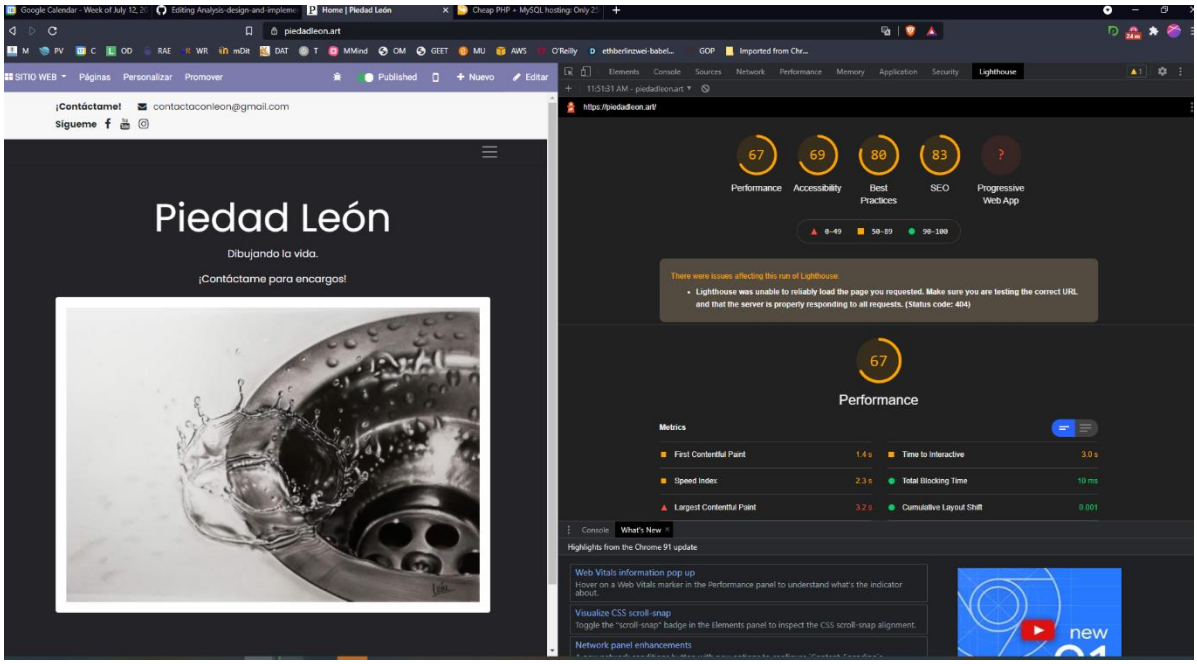


Ilustración 44. Ligthhouse sobre el artefacto de la fase 4

5.5.3 Guía de instalación

En el siguiente repositorio de github se detallan distintos procedimientos sobre el despliegue en crun + csql : [35]

- **Creación manual de un servicio en cloud run + csql**

- **Procedimiento de backup**
- **Procedimiento de actualización**
- **Usar http/2 para permitir backups mayores de 32 MB**
- **Cloud run es stateless, no persiste los volumes**
- **Instalación de odoo en una vm**

Desarrollo brevemente cada uno a continuación:

- **Creación manual de un servicio**

CE 14

Código de odoo

Fork de odoo/docker y configurar cloudbuild para que cuando cambie el repo (con las actualizaciones) se cree otra vez la imagen. Es mejor que coger un contenedor de bitnami, pues bitnami instala postgres en la misma docker network.

Configurar una github acción para que se haga pull del docker maestro.

Poner como default la branch master.

Crun

Se elige la región con menos latencia, Beligum (gcping) y los valores del contenedor.

Se configuran dos variables de entorno: host con la ip de la ddbb y dbport con el puerto de posgres. Nótese que se debe cambiar el valor port pro dbport en entrypoint.sh para que no colisione con el valor or defecto del port que expone cloud run

Si se configura http2 aparece este error:

```
upstream connect error or disconnect/reset before headers. reset reason: protocol error
```

entreypoint.sh y odoo.conf => dbport = 5432

Puede ser que este error sea debido a llamar una revisión con un nombre ya usado o a llamar un repo en mayúscula.

```
Your build failed to run: generic::invalid_argument: invalid build: invalid image name "eu.gcr.io/breadfree/myOdoo/bf-
```

```
test:a93ef07e1398c76d1c8bf55ae6e59cd73db3b4d0": could not parse reference: eu.gcr.io/breadfree/myOdoo/bf-test:a93ef07e1398c76d1c8bf55ae6e59cd73db3b4d0
```

Csql

Primero se reflexiona sobre qué instancia es mejor coger.

Luego sobre cómo conectar cloudsql con odoo, básicamente exponiendo parámetros como variables de entorno y cambiando parámetros del fichero de configuración. Como la clave maestra

Después se reflexiona sobre cómo posibilitar más conexiones al servicio.

Con la configuración en multiaz se espera 42.5€/m. No se configura backup pues se realiza este de forma externa. Se elige shared-core 1.7 GB

Primero, se descomenta la línea admin_password, escribiendo una de tal forma que esta password coincida con la password de odoo.conf.

Además se elige el Docekrfile como ./14.0/Dockerfile y la rama como master.

```
Database connection failure: could not connect to server: Connection timed out
```

Después de reflexionar, se crea un usuario odoo con contraseña odoo, para permtir que la aplicación se conecte a la bbdd. Como mediante cloudshell con ip privada dice que error, se da una ip pública para crear el usuario.

```
ERROR: (gcloud.sql.connect) It seems your client does not have ipv6 connectivity and the database instance does not have an ipv4 address. Please request an ipv4 address for this database instance.
```

Se cambian el puerto a 8069, y se introduce la variable de entorno PORTDB.

La password puede ser odoo u otra

```
\du

 create user odoo with password
'J2Gpv74A5yGPk735FstW9aBwEKGPetneCkCXADG6wFD8hh2BY7rUrVVuZwvqZ4ds';

 \du

 ALTER USER odoo WITH CREATEDB;

 \du
```

```
postgres=> create user odoo with password 'odoo';
CREATE ROLE
postgres=> \du
                                   List of roles
         Role name         |                         Attributes                         |     Member of
---------------------------+------------------------------------------------------------+--------------------
 cloudsqladmin             | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 cloudsqlagent             | Create role, Create DB                                     | {cloudsqlsuperuser}
 cloudsqliamserviceaccount | Cannot login                                               | {}
 cloudsqliamuser           | Cannot login                                               | {}
 cloudsqlimportexport      | Create role, Create DB                                     | {cloudsqlsuperuser}
 cloudsqlreplica           | Replication                                                | {pg_monitor}
 cloudsqlsuperuser         | Create role, Create DB                                     | {pg_monitor}
 odoo                      |                                                            | {}
 postgres                  | Create role, Create DB                                     | {cloudsqlsuperuser}

 create user odoo with password 'odoo';
 \du
 ALTER USER odoo WITH CREATEDB;
```

Ilustración 45. Código para crear roles en postgres cloud sql, odoo

Se pone accesible la bbdd. Se hace primero exponiéndola públicamente creando una red a la que acceden todos (0.0.0.0/0). Luego, se mejora la privacidad del diseño configurando la bbdd como privada, creando un conector vpc serverless.

Después de esto, como mejora de seguridad, se vuelve a hacer un fork de odoo/docker, ahora privado. Después se configura github actions para que haga pull del repo. Así se deben crear dos servicios en cloud run: uno de test con una github action automática y uno de prod con una github action manual. Luego se reflexiona que si lso cambios pasan los tests de odoo, parece razonable sólo hacer una action y que esta sea automática. Se debe disponer de una manera de comprobar que una actulización ha hecho caer el servicio. Por ejemplo, para cualquier incidencia consultar a un correo.

Nos percatamos de que al crear una bd aparece el siguiente error, que se subsana dotando de ciertos privilegios al usuario odoo.

```
Database creation error: permission denied to create database
```

[1](https://www.odoo.com/es_ES/forum/ayuda-1/programmingerror-permission-denied-to-create-database-64086)

```
ALTER USER odoo WITH CREATEDB;
```

Se quita finalmente el acceso público para comprobar que funciona con este servicio privado.

Cambiar password odoo en el conf por una autogenerada

Para borrar una database:

- 1 se le da el role de postgres al usuario odoo que posee la database, desde el role posgres
- 2 se hace a postgres owner de la db. Porque si no odoo conectado a breadfree no prodía borrarlo
- 3 postgres borra la db breadfree

```
postgres=> grant postgres to odoo;
GRANT ROLE
```

```
 gcloud sql connect db9 --user=odoo --database=breadfree --quiet
```

```
breadfree=> alter database breadfree owner to postgres;
ALTER DATABASE
```

```
postgres=> drop database breadfree;
DROP DATABASE
```

Aun así no funciona, se usa otra db y ya está

https://stackoverflow.com/questions/26684643/error-must-be-member-of-role-when-creating-schema-in-postgresql

```
postgres=> drop database breadfree;
```

Configurar

```
Error de usuario:
```

```
Odoo is currently processing a scheduled action.
Module operations are not possible at this time, please try again later or contact your system administrator.
```

Se reinicia y funciona.

```
postgres=> SELECT pg_terminate_backend (pid)
FROM pg_stat_activity
WHERE pg_stat_activity.datname = 'breadfree';
ERROR:  must be a member of the role whose process is being terminated or member of pg_signal_backend
```

Desde breadfree

CE 13 + session store (redis) + s3 (backup and images)

Código de odoo

- 1. Fork de odoo/docker y configurar cloudbuild para que cuando cambie el repo (con las actualizaciones) se cree otra vez la imagen..

- 2. Configurar una github acción para que se haga pull del docker maestro.

- 3. Poner como default la branch master.

Crun

- 1. Se elige la región con menos latencia, Belgium (gcping) y los valores del contenedor.

- 2. Se configuran dos variables de entorno: host con la ip de la ddbb y dbport con el puerto de posgres. Nótese que se debe cambiar el valor port pro dbport en entrypoint.sh para que no colisione con el valor por defecto del port que expone cloud run

Csql

- 1. Primero se reflexiona sobre qué instancia es mejor coger.

- 2. Luego sobre cómo conectar cloudsql con odoo, básicamente exponiendo parámetros como variables de entorno y cambiando parámetros del fichero de configuración. Como la clave maestra

- 3. Después se reflexiona sobre cómo posibilitar más conexiones al servicio.

- 4. Con la configración en multiaz se espera 42.5€/m. No se configura backup pues se realiza este de forma externa. Se elige shared-core 1.7 GB

- 5. Primero, se descomenta la línea admin_password, escriebiendo una de tal forma que esta password coincida con la password de odoo.conf.

- 6. Además se elige el Docekrfile como ./13.0/Dockerfile y la rama como master.

```
Database connection failure: could not connect to server: Connection timed out
```

Después de reflexionar se crea un usuario odoo con contraseña odoo, para permtir que la aplicación se conecte a la bbdd. Como mediante cloudsehll con ip privada dice que error, se da una ip pública para crear el usuario.

```
ERROR: (gcloud.sql.connect) It seems your client does not have ipv6 connectivity and the database instance does not have an ipv4 address. Please request an ipv4 address for this database instance.
```

[1](https://www.postgresql.org/docs/8.0/sql-createuser.html)

Se cambian el puerto a 8069, y se introduce la variable de entorno PORTDB.

La password puede ser odoo u otra más grande

```
 \du
 create user odoo with password 'J2Gpv74A5yGPk735FstW9aBwEKGPetneCkCXADG6wFD8hh2BY7rUrVVuZwvqZ4ds';
 \du
 ALTER USER odoo WITH CREATEDB;
 \du
```

Mismo Código que la ilustración anterior.

Se pone accesible la bbdd. Se hace primero exponiéndola públicamente creando una red a la que acceden todos (0.0.0.0/0). Luego, se mejora la privacidad del diseño configurando la bbdd como privada, creando un conector vpc serverless.

Después de esto, como mejora de seguridad, se vuelve a hacer un fork de odoo/docker, ahora privado. Después se configura github actions para que haga pull del repo. Así se deben crear dos servicios en cloud run: uno de test con una github action automática y uno de prod con una github action manual. Luego se reflexiona que si lso cambios pasan los tests de odoo, parece razoanble sólo ahcer una action y que esta sea automática. Se debe disponer de una manera de comprobar que una actulización ha hecho caer el servicio. Por ejemplo para culaquier incidencia consultar a un correo.

Nos percatamos de que al crear una bd aparece el siguiente error, que se subsana dotando de ciertos privilegios al usuario odoo.

```
Database creation error: permission denied to create database
```

```
ALTER USER odoo WITH CREATEDB;
```

Se quita finalmente el acceso público para comprobar que fucniona con este servicio privado.

Cambiar password odoo en el conf por una autogenerada

Para borrar una database:

- 1 se le da el role de postgres al usuario odoo que posee la database, desde el role posgres
- 2 se hace a postgres owner de la db. Porque si no odoo conectado a breadfree no prodía borrarlo
- 3 postgres borra la db breadfree

```
postgres=> grant postgres to odoo;
GRANT ROLE
```

```
breadfree=> alter database breadfree owner to postgres;
ALTER DATABASE
```

```
postgres=> drop database breadfree;

DROP DATABASE


postgres=> drop database breadfree;
```

Redis

Se añade el código de camptocamp. Se añade al script las dependencias necesarias. Lo instalo, pero con el requirement no sale. bdist wheel en el camp2camp

[1](https://github.com/aws/aws-cli/issues/4243), [2](https://stackoverflow.com/questions/34819221/why-is-python-setup-py-saying-invalid-command-bdist-wheel-on-travis-ci/49426845). [3](https://realpython.com/python-redis/) The requirment says it is needed redis 2.10.5, why it cannot be used a redis higher? WHich si the redis version that mathces? Hay un pyaml there. Se itnetna con un redis 4. Sino se va montando un 13 [4](https://cloud.google.com/memorystore/docs/redis/scaling-instances) Scaling is a bit hard

[5](https://cloud.google.com/memorystore/docs/redis/pricing) +32€. ¿Por qué no se ven los módulos, sólo si los hago uno a uno sí? no aparecen los módulos aunque la db sea nueva => nos módulos extra.

```
Could not get content for /website/static/src/scss/options/user_values.custom.web.assets_common.scss defined in bundle 'web.assets_common'.
```

- **Procedimiento de backup**

Se plantean dos ocpiones:

Primero, con un scheduler ejecutar cloud run. Pero puede que el script dependa de la versión y además será un código complicado y que haya que mantener.

- Addons

Existen addons compatibles gratuitos

[1](https://apps.odoo.com/apps/modules/14.0/odoo_backup_sh/)

[2](https://apps.odoo.com/apps/modules/14.0/ir_attachment_s3/)

Montarlos sobrer el Dockerfile. Hacer un Dockerfile con este. Crear una db6 de pruebas inyectándola la password.Añadirle a este Dockerfile el addon de backup. Meter las credenciales de aws en ello. Configurarla cómo funciona.

Aparece el siguiente error:

```
Skipping database bf-dev because of modules to install/upgrade/remove.
```

Se propone arranca la db, y entocnes instalar y darle a udpate.

No hay bd, no hay ningún error, todo puede ser por haber clikado dos veces en crear bd. Se crea un bd8 y se borra esta.

```
Imposible instalar el módulo "odoo_backup_sh" porqué hay una dependencia externa no resuelta: Python library not installed: boto3
```

Se intenta hacer run de pip install. En la documentation se dice que se debe elegir un bucket un un folder. Se intenta crear un folder en el bucket, pero no se sabe si se refiere a un folder en el bucket o en el contenedor de odoo.

Parece que la región más barata es us-east-1, N Virginia. Además no tiene gran latencia.

Cuando descargo el zip dice que no tiene permiso.

```
The request signature we calculated does not match the signature you provided. Check your key and signing method.
```

Parece que el error era debido a que el nombre del bucket de s3 no debe cotnener barras baja.

- **Procedimiento de actualización**

Se plantea una actualización automática, usando ramas del repositorio y github actions

- **Usar http/2 para permitir backups mayores de 32 MB**

Se pregunta y elabora este hecho en mi siguiente pregunta en stack overflow: [36]

Q: “¿Cómo configurar en un Dockerfile un nginx frente al mismo contenedor (google cloud run)? Intenté hacer que un servidor python (odoo) manejara http / 2”

“

Mi pregunta es, ¿cómo configurar en un Dockerfile un nginx frente a un contenedor? Vi otras preguntas [5], y parece que la única forma de permitir http / 2 en odoo en la ejecución en la nube es crear un contenedor nginx, ya que los sidecars no están permitidos en gcrun. Pero también leí que con el supervisor se puede hacer. ¿Alguien ha podido hacer eso para manejar http / 2 y así aumentar la cuota máxima de solicitudes de ejecución en la nube?

Quería probar esto: en entrpoint.sh, escriba un comando para instalar nginx y luego establezca su configuración para que se use como un proxy para permitir http2. Pero le pregunto aquí porque no estoy seguro de si funcionará, ya que leí en [2] que nginx no funcionará con un servidor Python.

Toda la historia: estoy implementando un odoo ce en google cloud run + cloud sql. Configuré un odoo como una demostración con 5 publicaciones de blog y cuando intenté descargarlo, dice que la solicitud era demasiado grande. Entonces, imaginé que esto se debía a la cuota de solicitud de ejecución en la nube de 32 MB [1], ya que el tamaño de la copia de seguridad era de 52 MB. Luego, vi que la cuota para conexiones http / 2 era ilimitada, así que activé el botón http / 2 en la ejecución en la nube. A continuación, cuando accedí al servicio, apareció un error relacionado con "fallo de conexión".

Para hacerlo, pensé en dos formas: una, estaba actualizando el servidor http de odoo a uno que pueda manejar http / 2 como Quark. Esta primera forma me parece imposible, porque me obligaría a reescribir muchas piezas de odoo, tal vez. Luego, la segunda opción en la que pensé fue ejecutar

frente al contenedor odoo (que ejecuta un servidor web Python en un Werkzeug), un nginx. Leí en la web que nginx podría actualizar las conexiones a http / 2. Pero también leí que Cloud Run estaba ejecutando su balanceador de carga interno [2]. Entonces, entonces mi pregunta: ¿sería posible ejecutar en el mismo contenedor odoo un nginx que expone este servicio en la nube?

Referencias:

[1] https://cloud.google.com/run/quotas

[3] https://linuxize.com/post/configure-odoo-with-nginx-as-a-reverse-proxy/

[4] https://github.com/odoo/docker/tree/master/14.0

[5] ¿Cómo configurar nginx frente al nodo en la ventana acoplable para Cloud Run?

A: "¿Alguien ha podido hacerlo para manejar HTTP / 2 y para aumentar la cuota de solicitud MAX Run Max?"

Manipular HTTP / 2 no le ayuda a aumentar sus solicitudes máximas por límite de contenedores en la Cloud Run.

HTTP / 2 solo le ayuda a reutilizar las conexiones TCP para enviar solicitudes concurrentes a lo largo de una sola conexión, pero Cloud Run no realmente cuenta las conexiones para que no esté en la pista correcta aquí. Http / 2 no te ayudará.

Cloud Run hoy ya apoya 1000 instancias de contenedores (con 250 solicitudes concurrentes en vista previa), por lo que son 250,000 solicitudes simultáneas de su servicio. Si necesita más, contacte al soporte.

"Pero, le pregunto aquí, ya que no estoy seguro de si funcionará, ya que leí en [2] que NGINX no funcionará con un servidor de Python."

Suena incorrecto.

Si configura un contenedor de multi-proceso, puede ejecutar Python detrás de Nginx en Cloud Run. Pero como dijiste, Cloud Run no necesita NGINX.

En general, no necesita HTTP / 2 en este escenario.

- **Cloud run es stateless, no persiste los volumes**

Q: ¿Ha encontrado una manera de conservar los módulos principales de odoo en v14 de forma diferente en un volumen? Entonces, ¿es posible implementar odoo en gcloud run?

Quiero implementar ODOO lo más barato posible. Lo intenté con GCLOUD SQL (15-30 € / m) + Cloud Run. Pero después de que pasaron unos minutos, la interfaz ODOO me muestra una pantalla blanca con tantos registros en la consola similar a esta:

```
GET4041.04 KB24 msChrome 91 https://bf-dev3-u7raxlu3nq-ew.a.run.app/web/content/290-f328144/1/website.assets_editor.css
```

Mi interpretación es que, como la Cloud Run es sin estado, y los archivos estáticos web parecen estar almacenados en el módulo central, después de que el contenedor se mata, esta información se pierde. Como he estado trabajando un mes en busca de una solución, antes de probar cualquier otra forma de implementar, le pregunto a la comunidad: ¿Ha encontrado una forma de persistir en los módulos de núcleo ODOO en V14 diferentes forman un volumen? ¿Y así es posible desplegar ODOO en GCLOUD RUN?

Aquí escribí todas las ideas que probé:

Primero, pensé que estos archivos CSS fueron almacenados en la sesión de WerkzeUg, así que probé dos adictos que almacenaron esta sesión en un lugar diferente al FileStore. Estos complementos fueron Camptocamp Odoo-Cloud-Platform-14.0 / Session-Redis y Misc-Addsons-13.0 / base_session_store_psql. Pero, entonces el problema persistió.

Luego leí que el archivo estático CSS y JS generado en el editor web se almacenan en ODOO como archivos adjuntos, y los complementos Misc-Addons-13.0 / IR_ATTACHMENT_S3 podrían almacenar estos archivos en S3. Pero, aunque configuré este complemento, el problema persistió.

A continuación, encontré este enlace que describe la necesidad de regenerar los activos para que se almacenen en el DB. Pero, aunque hice que el problema persistió.

Finalmente, pensé en implementar ODOO de otras maneras. La forma de directamente en una máquina virtual parece ser la más minimalista y estándar, y parece que las más posibilidades de trabajar, aunque será difícil implementar gitops. Se pueden implementar contenedores en la VM a través de Docker, componga lo que ayudará a implementar actualizaciones. GKE Anthos parece implementar gitops también y parece persistir volúmenes, pero en la descripción muestra a GKE

Anthos es apátrida. Finalmente, está la forma de desplegarse en un clúster de K8S, de esta manera implementará contenedores y permitirá autoscalizar VS la forma de composición DOCKER en una VM. Pero es cierto que parece ser más caro y es más difícil de implementar. Respecto a parecen ser más caros, se piensa en probar pocas máquinas de nodos de trabajo, por lo que el costo permanece pequeño durante la noche. Con respecto a la dificultad de implementar, se desea implementar gitops, por lo que parece que se debe agregar Argo u otro. Además, escuché a Gke AutoPilot tiene un buen nivel libre y es más fácil de implementar.

A: Cloud Run no es la buena solución para eso. De hecho, si la sesión de WerkzeUg está persistida en la memoria, el mismo cliente no está seguro de acceder a la misma instancia cada vez y, por lo tanto, perdió el archivo incluso en medio de una sesión.

La mejor solución es usar VM con la configuración de la sesión pegajosa. Puede usar el despliegue de la escuela antigua en el motor de cómputo, o la solución nativa de la nube con GKE / K8S. Es más o menos el mismo costo si solo tiene 1 clúster (el primero es gratis)

Sólo una corrección sobre Gke Anthos. Creo que hablas de Cloud Run en Anthos, y sí, es como correr en la nube, pero usa KNative en GKE para administrar los contenedores, y también es sin servidor. Pero GKE puede manejar el despliegue de estado, como necesites con Odoo

Q: Tengo una duda final @guillaumeBlaquiere: luego, en la medida en que entiende el punto, ODOO se puede implementar en el motor de cómputo, pero los grupos de autoscalización no serán útiles, ya que no habrá forma de vincular a uno de los datos de VM con otro nuevo datos de VM. Pero en el caso de K8, parece que puede funcionar, ya que con los datos del concepto de volumen podrían compartirse entre las instancias. ¿No es así?

A: Cargar Balancer + VM es aproximadamente igual a la implementación GKE / K8S: la restricción técnica es la misma, solo la parte de implementación / administración es diferente. Por lo tanto, puede usar el mismo disco entre varias VM, pero solo en la lectura. Es lo mismo con GKE. Excepto si usa un sistema compatible con NAS, como el servicio de FileStore, que le permite realizar leer y escribir desde diferentes VM (o PODs, es la misma solución con GKE)

https://stackoverflow.com/questions/68334528/have-you-found-a-way-of-persisting-the-odoo-core-modules-in-v14-different-form-a

Esto lleva a problemas de visualización de js y css](https://stackoverflow.com/questions/68334528/have-you-found-a-way-of-persisting-the-odoo-core-modules-in-v14-different-form-a).No se muestra css](https://www.odoo.com/es_ES/forum/ayuda-1/no-css-no-images-after-user-logout-from-website-127642)

```
Failed to load resource: the server responded with a status of 404
() web.ssets frontend.css:1
```

[si cloud run no persite los volúmenes, los addons se van](https://www.odoo.com/es_ES/forum/ayuda-1/how-to-configure-the-filestore-in-high-availability-172440)

[Con attachments](https://github.com/itpp-labs/misc-addons/tree/11.0/ir_attachment_s3)

[Otros casos](https://www.odoo.com/es_ES/forum/ayuda-1/css-and-js-files-are-not-loading-96868)

[Parece que sí es posible con autoscaling groups](https://dokumen.tips/business/simple-odoo-erp-auto-scaling-on-aws.html) [2](https://www.slideshare.net/lecadoujr/simple-odoo-erp-auto-scaling-on-aws)

Se elige París como localización de bucket: [1](https://www.cloudping.info/), [2](https://aws.amazon.com/s3/pricing/)

```
eb/content/258-3d5e37d/web.assets_common.css:1 Failed to load
resource: the server responded with a status of 404
```

```
Bad Request
```

Session expired (invalid CSRF token)

- **Instalación de odoo en una vm**

A partir de la siguiente documentación se confecciona el siguiente script para desplegar odoo en una máquina virtual:

https://www.odoo.com/documentation/14.0/setup/install.html

https://www.youtube.com/watch?v=ud0z9GB6phs

https://www.cybrosys.com/blog/how-to-install-odoo-14-on-ubuntu-20-04-lts

```
sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get install openssh-server fail2ban -y

sudo adduser --system --home=/opt/odoo --group odoo

sudo apt-get install -y python3-pip

sudo apt-get install python-dev python3-dev libxml2-dev libxslt1-dev zlib1g-dev libsasl2-dev libldap2-dev build-essential libssl-dev libffi-dev libmysqlclient-dev libjpeg-dev libpq-dev libjpeg8-dev liblcms2-dev libblas-dev libatlas-base-dev

sudo apt-get install -y npm

sudo ln -s /usr/bin/nodejs /usr/bin/node

sudo npm install -g less less-plugin-clean-css

sudo apt-get install -y node-less

sudo apt-get install postgresql -y

sudo su - postgres

createuser --createdb --username postgres --no-createrole --no-superuser --pwprompt odoo14
```

psql

```
ALTER USER odoo14 WITH SUPERUSER;

\q

exit

sudo apt-get install git -y

sudo su - odoo -s /bin/bash
```

```
git clone https://www.github.com/odoo/odoo --depth 1 --branch 14.0 --single-branch .

exit

sudo pip3 install -r /opt/odoo/requirements.txt

sudo wget https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.5/wkhtmltox_0.12.5-1.bionic_amd64.deb

sudo dpkg -i wkhtmltox_0.12.5-1.bionic_amd64.deb

sudo apt install -f


sudo cp /opt/odoo/debian/odoo.conf /etc/odoo.conf

sudo nano /etc/odoo.conf

sudo chown odooUser: /etc/odoo.conf

sudo chmod 640 /etc/odoo.conf

sudo mkdir /var/log/odoo

sudo chown odooUser:root /var/log/odoo

sudo nano /etc/systemd/system/odoo.service

sudo chmod 755 /etc/systemd/system/odoo.service

sudo chown root: /etc/systemd/system/odoo.service

sudo systemctl start odoo.service

sudo systemctl status odoo.service


sudo apt install postgresql -y

wget -O - https://nightly.odoo.com/odoo.key | apt-key add -

echo "deb http://nightly.odoo.com/14.0/nightly/deb/ ./" >> /etc/apt/sources.list.d/odoo.list

apt-get update && apt-get install odoo
```

5.5.4 Evaluación

Finalmente, en cuanto al marco de referencia, este artefacto permite todos los subcasos de uso de tienda online y de gestor de operaciones. En cuanto a usabilidad recibe un 3, en cuanto a seguridad, si bien se configura un plugin para que se realicen los backups diariamente, y se crean roles, no se permite por defecto la autenticación de doble factor ni la alerta. En cuanto a precio un 1 con 40€/mes.

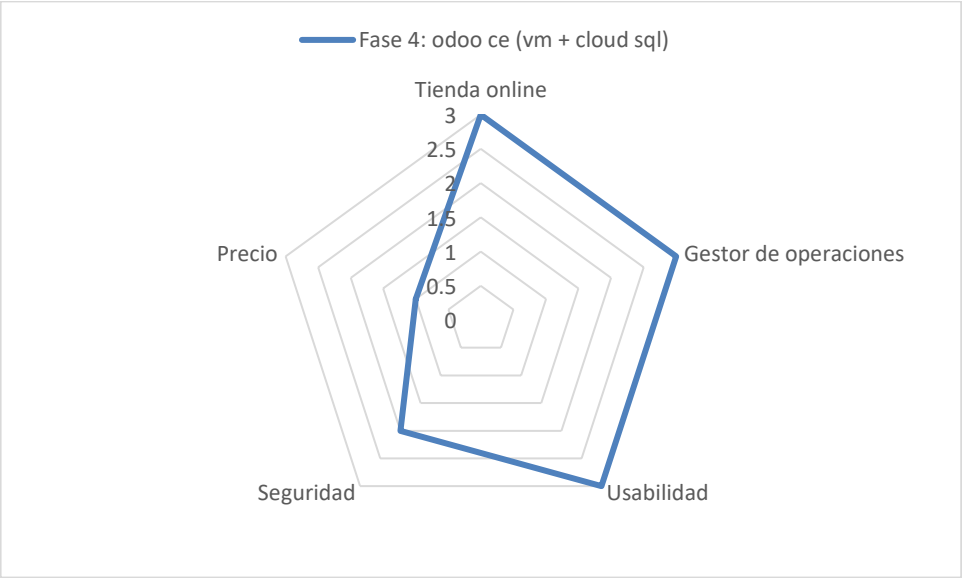


Ilustración 46. Marco comprativo sobre el artefacto de la fase 4

5.6 FASE 5: (WP + WC + PLUGINS) + NAMECHEAP

5.6.1 Análisis

En esta última fase se plantea utilizar WordPress con "Woocommerce" y con los plugins necesarios para ofrecer esos dos casos de uso. Si bien primero se usa el plan starter de 2€ al mes, luego se decide cambiar al plan intermedio turbo para que la web cargue bien tras cargar en ella 1600 productos (entre 69 cuadros * variante de tamaño * variante de añadido cuadro o passepartout)

5.6.2 Diseño

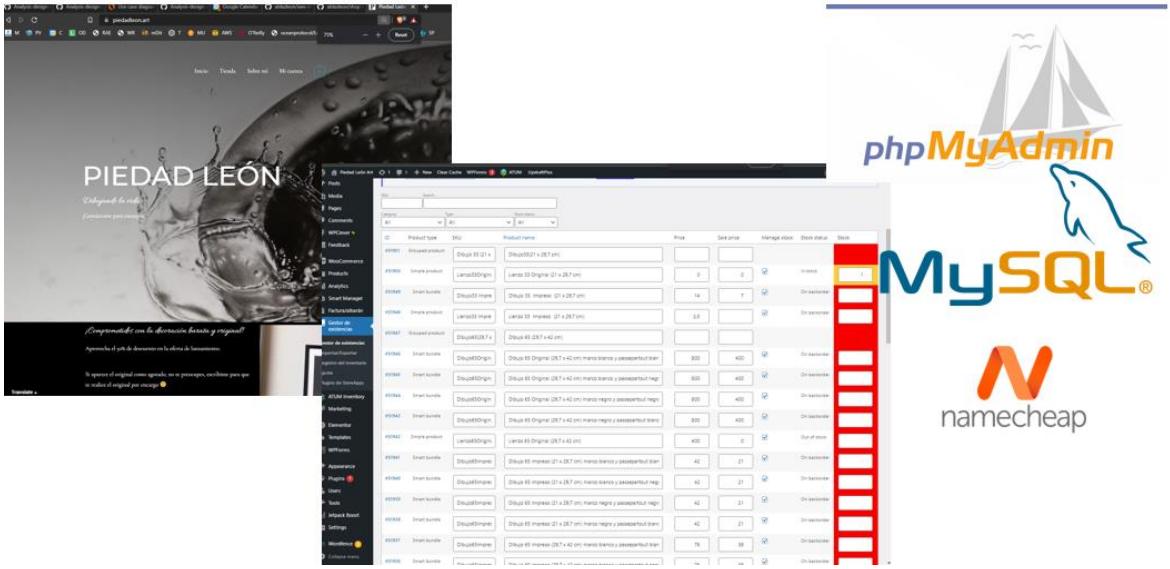
A continuación, se describe cómo se plantea cada caso de uso y qué plugins extra han sido necesario para ello.

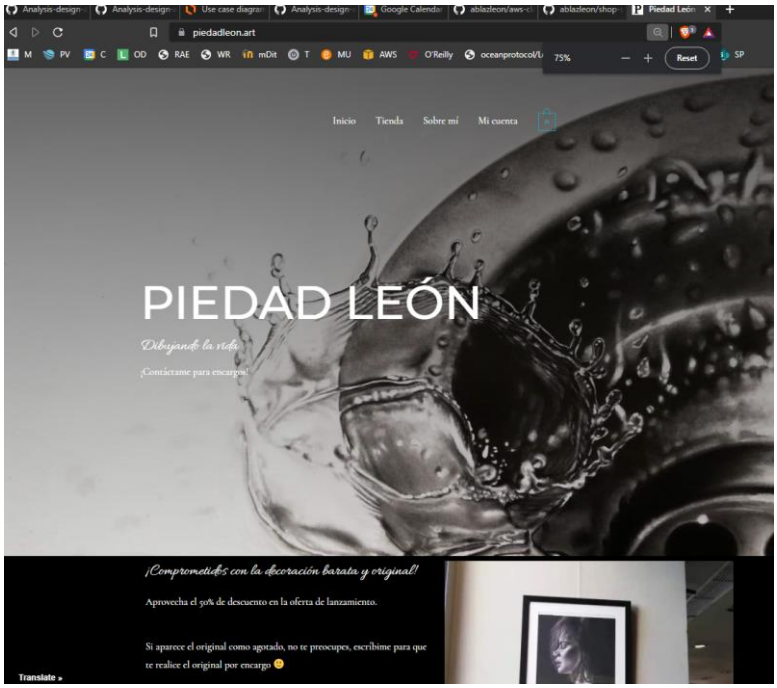
Visualización de catálogo. El usuario llega a piedadleon.art. consulta la landing Page, le informa de una oferta. Decide leer sobre la artista. Decide consultar el catálogo. Puede ver distintos cuadros y

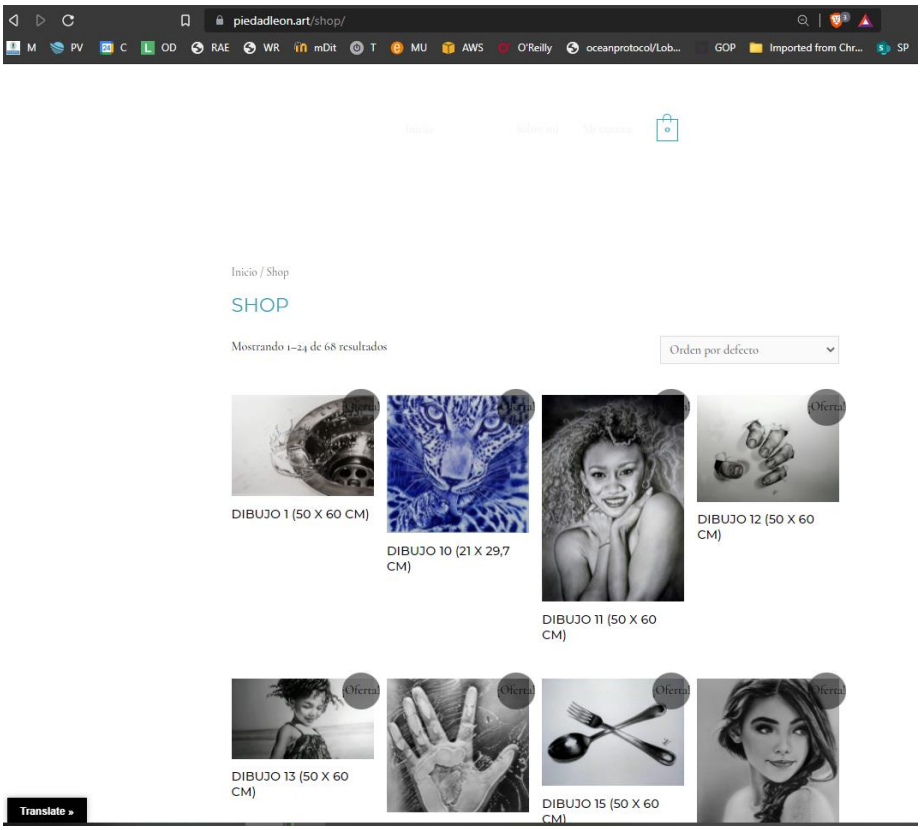
además elegir distintos tamaños y si lo desea impreso o original. El catálogo se ha configurado con woocommerce. La posibilidad de cambiar el lenguaje con un plugin de Google translate.

Storefront

Dashboard







Inicio / Shop / Dibujo en 50 x 60 cm / Dibujo 1 (50 x 60 cm)

DIBUJO 1 (50 X 60 CM)

7,00€ – 600,00€

0	Dibujo 1 Impreso (50 x 60 cm)	~~44,00€~~ 22,00€
0	Dibujo 1 Impreso (29,7 x 42 cm)	~~28,00€~~ 14,00€
0	Dibujo 1 Impreso (21 x 29,7 cm)	~~14,00€~~ 7,00€
0	Dibujo 1 Impreso (50 x 60 cm) marco negro y passepartout blanco	~~108,00€~~ 54,00€
0	Dibujo 1 Impreso (50 x 60 cm) marco negro y passepartout negro	~~108,00€~~ 54,00€
0	Dibujo 1 Impreso (50 x 60 cm) marco blanco y passepartout negro	~~108,00€~~ 54,00€
0	Dibujo 1 Impreso (50 x 60 cm) marco blanco y passepartout blanco	~~108,00€~~ 54,00€
0	Dibujo 1 Impreso (29,7 x 42 cm) marco negro y passepartout blanco	~~76,00€~~ 38,00€
0	Dibujo 1 Impreso (29,7 x 42 cm) marco negro y passepartout negro	~~76,00€~~ 38,00€
0	Dibujo 1 Impreso (29,7 x 42 cm) marco blanco y passepartout negro	~~76,00€~~ 38,00€

Ilustración 47. Arquitectura, landing page y visualizar catálogo

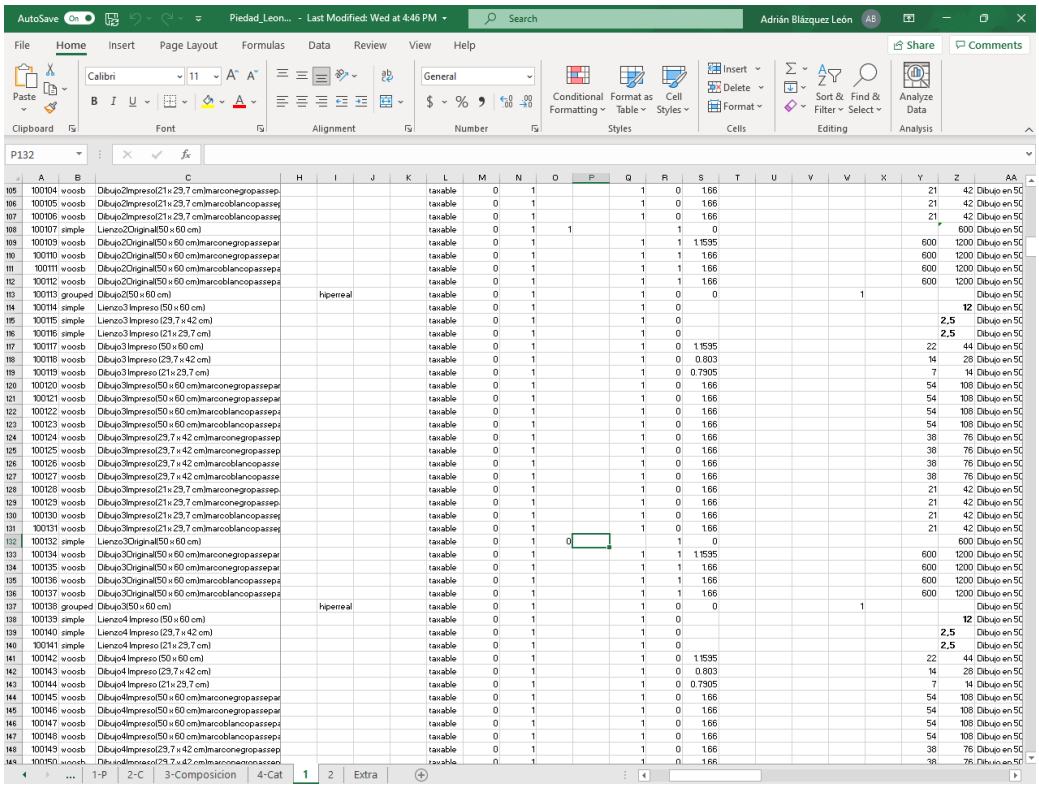


Ilustración 48. Excel en el que se expresan los productos

Sobre esta funcionalidad, nótese que primero se planteó un modelo de datos en el que se desglosaban las variantes tamaño e impreso u original, de forma que el csv salía como de 200 filas. Pero más adelante, se identificó el problema de seguir el stock de productos secundarios como el embalaje, de forma que se vuelve necesario encontrar una forma de expresar el producto dibujo 1 tamaño a2 para que cuando se compre se descuente no sólo el lienzo, sino los otros elementos que lo componen, del stock. Como el tubo, el sello identificativo o la etiqueta de envío. Entonces se plantea usar el plugin wpc clever bundle expresando los productos así, como productos simples en vez de como productos con variantes.

Po otro lado una vez se ha añadido al carrito se puede proceder a la compra. Primero se comprueba el stock por ejemplo del dibujo 1 con la herramienta smart manager, se observa que sale en backorder, y con stock de tubo 1

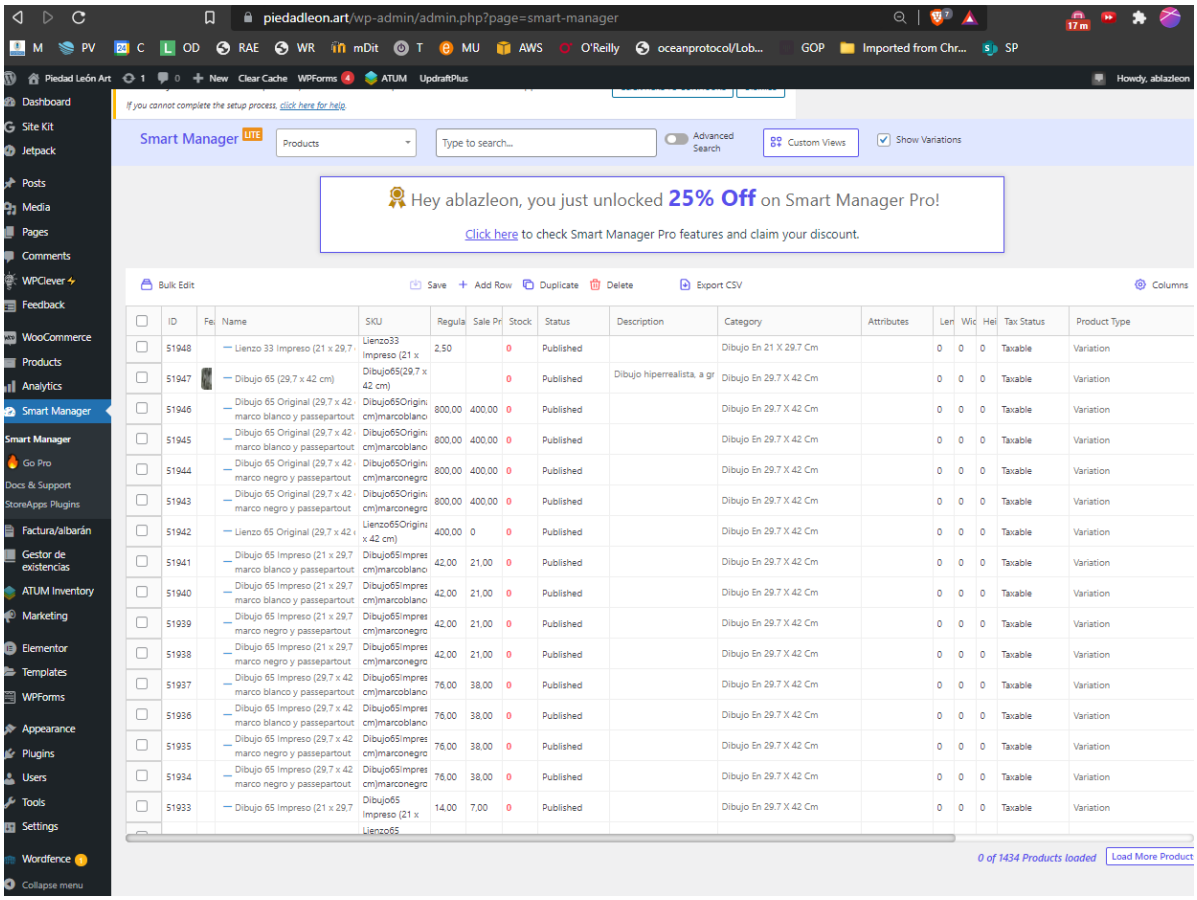


Ilustración 49. Smart manager

Para el caso de dibujos impresos se ponen los elementos como sin límite, porque imprimir un dibujo apenas toma tiempo en comparación con pintarlo. Se finaliza compra, se observa que en woocommerce pagos se ha configurado tres, tarjeta PayPal y stripe, habiendo previamente creado una cuenta bancaria aislada en un banco. Para no pagar cuotas adicionales se opta por el pago en mano para el ejemplo. Se compra, se comprueba que la orden aparece y que llega un correo. Luego se comprueba como el stock de tubo ha disminuido en 1

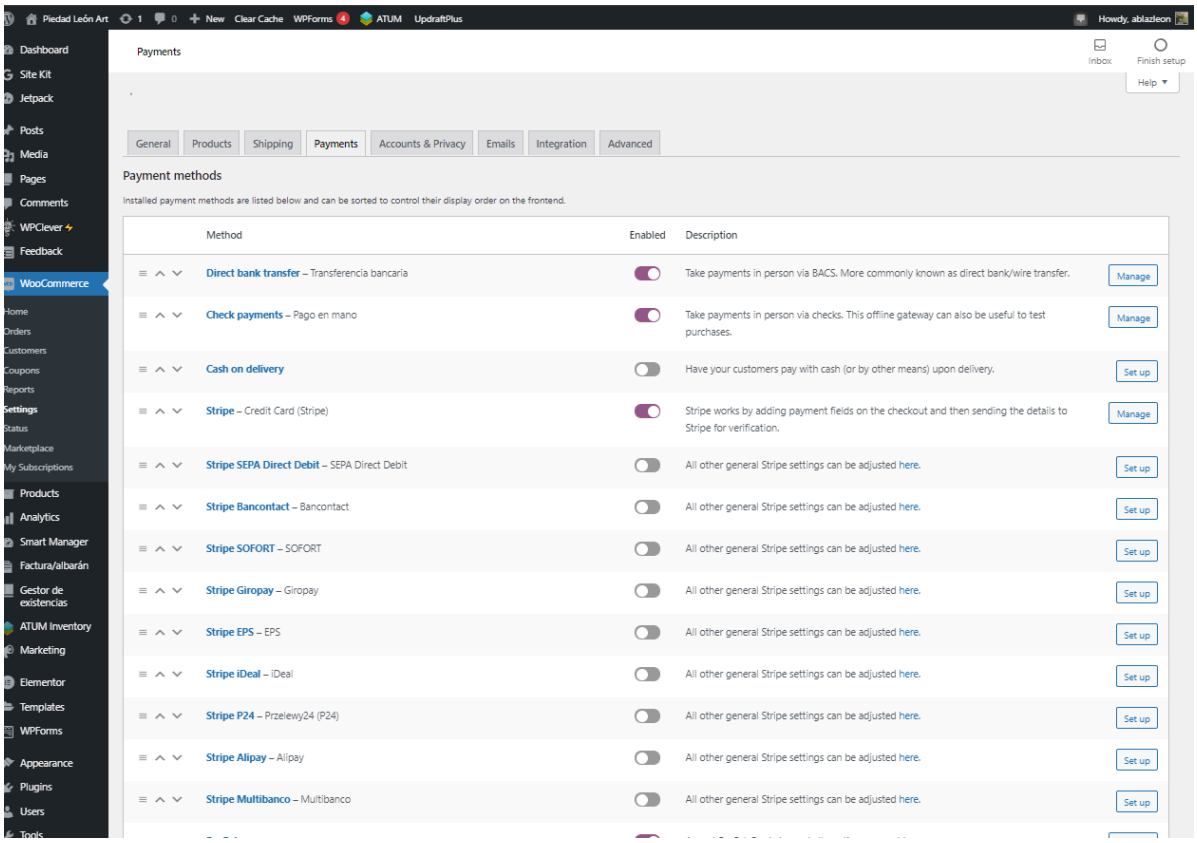


Ilustración 50. Métodos de pago en woocommerce

Por otro lado, para lienzos originales agotados como el 3 se compra y se observa como no se deja comprar más, ni se dejan comprar varios a la vez.

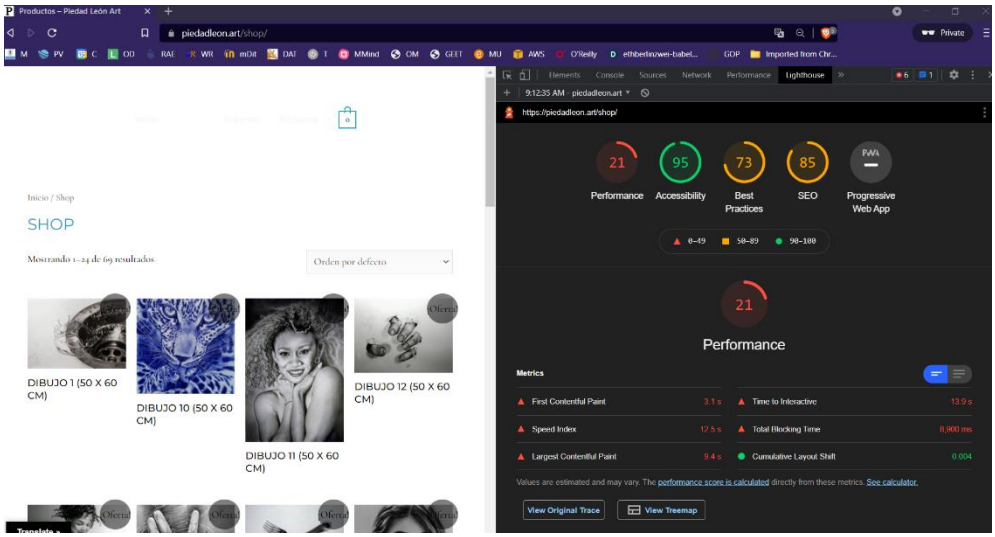
Segundo caso de uso seguimiento, Se ha configurado la cuenta de email de cliente y la del gestor ha llegado el mensaje se procesa la orden llega otro mensaje tanto a cliente como a gestor. Se procesa el pedido y llega otro mensaje.

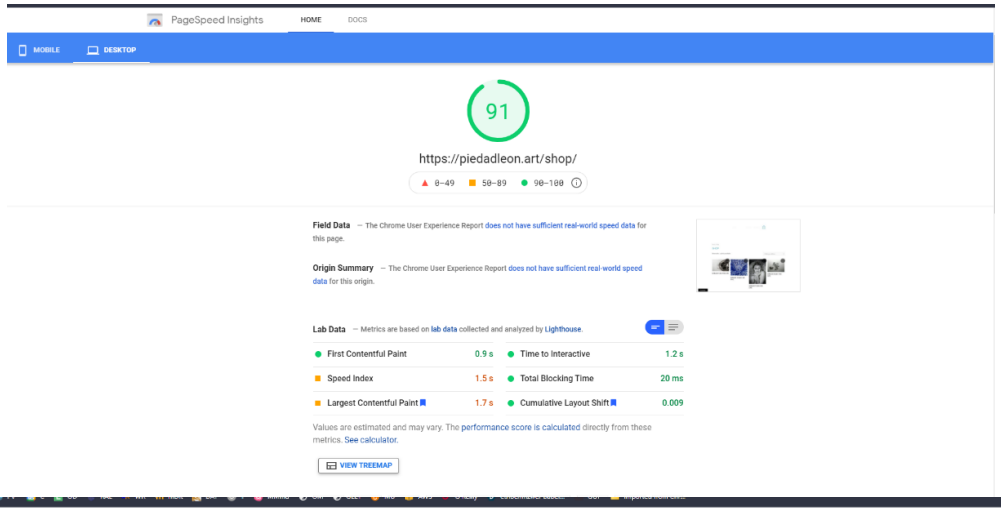
Constructor. Cómo se ha visto se ha creado un usuario gestor de tienda, este puede acceder y mediante un configurador de la tienda, no hace falta ni implementar código html, como por ejemplo elementor, pueden cambiar la apariencia, los mensajes.

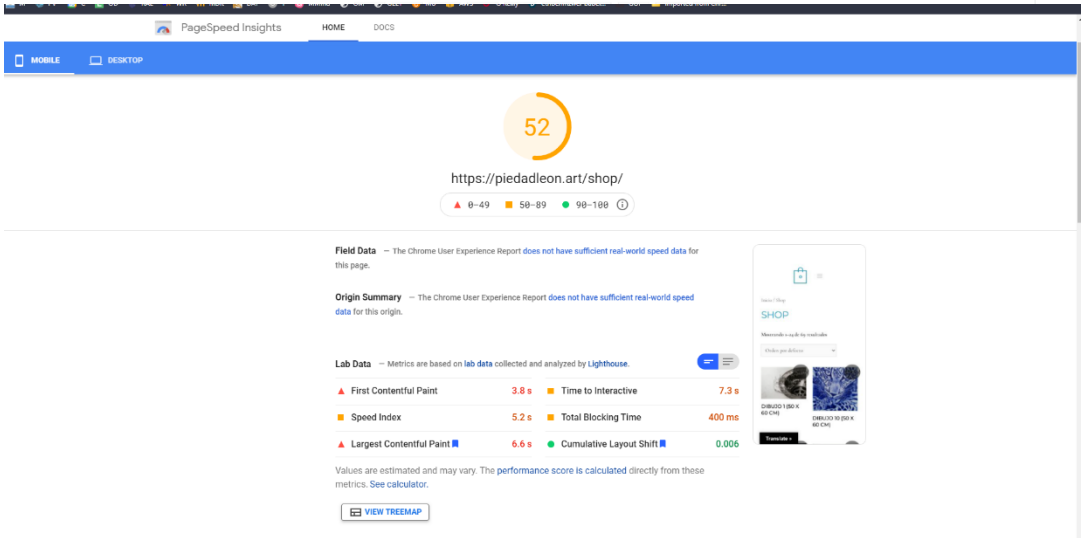
Por último, el siguiente caso de uso de gestor de tienda. Se ha comprobado que se puede visualizar el stock por ejemplo mediante el plugin de gestor de existencias, y se pueden calcular los costes tanto los de envío de manera automática en el proceso de checkout.

Finalmente, se comenten cómo se han implementado los requisitos no funcionales de disponibilidad y seguridad. En cuanto a disponibilidad, se ejecuta lighthouse. Se observan en las siguientes capturas que para la versión del plan starter el rendimiento era menor que para la versión de plan turbo. En cuanto a seguridad, aplicando las recomendaciones de Magerit identificando como activos el control de los datos de la web y los procesos de compra, se plantean salvaguardas recomendadas con Pilar

como garantizar un sistema de autenticación de entradas y 2Fa de acceso. Esto se implementa con el plugin Wordfence.







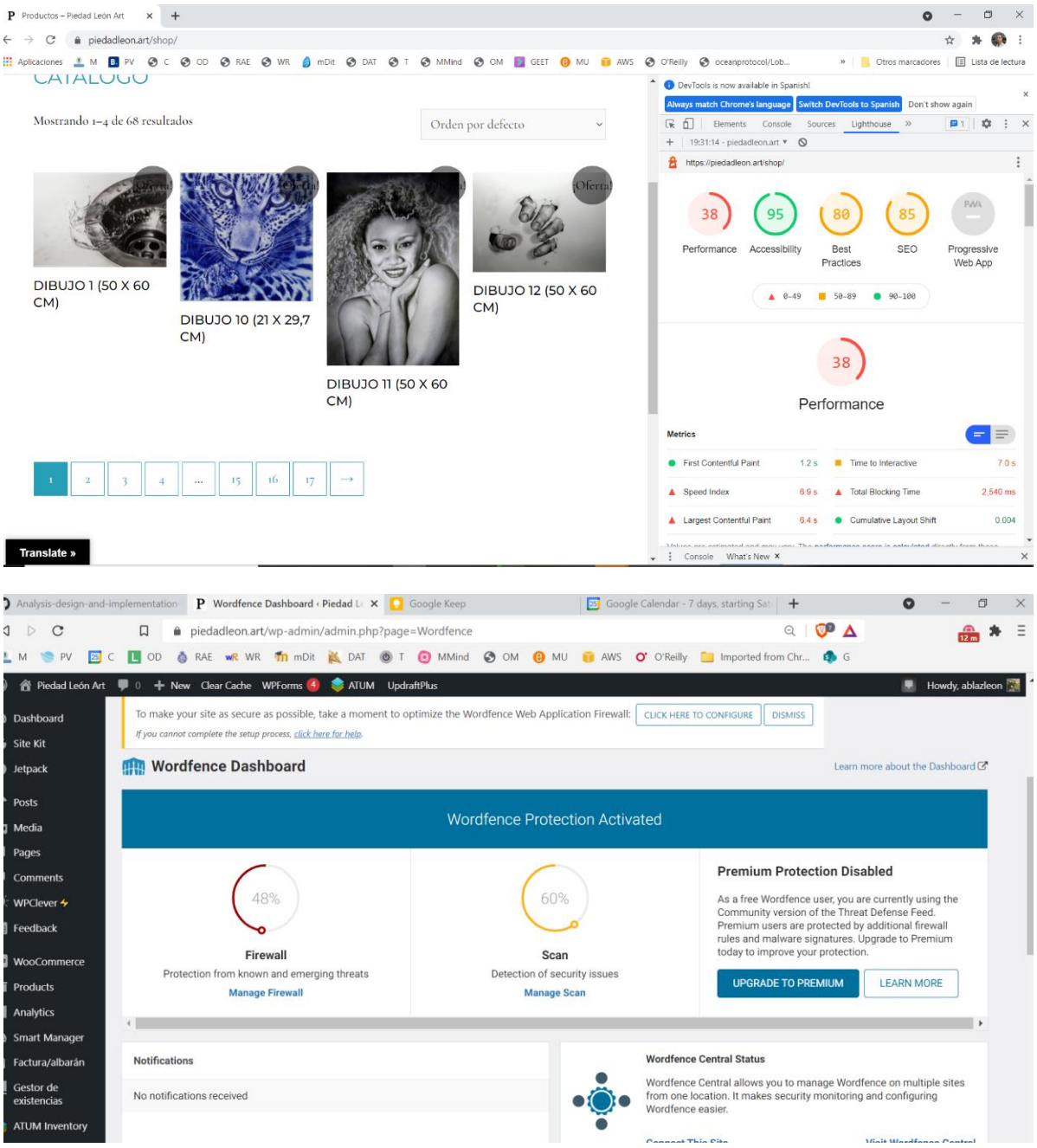


Ilustración 51. Rendimiento en lighthouse, primero de la versión starter y dos siguientes de la versión turbo. Y captura sobre el plugin de wordfence

5.6.3 Guía de instalación

Para instalar una solución así, primero se debe acceder al hosting y configurarlo, en este caso namecheap. Después, a partir del servicio wordpress inicial, se instalan los plugins que se han comentado en el apartado anterior, y se importan el csv de los productos confeccionado.

5.6.4 Evaluación

Finalmente, en cuanto al marco de referencia, esta solución implementa ambos casos de uso, y como se observa en las imágenes tiene un score de usabilidad alto. En cuanto a seguridad cumple con los tres requisitos y en cuanto a precio no sube de los 4€/mes.

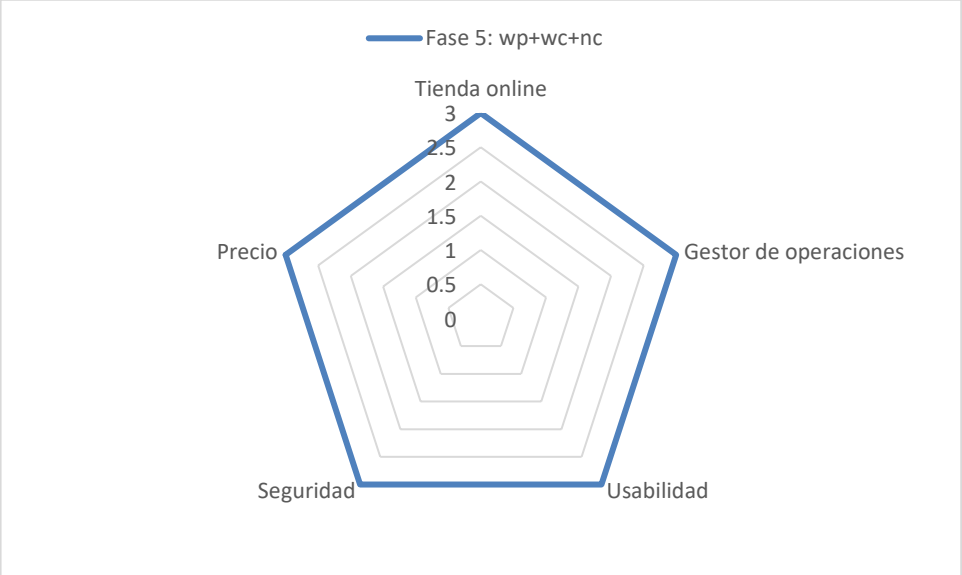


Ilustración 52. Comparativa sobre el marco de referencia artefacto de la fase 5

6 CONCLUSIONES

6.1 CONCLUSIONES

Este proyecto se concluye describiendo qué se ha conseguido: primero retomando los objetivos y luego en general describiendo qué retos he superado.

a. Objetivos

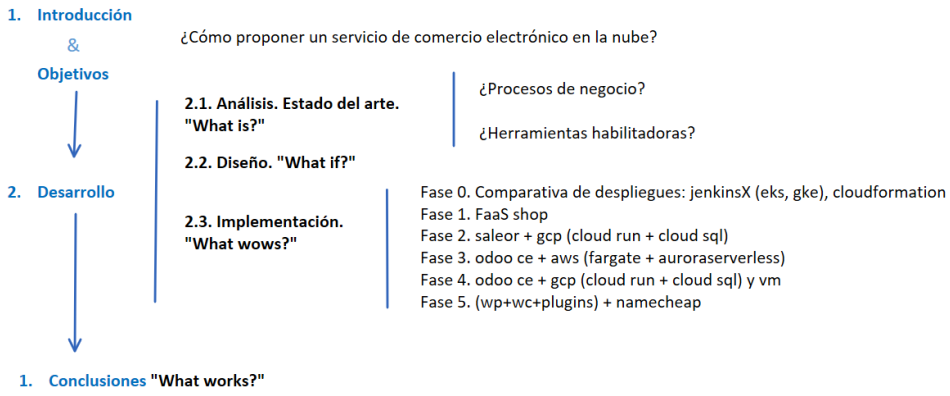


Ilustración 53. Esquema de las ideas del TFM

En este proyecto se han discutido estos tres objetivos de analizar, diseñar e implementar con relación a un servicio de comercio electrónico en la nube: primero se han identificado las necesidades del cliente y las herramientas en el mercado. Luego se ha presentado que lo hipotético es encontrar un Shopify por menos de 30€ con gestor de inventarios. Se han analizado más de **15 soluciones SaaS y más de 15 soluciones IaaS/PaaS, y se han desplegado 4 proyectos de código libre completos,** presentándose 3 soluciones en producción. Como se observa en la imagen siguiente, se ha elaborado scripts para levantar infraestructura de ci/cd en k8s y AWS, servicio para pymes como woocommerce o FaaS, proyectos de código libre punteros como saleor y odoo, y desplegado en microservicios. Además de proponer dos servicios web en producción actualmente breadfree.es y piedadleon.art.

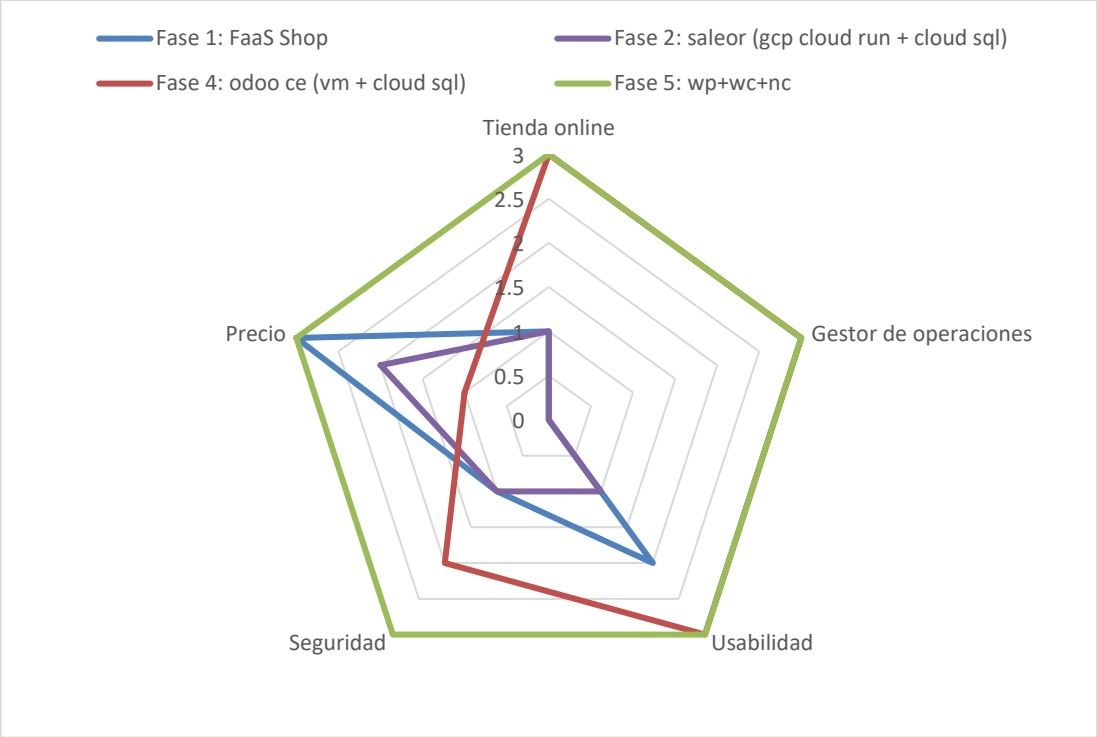


Ilustración 54. Marco de referencia sobre el que comprar las 3 soluciones en fase beta

b. Retos

Cómo conclusión estoy contento con lo que he aprendido, el enfoque que hemos seguido ha sido tanto aprender despliegue en la nube, avanzado en lo aprendido en las asignatura de Big data fundamentos y arquitectura o blockchain fundamentos y arquitecturas, en las que no se había indagado tanto. Además, resalto haber practicado técnicas de desarrollo con react aprendidas en blockchain desarrollo de aplicaciones Eso en cuanto a “hard skils” que he adquirido. En cuanto a “soft skills” destaco sobre todo esa capacidad de dialogar y entender a los clientes a lo largo de la duración de todo el proyecto, de forma que era capaz no sólo de proponer soluciones técnicas sino de poder preguntar y entender de qué manera les cuadraba la solución propuesta en ese momento.

6.2 LÍNEAS FUTURAS

En cuanto a líneas futuras quiero resaltar tres: por un lado el haber identificado como un servicio muy útil el poder ofrecer odoo ce desplegado en kubernetes, siempre y cuando se encuentren más de 5 usuarios a los que atender. Por otro lado, en cuando al servicio de woocomerce, si bien se ha planteado este hasta una fase beta de pruebas, faltaría consultar el precio de los materiales con los

proveedores y ejecutar una estrategia de puesta a producción. Y finalmente en cuanto al marco de referencia en general, faltaría desarrollarlo para que el servicio fuera más completo, por un lado, mejorar la usabilidad y sobre todo el seo y por otro hacer un análisis de seguridad integral, por ejemplo usando magerit y Pilar. La solución de saleor y odoo es muy estable pues se han planteados pipelines de integración y despliegue continuos. En cambio, si bien la solución de woocommerce namecheap es mucho más barata y posee la funcionalidad deseada, no se consigue proveer de una infraestructura tan estable.

7 REFERENCIAS

Se ha consultado la siguiente documentación:

[1] Shopify, «Shopify,» 2021. [En línea]. Available: https://www.shopify.com/.

[2] Santrel Media, «Shopify Tutorial for Beginners (Full Tutorial) - Create A Professional Online Store,» October 2021. [En línea]. Available: https://www.youtube.com/watch?v=ACqrKzY-j-s.

[3] Shopify, «Emailing order status updates,» 2021. [En línea]. Available: https://help.shopify.com/en/manual/orders/status-tracking/order-status-email.

[4] Shopify, «Tracking and adjusting inventory,» October 2021. [En línea]. Available: https://help.shopify.com/en/manual/products/inventory/track_inventory.

[5] Shopify, «Shipping rates,» October 2021. [En línea]. Available: https://help.shopify.com/en/manual/shipping/understanding-shipping/shipping-rates.

[6] Shopify, «Sets, Bundles, Composite Products,» October 2021. [En línea]. Available: https://community.shopify.com/c/shopify-apps/sets-bundles-composite-products/td-p/480579.

[7] Wix, «Wix,» October 2021. [En línea]. Available: https://www.wix.com/ecommerce/website.

[8] Wordpress.org, «Wordpress.org,» October 2021. [En línea]. Available: https://wordpress.org/.

[9] Woocommerce, «Woocommerce,» October 2021. [En línea]. Available: https://woocommerce.com/.

[10] Adobe, «Magento,» October 2021. [En línea]. Available: https://magento.com/.

[11] Salesforce, «Salesforce,» October 2021. [En línea]. Available: https://www.salesforce.com/eu/.

[12] SAP, «Software para gestión de pequeñas y medianas empresas,» October 2021. [En línea]. Available: https://www.sap.com/spain/products/sme-business-software/products.html.

[13] Etsy, «Etsy,» October 2021. [En línea]. Available: https://www.etsy.com/.

[14] C. Hive, «Etsy vs Shopify - Pros and Cons 2021 for Handmade Business,» October 2021. [En línea]. Available: https://www.youtube.com/watch?v=xfj7X-rA4_I&t=944s.

[15] A. Das, «It's FOSS,» October 2021. [En línea]. Available: https://itsfoss.com/open-source-ecommerce/.

[16] Saleor, «Saleor,» Octubre 2021. [En línea]. Available: https://saleor.io/.

[17] Amazon, «Amazon FBA,» Octubre 2021. [En línea]. Available: https://sell.amazon.com/pricing.

[18] Glovo, «Glovo,» Octubre 2021. [En línea]. Available: https://glovoapp.com/.

[19] Katana, «Katana,» Octubre 2021. [En línea]. Available: https://apps.shopify.com/katana-mrp-manufacturing-and-inventory-management.

[20] A. Blázquez, «1-State of the art.md,» 2021. [En línea]. Available: https://github.com/ablazleon/Analysis-design-and-implementation-of-efficient-business-software-tools/blob/main/1-State%20of%20the%20art.md.

[21] Ngork, «Ngork,» Octubre 2021. [En línea]. Available: https://ngrok.com/.

[22] F. Richter, «Amazon Leads $150-Billion Cloud Market,» 5 Julio 2021. [En línea]. Available: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/.

[23] E. Jones, «Cloud Market Share – a Look at the Cloud Ecosystem in 2021,» 8 September 2021. [En línea]. Available: https://kinsta.com/blog/cloud-market-share/.

[24] A. Blázquez, «Proyecto Jenkins X,» 2020. [En línea]. Available: https://github.com/ablazleon/env-k8-jenkins.

[25] A. Blázquez, «Aws cloudformation project,» 2020. [En línea]. Available: https://github.com/ablazleon/aws-cloudformation-deploy.

[26] Kubernetes, «Update Intro,» 2020. [En línea]. Available: https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/.

[27] AWS, «EKS,» 2020. [En línea]. Available: Https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html.

[28] AWS, «EKS workshop,» 2020. [En línea]. Available: Https://eksworkshop.com/prerequisites/k8stools.

[29] mdjnewman, «Script for running cloudformation yaml,» 2020. [En línea]. Available: Https://gist.github.com/mdjnewman/b9d722188f4f9c6bb277a37619665e77.

[30] A. Blázquez, «ArteLeon,» 2020. [En línea]. Available: https://github.com/ablazleon/ArteLeon.

[31] WinstonJs, «WinstonJs,» 2020. [En línea]. Available: https://github.com/winstonjs/winston.

[32] Stackify, «What is structured login and why developers need it,» 2020. [En línea]. Available: https://stackify.com/what-is-structured-logging-and-why -developers-need-it.

[33] Saleor, «SPectrum chat how to set a bg image,» 2020. [En línea]. Available: https://spectrum.chat/saleor/saleor-storefront/how-to-set-a-background-image-a-bg-color-and-home-screen-webapps-name~c7ba38ab-bd95-4c42-a325-9e4efa98fd8e.

[34] Saleor, «Spectrum chat añadir strings en otros idiomas,» 2020. [En línea]. Available: https://spectrum.chat/saleor/saleor-storefront/anyone-has-achieved-showing-the-storefront-in-another-language~7a738b20-226c-454e-a18c-d0b25c913dd4.

[35] A. Blázquez, «Procedimientos odoo ce (crun + csql),» [En línea]. Available: https://github.com/ablazleon/Analysis-design-and-implementation-of-efficient-business-software-tools/blob/main/Procedimientos.md.

[36] A. Blázquez, «Stack Overflow. Procedimiento de backup y http2 en odoo,» [En línea]. Available: https://stackoverflow.com/questions/68130278/how-to-set-in-a-dockerfile-an-nginx-in-front-of-the-same-container-google-cloud.

[37] Online media masters, «8 Tips To Reduce JavaScript Execution Time In WordPress,» 2021. [En línea]. Available: https://onlinemediamasters.com/reduce-javascript-execution-time-wordpress/.

[38] Onely, «How to Reduce WordPress Load Time by 4.6 Seconds in an Hour,» 2021. [En línea]. Available: https://www.onely.com/blog/how-to-reduce-wordpress-load-time/.

[39] Saleor, «Saleor spectrum habilitar comrpesión de texto,» 2020. [En línea]. Available: https://stackoverflow.com/questions/57504666/enable-text-compression-using-react-webpack-and-apache.

8 ANEXOS

8.1 GESTIÓN DE TIEMPO

Se calcula el tiempo dedicado en cada fase mediante toogl y pomodoro como cronómetro. Se dedicó 400 horas a las primeras fases, desde junio hasta febrero, a partir de entonces se empezó a cronometrar el tiempo.

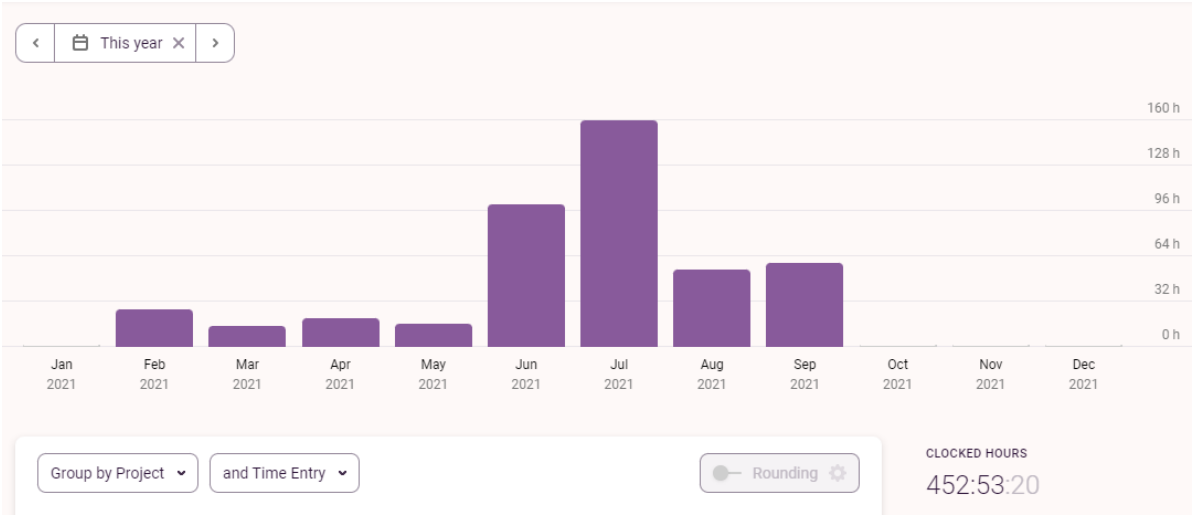


Ilustración 55. Diagrama de toogl reflejando las horas cronometradas sólo desde febrero

8.2 IMPACTO DE ESTE PROYECTO

En este apéndice se reflexiona de manera cualitativa sobre el posible impacto que este proyecto pueda causar, en cuanto a que este proyecto trata de analizar, diseñar e implementar un servicio de comercio electrónico en la nube. Se considerará el impacto social, el económico, el ambiental y las implicaciones éticas.

8.2.1 Impacto social

El impacto social se define como el cambio en las actividades humanas. Se identifican tres grandes cambios en las actividades humanas:

- primero que este proyecto causa un cierto impacto social en cuanto a que permite automatizar ciertas tareas empresariales que antes sólo podrían ser automatizadas si la empresa cuenta de gran inversión (+100€/mes). Por ello, si ciertos clientes pueden digitalizar sus procesos de negocio, pueden ser más competitivos.
- Por un lado, este hecho, por otro lado, se destaca una consecuencia de que las pymes sean más competitivas, que el país sea más próspero. Se reflexiona en ciertos documentos de la Unión Europea que está relacionada la prosperidad de un país con el número de pequeñas empresas de este. [40] Por ejemplo, se plantea que reforzando la digitalización de las pequeñas empresas se puede reforzar a que compitan con las grandes empresas o plataformas; puesto que en general las plataformas como Amazon pueden ofrecer productos más baratos y más accesibles en información que las pequeñas empresas, los comercios minoristas por su localización cercana pueden competir mejor con las plataformas si disponen de un canal de venta online.
- En último lugar, se plantea que, si bien el acceso a más empresas de procesos de negocio digitalizados las hace más competitiva se aumenta el tratamiento automatizado de la información lo que obliga al ciudadano a ser responsable de su derecho a la privacidad a través de la aplicación del Reglamento General de Protección de Datos.

8.2.2 Impacto económico

En relación con el impacto económico, es decir a lo rentable del servicio, nótese que se han implementado varios artefactos buscando en que cada uno, que sea rentable.

El artefacto de la fase 1, en cuanto a diseño es el más rentable por basarse en serverless y sólo gastar cuando se usa un servicio. Sin embargo, en la práctica, no es rentable, puesto que implica desarrollar y asegurar la calidad de muchos servicios que no han sido implementados todavía en ningún proyecto de código libre.

Los artefactos de la fase 2, 3 y 4, son funcionales, pero sólo rentables para empresas que puedan invertir en su canal online más de 40€/mes. Sino, cómo se ha desarrollado en cada artefacto, el servicio se degrada enormemente si no se eligen servicios en la nube de suficiente calidad.

El último artefacto destaca por ser a priori el más rentable. Así, si cuesta 4€/mes se puede plantear ofrecer al precio de wix de 20€/mes ganado 15€/mes por cada cliente

8.2.3 Impacto ambiental

En relación con el impacto al ecosistema, se plantea que este proyecto ofrece la posibilidad a las pequeñas tiendas de poder automatizar tareas, como el servicio de catálogo o la gestión de inventarios. Se destaca a continuación, que esto puede contribuir a varios cambios:

- a reducir el uso del transporte para adquirir bienes, y relegarlo a un servicio mayorista de transporte de paquetería. Si el cliente no está obligado a desplazarse a la localización de u artista para adquirir sus obras, si su transporte impacta más al medio ambiente que un servicio de transporte mayorista, el impacto al ecosistema puede ser menor.
- O, si los trámites empresariales se pueden realizar de forma online, se favorece que los trabajadores operen de forma remota y que no estén obligados a desplazarse, lo cuál puede reducir el impacto al ecosistema.
- En los dos cambios anteriores se ha reflexionado cómo poder ofrecer un canal online al cliente puede reducir el impacto al ecosistema por un uso más eficiente del transporte. Pero otra posible consecuencia puede ser el cambio de usar gran cantidad de papel a usar servicios informáticos para los trámites. Los servicios informáticos consumen electricidad; si esta se obtiene de forma sostenible para alimentar los centros de datos el impacto al exosistema puede ser menor. Por otro lado, reducir el uso de grandes cantidades de papel, puede contribuir a necesitar menos hectáreas para plantaciones de eucalipto que acidifican la tierra.

8.2.4 Implicaciones éticas

Finalmente, en relación con las implicaciones éticas, es decir, a cómo se afecta la naturaleza de las acciones, se prevé proporcionar un servicio barato de ahorrar tiempo a colectivos que no podrían favorecerse de estos servicios, posiblemente ayudando a que la sociedad se desarrolle de forma mejor. No sólo, por la reflexión realizada en el apartado anterior de que un país con pequeñas empresas es un país más próspero, sino que un país en el que un ciudadano puede desarrollarse más libremente porque puede acceder a herramientas que le benefician porque son baratas, se plantea que sea un país mejor. Es cierto, que esto implica que nos concienciemos para destacar el papel de la formación en nuevas tecnologías para hacer más competentes a los trabajadores, puesto que si ciertos ciudadanos, por ejemplo los de mayor edad, no desea formarse, puede producirse el fenómeno de la "brecha digital".

8.3 PRESUPUESTO ECONÓMICO

Este apéndice describe el presupuesto que se plantea de este proyecto:

El precio por hora de un ingeniero se obtiene del COIT 2017. (COIT). De ahí se Encuentra el coste por hora, basándose en que una persona-mes consiste en 8 horas 22 días al mes.

Media anual	Persona mes = Media anual /11	Horas al mes	Precio por hora
53,000.00 €	4,818.18 €	176	27.38 €

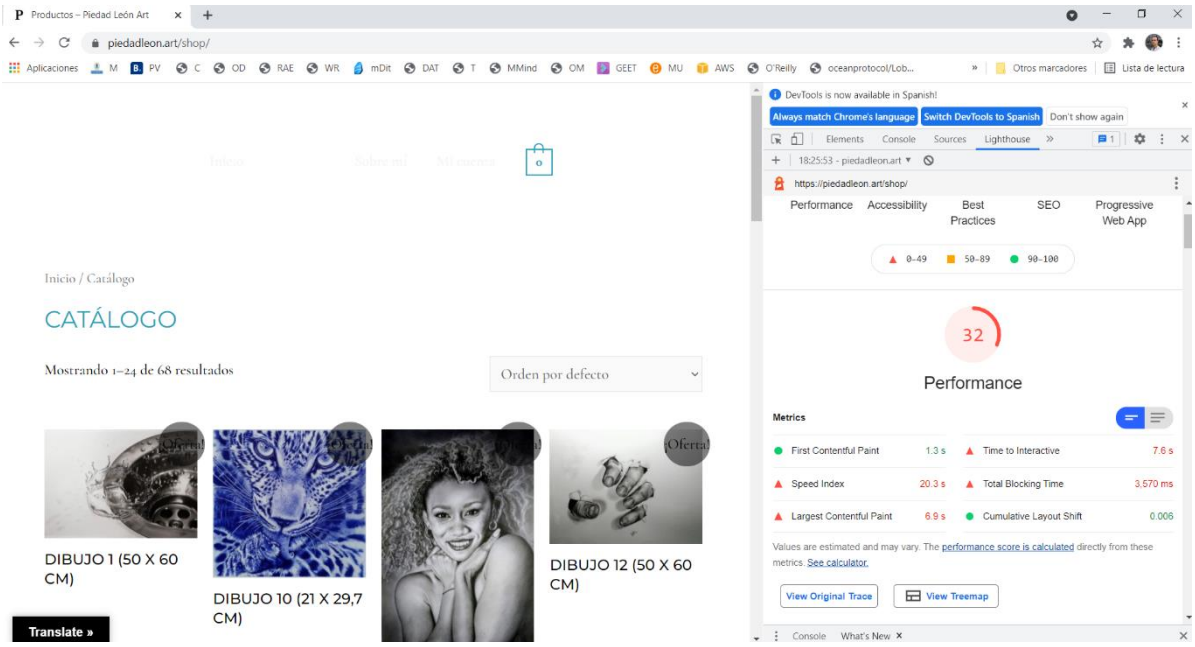
Se propone un beneficio industrial solo del 5%, ya que las horas de este año han servido más a un fin investigador que a uno comercial.

Costes directos				
Desarrollo		**Hours**	**Price/hour**	**Total**
		850	€ 27	€ 22.950
Inversión en medios materiales	**Inversión inicial**	**Use in months**	**Amortización(años)**	**Total**
Ordenador (Software incluido)	€ 500.00	15	5	€ 75,00
Other				
Costes indirectos: general expenses			25%	€ 5756,75
Beneficio industrial (sobre costes indirectos + directo)			5%	€ 1435
Budget Subtotal				€ 30.216
IVA			21%	€ 6345
Total				**€ 36.561**

8.4 LISTA DE ILUSTRACIONES

8.5 MEJORA DE LA CARGA DEL ARTEFACTO 5

A continuación, se desarrolla el reto de aumentar la usabilidad de este artefacto final aplicando lighthouse para mejorar la performance. Para ello se parte de una performance en torno a 25, y se aplican las ideas de estos tutoriales ([37]y [38]) , tratar de comprimir el tamaño de las imágenes, lo que como se observa en la imagen no es posible, hacer que elementor cargue los activos de otra forma y reduciendo el número de productos del catálogo a 4. Además, se incluye una imagen final en relación al hecho de que sitios web competidores como postery no evitan reducir la calidad de las imágenes que enseñan en la web, con el fin de evitar que un actor malicioso se descargase esta y la imprimiese como póster.



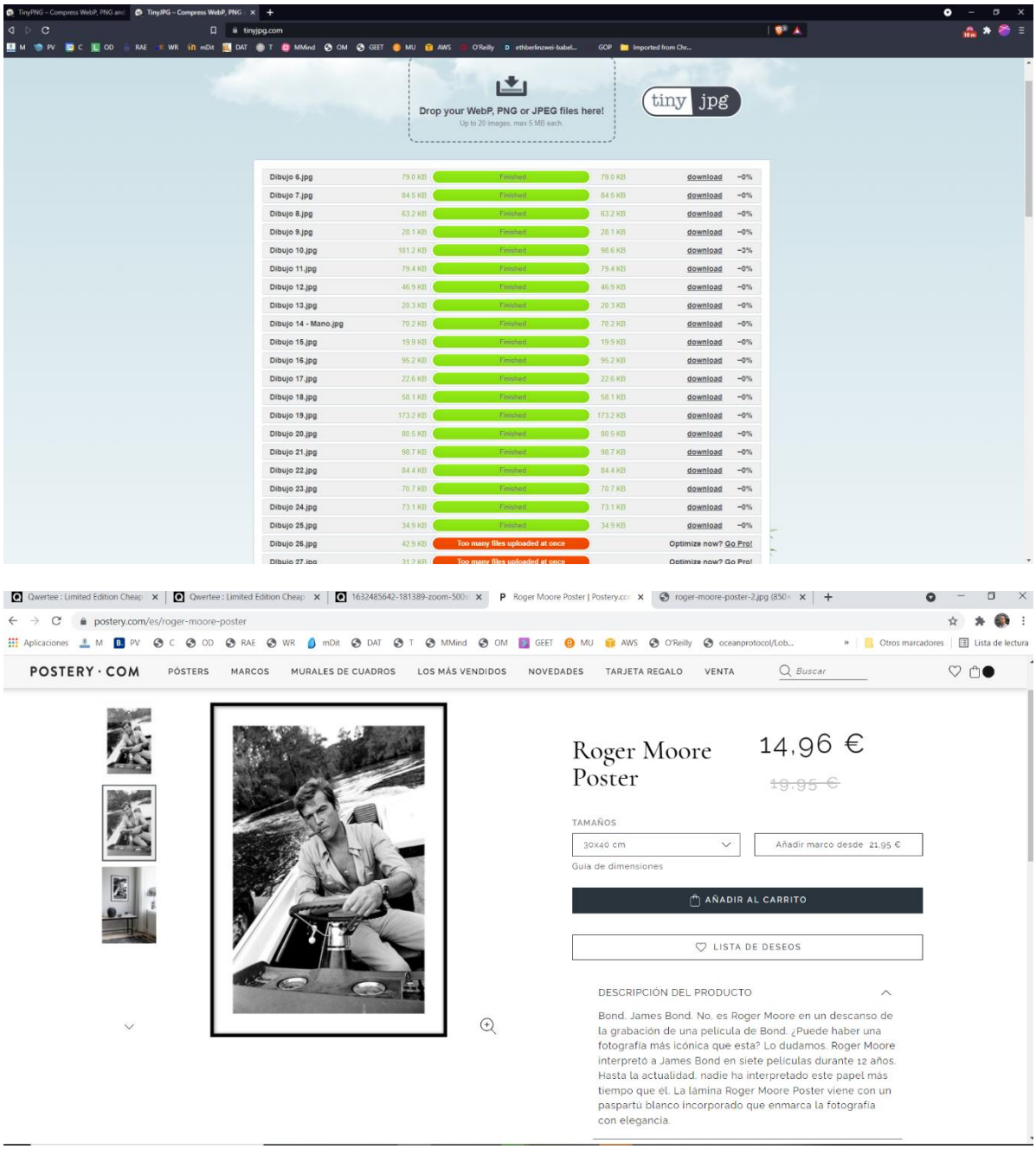


Ilustración 56. Primero ligthouse con turbo y elemento asset activado pero con 24 productos en cada página. En la segunda imagen de tinypng diciendo que no puede comprimir más las imágenes y por último imagen de postery donde se observa que las imágenes tienen calidad