2022

DESARROLLO DE UNA PLATAFORMA WEB DE E-COMMERCE CONFIABLE Y ESCALABLE ORIENTADA AL SECTOR DE LAS MIPYMES

ABARCAS VIDAL, EDGARD LEONARDO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE

"DESARROLLO DE UNA PLATAFORMA WEB DE E-COMMERCE CONFIABLE Y ESCALABLE ORIENTADA AL SECTOR DE LAS MIPYMES"

EDGARD LEONARDO ABARCAS VIDAL

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUIA: **NICOLÁS JARA CARVALLO**

PROFESOR CORREFERENTE: **FRANCISCO CABEZAS BERRÍOS**

Agradecimientos

A mis papás, que sin ellos no estaría acá, a quienes les debo la vida y más. A mi hermano que me apoya, me ayuda, de quien aprendo y estoy orgulloso. Y todas las personas, partiendo por todos los profesores del colegio y la Universidad que han sido parte de mi formación como profesional y persona, destacando al profe Barros, mi profe de matemáticas de la básica, quien me hablara por primera vez de esta carrera, y con quienes la estoy terminando ahora, el profesor Nicolás y el profesor Francisco que me han acompañado en esta última etapa. Y bueno, tantas personas que han sido parte, compañeros, amigos, que me han apoyado, ayudado, acompañado, alentado, consolado, con quienes he reído, disfrutado, estudiado, trasnochado, sufrido y pasado muchos momentos. De una u otra forma a todos ellos le debo parte de lo que soy y donde estoy, gracias a todos por acompañarme en este viaje. Digo viaje porque tengo varias concepciones y metáforas de la vida, una de ellas es la de un tren, y las etapas de la vida son algo así como las estaciones, las personas te acompañan en dicho tren. A veces crees que estás solo, pero no, es solo que algunas veces no ves a los demás por estar en distintos vagones. Así también hay algunos que se bajan de tu tren, dejan de ser parte de tu vida, por distintas razones, ya sea que no vamos al mismo lugar, los caminos se separan o van a ocuparse de su propio viaje, pero por muy poquito que hayan estado no dejan de ser importantes y siempre nos dejan algo, algún recuerdo, alguna enseñanza. También agradecer o dedicar esto a quienes ya terminaron su viaje, a quienes ya no están y con quienes me hubiese gustado compartir este momento, pero sé que de una u otra forma siempre están y estarían orgullosos de ver en lo que me he convertido.

Resumen

La mayoría de las empresas del sector MiPymes tienen un importante déficit en lo que respecta a comercio electrónico. InforedChile, uno de los mejores directorios web de MiPymes y servicios públicos del país, busca ayudar a estas empresas y generar también, una oportunidad de negocio.

El objetivo principal de este proyecto es crear una plataforma e-Commerce junto a la página de InforedChile que permita la comunicación entre las MiPymes y sus potenciales compradores de manera segura. Con este fin, surge la pregunta: ¿De qué manera se puede implementar una plataforma e-Commerce que permita una comunicación confiable entre las empresas MiPymes y sus compradores?

Para responder esta pregunta, primero se estudiará las soluciones existentes y las alternativas para el desarrollo e implementación del sitio. Luego se establece el diseño, se genera una propuesta de solución y finalmente se establecen los objetivos, requisitos, arquitectura y componentes que tendrá el sistema.

Los resultados muestran que tomar un software open-source que pueda adoptar múltiples tiendas, utilizarlo como base para implementar el marketplace y complementarlo con el uso de herramientas externas de CloudComputing es mejor opción que generar una solución desde cero.

Abstract

Most of the companies in the MSMEs sector have a significant deficit when it comes to electronic commerce. InforedChile, one of the best MSMEs web directories and public services in the country, seeks to help these companies and also generate a business opportunity.

The main objective of this project is to create an e-Commerce platform along with the InforedChile website that allows safe communication between MSMEs and their potential customers. To this end, the question arises: How can an e-Commerce platform be implemented that allows reliable communication between MSMEs and their buyers?

To answer this question, the existing solutions, the alternatives for the development and the site implementation will be studied. Then the design is set, a solution proposal is generated and finally the objectives, requirements, architecture and components that the system will have are established.

The results show that taking open-source software that can be adopted by multiple stores, using it as a base to implement the marketplace and complementing it with the use of external Cloud Computing tools is a better option than building a solution from scratch.

Glosario

- **MiPyme:** Micro pequeña y Mediana Empresa
- **Indexar:** Incluir en el índice de internet o sitio web el contenido de otro.
- **e-commerce:** Comercio electrónico
- **Comisión:** La comisión es la cantidad que se cobra por realizar transacciones comerciales que corresponden a un porcentaje sobre el importe de la operación.
- **Open-source:** Software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de código abierto o forman parte del dominio público.
- **API:** Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones (Application Programming Interface)
- **Ruby:** Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos
- **Plugins:** Un plug-in o complemento es una aplicación que se relaciona con otra para agregarle una función nueva y generalmente muy específica.
- **Cloud computing:** El cloud computing o la computación en la nube, es una tecnología que permite acceso remoto a softwares, almacenamiento de archivos y procesamiento de datos por medio de Internet, siendo así, una alternativa a la ejecución en una computadora personal o servidor local.
- **Hosting web:** Un hosting es un servicio de alojamiento para sitios web, el cual aloja los contenidos de tu web y tu correo electrónico para que puedan ser visitados en todo momento desde cualquier dispositivo conectado a Internet.
- **TI:** Tecnología de la Información

- **Migración:** Proceso que necesitamos hacer para transferir los datos de un sistema a otro mientras cambiamos el sistema de almacenamiento donde se encuentran los datos, o bien mientras se practican las modificaciones necesarias en la base de datos o la aplicación que los gestiona.

- **Stakeholder:** Un stakeholder es el público de interés para una empresa que permite su completo funcionamiento. Se refiere a todas las personas u organizaciones que se relacionan con las actividades y decisiones de una empresa como: empleados, proveedores, clientes, gobierno, entre otros.

- **Script:** Secuencia de comandos o guión. Es un término informal que se usa para designar a un programa relativamente simple.

- **Privilegio:** Derecho de un usuario a realizar una tarea específica, que suele afectar a un sistema completo en lugar de un objeto determinado. Los privilegios los asignan los administradores a usuarios individuales o grupos de usuarios, como parte de la configuración de seguridad del equipo.

- **Frontend:** Frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

- **MA:** Marketred API

- **FE:** FrontEnd

- **IP:** 1. Identity Provider. 2. Número que identifica de forma individual la conexión de un equipo o dispositivo a una red interna o externa.

- **Intuitivo:** Se utiliza para describir interfaces de programas informáticos, haciendo referencia a que son de fácil entendimiento para un usuario que los vea por primera vez.

- **Levantar:** Habilitar para uso, publicar
- **Storefront:** Sitio principal en donde se presentan las tiendas, los objetos y servicios que son vendidos
- **Backend:** El backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web.
- **VPC:** Amazon Virtual Private Cloud (VPC) es un servicio que le permite lanzar recursos de AWS en una red virtual aislada lógicamente que usted defina.
- **Dominio:** Un nombre de dominio (a menudo denominado simplemente dominio) es un nombre fácil de recordar asociado a una dirección IP física de Internet.
- **Clúster:** El término clúster se aplica a los sistemas distribuidos de granjas de computadoras unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen un único servidor.
- **ECS:** Amazon Elastic Container Service (ECS) es un servicio de administración de contenedores altamente escalable y de alto rendimiento que admite contenedores Docker y le permite ejecutar aplicaciones fácilmente en un clúster administrado de instancias de Amazon Elastic Compute Cloud (Amazon EC2).
- **Peering:** El peering (conexión directa, intercambio de tráfico o emparejamiento) es la interconexión voluntaria de redes de Internet administrativamente independientes con el fin de intercambiar tráfico entre los usuarios de cada red.
- **CIDR:** (Classless Inter-Domain Routing), Enrutamiento entre Dominio sin Clase. Para evitar que los enrutadores de red se sobrecarguen con rutas, se utiliza una técnica llamada CIDR. Son prefijos de red con longitud variable. Con este método se consiguen unir varios prefijos grandes, que sean consecutivos, en uno más pequeño, reduciendo el número de entradas en las tablas de los routers.

- **RDS:** Amazon Relational Database Service (Amazon RDS) es un servicio administrado que facilita las tareas de configuración, operación y escalado de una base de datos relacional en la nube.
- **Instancia:** Una instancia de un programa es una copia de una versión ejecutable del programa que ha sido escrito en la memoria del computador.
- **Inbound:** Entrante, de entrada
- **Repositorio:** Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software.
- **ECR:** Amazon ECR es un registro de contenedores completamente administrado que ofrece alojamiento de alto rendimiento, por lo que puede implementar imágenes y artefactos de aplicaciones de manera confiable en cualquier lugar.
- **Imagen:** archivo informático donde se almacena una copia o imagen exacta de un sistema de archivos o software.
- **Build:** Versión operativa de un producto software que incorpora un subconjunto de las funciones que se incluirán en el producto final. Generalmente se identifica por un número de compilación, en lugar de por un número de versión.
- **Push:** Introducir, insertar, cargar. Operación sobre una pila consistente en añadir un elemento a la misma.
- **Balanceador de carga:** un balanceador de carga es una herramienta, la cual direcciona a un cliente al servidor web que se encuentre con mayor disponibilidad entre los que cuentan con el mismo contenido.
- **Enrutar:** Determinar el itinerario que debe seguir un paquete de datos dentro de una red de comunicación para llegar a su destino.

- **Certificado SSL:** Un certificado SSL es un certificado digital que autentica la identidad de un sitio web y habilita una conexión cifrada. La sigla SSL significa Secure Sockets Layer (Capa de sockets seguros), un protocolo de seguridad que crea un enlace cifrado entre un servidor web y un navegador web.

- **ACM:** AWS Certificate Manager es un servicio que le permite aprovisionar, administrar e implementar con facilidad certificados de capa de conexión segura/seguridad de la capa de transporte (SSL/TLS) públicos y privados para su uso con servicios de AWS y recursos internos conectados.

- **URI:** Un identificador de recursos uniforme o URI (Uniform Resource Identifier) es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

- **EC2:** Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable.

- **CloudWatch:** Amazon CloudWatch es un servicio de monitorización y observación creado para ingenieros de DevOps, desarrolladores, ingenieros de fiabilidad de sitio (SRE) y administradores de TI. CloudWatch ofrece datos e información procesable para monitorizar sus aplicaciones, responder a cambios de rendimiento que afectan a todo el sistema, optimizar el uso de recursos y lograr una vista unificada del estado de las operaciones.

- **DevOps:** DevOps es un conjunto de prácticas que agrupan el desarrollo de software y las operaciones de TI.

- **URL:** URL significa Uniform Resource Locator y es la dirección única y específica que se asigna a cada uno de los recursos disponibles de la World Wide Web para que puedan ser localizados por el navegador y visitados por los usuarios.

Índice general

1. INTRODUCCIÓN

¿Sabían que en 2020 las ventas por internet se han triplicado en Chile en comparación al año anterior? [1] El Covid-19 marcó un “antes” y un “después” en la vida de las personas. Los confinamientos obligatorios, el uso de mascarillas y el distanciamiento social representaron un problema también para los comercios, los cuales han tenido que adaptarse y reinventarse

Es por esto que el e-commerce o comercio electrónico, ha tomado cada vez mayor relevancia en la compra de productos y servicios en el último tiempo, de hecho, su crecimiento desde su creación ha sido de forma exponencial.

Frente a esto, el segmento de MiPYMES se encuentra en una gran desventaja ante las grandes empresas, que en su mayoría, cuentan con plataformas de e-commerce establecidas. Por otro lado, en el caso de las MiPYMES, menos de un tercio realiza comercio electrónico [2] o sus canales e-commerce no ofrecen la confianza necesaria a los consumidores para realizar compras, por lo que poseen un importante déficit en ventas y experiencia en este ámbito.

Por lo que nace la necesidad de conexión con este canal de ventas por parte de este segmento de empresas. Entonces “¿Cómo podemos ayudar a las empresas del segmento MiPymes en cuanto a comercio electrónico y visualización en internet?“

InforedChile, uno de los mejores directorios web de MiPymes y servicios públicos del país, busca ayudar a estas empresas y generar también, una oportunidad de negocio. El objetivo principal del proyecto es crear una plataforma e-Commerce en forma paralela a la página de InforedChile, la cual permita la comunicación entre las MiPymes con sus potenciales compradores, además de la generación de transacciones de compra desde el catálogo de venta ofrecido por cada empresa. Con este fin, surge la pregunta: ¿De qué manera se puede implementar una plataforma e-Commerce que permita una comunicación confiable entre las empresas MiPymes y sus compradores?

En este informe se documentan los pasos necesarios para dar respuesta a esta pregunta, primero se estudiaron las soluciones existentes y las alternativas para el desarrollo e implementación del sitio. Luego se dio paso al diseño, se generó una propuesta de solución, se establecieron los objetivos, requisitos, arquitectura y componentes que tendrá el sistema.

2. ESTADO DEL ARTE

2.1. Comercio electrónico

En 1960 surge la iniciativa de Electronic Data Interchange (EDI), que permitió compartir información entre empresas de manera electrónica, sentando las bases de lo que sería el comercio electrónico e incluso Internet. Luego en 1979, se crea el teleshopping de ventas por catalogo donde la televisión se convirtió en una fuente de compraventas a distancia. [3]

En 1981 se llevó a cabo la primera transacción comercial usando Internet, cuando Thompson Holidays conectó a sus agentes de viajes para que pudieran tener acceso a los productos disponibles haciendo uso de Internet en tiempo real. Una década después, en 1991 la National Science Fundation (NSF) aprobó el uso de Internet con usos comerciales.

Un año después, en 1992, surge el primer proyecto de tienda online propiamente tal, era un sistema que imitaba a los tradicionales tablones de anuncios y que permitía que los usuarios vendieran y compraran libros. Más adelante, esta página evolucionó hasta convertirse en `Books.com`. En 1994, Netscape implementó SSL, lo que permitió el envío de datos personales a través de Internet de manera segura, lo que también supuso un paso importante en la evolución del comercio electrónico a nivel mundial.

Otro hecho importante en la evolución del comercio electrónico se debió a la irrupción de la tecnología celular. De hecho, en 1997, Coca Cola fue la primera empresa que implantó un sistema que permitía realizar compras de sus productos a través del teléfono celular. Un año más tarde, en 1998, se fundó PayPal, lo que supuso un nuevo impulso al comercio electrónico gracias a las facilidades y seguridad en el pago que ofrecía. En 2006 Google haría lo propio y lanzaría Google Checkout.[4]

Desde el 2010 en adelante las redes sociales pasan a liderar la web y se da la masificacion de tiendas online. Muestra de ello es que en 2014 Amazon se alía con Twitter, dando como resultado la opción de que sus usuarios realicen compras online directamente mediante tweets. Ese mismo año, los beneficios producidos por las ventas online en todo el mundo alcanzan los 1.500 millones de dólares, llegando a suponer un incremento de más del 20 % respecto al año anterior. A partir de ese momento, y con la irrupción de la tecnología smartphone, las ventas del comercio electrónico no han hecho nada más que subir, creando un mercado mucho más dinámico y eficiente, así como un mercado que alcanza a millones de usuarios conectados en todo el mundo. [3] [4]

2.2. Tipos de comercio electrónico

Se suelen reconocer cuatro grandes tipos de comercio: B2B, B2C, C2B y C2C, pero los avances tecnológicos han ayudado a que las ventas por internet lleguen cada vez a más industrias. Esta expansión ha traído como resultado el surgimiento de nuevos tipos de e-commerce:

2.2.1. Business to Business (B2B)

Business to Business (negocio a negocio) se refiere a una transacción comercial entre dos empresas que tienen presencia y operaciones en internet. En este tipo de negocio no intervienen los consumidores finales.

Para poder participar en este tipo de comercio electrónico, es recomendable tener experiencia previa en relaciones comerciales con otras marcas. También se sugiere contar una logística lo suficientemente robusta, como para poder satisfacer las necesidades de los clientes, o sea, las otras empresas.

Ejemplo: Una empresa que hace compras al por mayor a otra empresa.

2.2.2. Business to Consumer (B2C)

Business to Consumer (negocio a consumidor) fomenta las relaciones comerciales entre negocios de cualquier magnitud y consumidores. Las tiendas online y los marketplaces entran en esta categoría.

Este tipo de comercio está orientado a satisfacer las necesidades del cliente final, por lo que el servicio de atención debe ser una prioridad para las empresas que forman parte de esta clasificación.

Ejemplo: *Los marketplaces como Mercado Libre, Amazon o Linio.*

2.2.3. Consumer to Business (C2B)

Consumer to Business (consumidor a negocio) se refiere a aquellas transacciones en las que consumidores o profesionales que trabajan de forma independiente ofrecen sus servicios o productos a las empresas.

Este tipo de comercio electrónico se ha vuelto cada vez más común, gracias al uso de medios digitales como blogs, redes sociales, videos y podcasts.

Ejemplo: *Un influencer que recibe una comisión por cada persona que visita un enlace publicado en una de sus redes sociales.*

2.2.4. Consumer to Consumer (C2C)

El último que podríamos considerar entre los tipos de e-commerce más tradicionales o conocidos, es el comercio electrónico Consumer to Consumer (consumidor a consumidor).

Como su nombre lo dice, el comercio C2C se presenta cuando el intercambio se lleva a cabo entre consumidores y no hay ningún tipo de empresa de por medio. Se trata de una transacción más informal y, regularmente, a precios más bajos que en los otros tipos de comercio.

Ejemplo: *Una persona que anuncia su auto en el Marketplace de Facebook o en plataformas de ventas de productos usados.*

2.2.5. Government to Consumer (G2C)

Government to Consumer (gobierno a consumidor) es la clasificación del comercio electrónico en la que se realizan transacciones monetarias digitales entre gobiernos o administraciones y consumidores, a través de portales oficiales.

En cuanto a las ventajas de este tipo de comercio —además de la seguridad y la rapidez para efectuar trámites— es que el consumidor tiene acceso a las plataformas de pago a cualquier hora y desde cualquier lugar.

Ejemplo: *Trámites digitales como el pago de la licencia de conducir o el pago de multas vehiculares.*

2.2.6. Business to Government (B2G)

Business to Government (negocio a gobierno), también conocida como Business to Administration (B2A), es aquella en la que una entidad gubernamental compra bienes o servicios digitales a empresas a través del comercio electrónico.

Ejemplo: *Cuando el gobierno contrata a una empresa para que desarrolle una página web.*

2.2.7. Business to Employee (B2E)

Business to Employee (negocio a empleado) se refiere a las compras que las empresas realizan para sus trabajadores.

Esta es una estrategia bastante común en el mundo laboral, sobre todo en mercados competitivos en los que hay mucha rotación de personal.

Ejemplo: *Cuando una empresa paga la licencia de un software para que sus empleados puedan usarlo.*

2.3. E-commerce en Chile

Ya hemos hablado de la evolución del comercio electrónico en términos generales, pero ¿qué pasa específicamente con Chile?. A mediados de los ochenta, las principales universidades de Chile incursionan en proyectos llevados adelante en sus centros de informática y computación a través de la plataforma UUCT, que conectaba a las universidades y junto con ello se logró el envío del primer correo electrónico en el país

Estos proyectos universitarios sirvieron de preparación para que el país entrara a la era digital. El avance tecnológico no fue nada fácil, el impulso decisivo ocurrió al crearse el primer sitio web en Chile

Ya para el año 2017, Chile encabeza la lista de acceso a internet en Latinoamérica y en 2019 ya se dispone de un entorno de e-commerce sólido de ventas online en el país, registrando para ese año un liderazgo en América Latina en las ventas per cápita: USD 320 en Chile, USD 192 en Mexico, USD 155 en Colombla, USD 145 en Argentina y USD 121 en Perú, de acuerdo a cifras manejadas por eMarketer

Sin embargo, la cultura de comercio electrónico en el país seguía siendo escasa, esto, dado que en Chile el comercio digital no era prioridad. [3]

Pero este modo de compra resultó ser bastante cómodo. Los empresarios descubrieron que permitía un ahorro de tiempo y, en ocasiones, de dinero. Además, la restricción de horarios no aplica en este modelo de ventas y es muy probable que, aunque pase la crisis por la pandemia, las compras en línea mantengan su ritmo de crecimiento. Pero la verdad es que ya tenía un crecimiento notable antes de esta situación, así como podemos apreciar en el gráfico 2.1

Fig. 2.1: Ventas B2C por ecommerce en Chile 2000 al 2021. [5]

Las ventas por internet en 2019 alcanzaron USD 6.079 millones, lo que representa un crecimiento del 20 % respecto de 2018, que alcanzó los USD 5.200 millones.

Durante el 2020 las ventas por internet crecieron un 55 % respecto del 2019 y las ventas fueron de USD 9.400 millones, cambio acelerado por el Covid-19 y las restricciones de compra físicas producto del confinamiento. Se proyecta que para este 2021, las ventas por Internet hayan llegado a USD 12.000 millones, con un crecimiento estimado respecto del 2020 de 25 %. [5]

2.4. Soluciones Existentes

2.4.1. Mercantil

En primera instancia, dentro de las soluciones existentes encontramos Mercantil, esta página se presenta como el Directorio de Empresas de Chile, información detallada con más de 200.000 empresas y más de 5.500 actividades de negocios, además de tener portales B2B y B2C asociados, cuentan con una serie de alternativas para que la empresa figure en internet, como catálogos virtuales, advertising, aumentando así las posibilidades de contactar nuevos clientes.

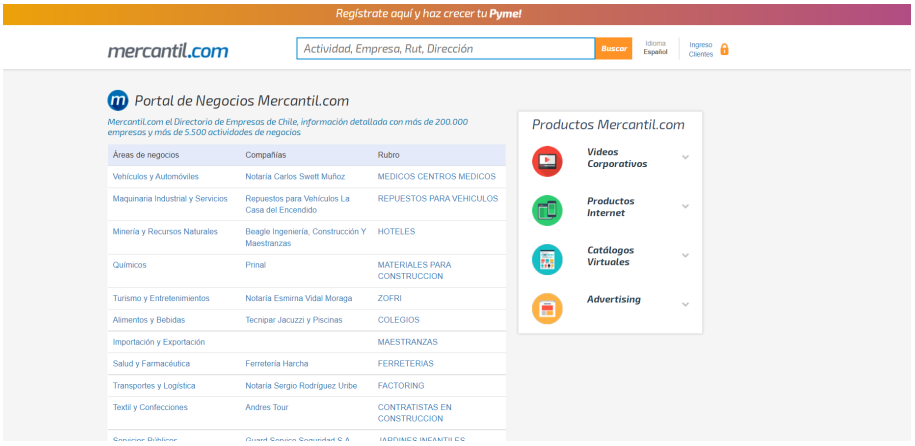


Fig. 2.2: Mercantil

Lo que ofrece Mercantil es bastante similar a lo que ofrece InfoRed en estos momentos, cuál sería el valor agregado que tendría ahora el sitio web de InfoRed que lo haría diferenciarse de esta plataforma, es que permitiría que las empresas realizaran directamente en su sitio web comercio electrónico al incorporar el marketplace a sus servicios, permitiendo además una oportunidad de negocio con las comisiones por venta que se realicen en su sitio web.

A raíz del boom que ha tenido este último tiempo el comercio electrónico por el Covid, las cuarentenas, además del estallido social, muchas tiendas grandes de Retail han puesto a disposición sus plataformas de ventas online para que otras empresas, no necesariamente Pymes, ofrezcan sus productos a modo de marketplace. Si bien este proyecto está enfocado los marketplaces, por lo que las opciones establecidas actualmente por Falabella, Linio, Ripley, Paris, entre otros, estarían dentro de dicha categoría, en este apartado queremos centrarnos en aquellas opciones de marketplace más especializadas en lo que son las MiPymes y que no estén ligadas a las empresas grandes de retail porque no serían éstos los principales competidores del proyecto que busca establecer InfoRedChile.

2.4.2. MercadoLibre

Si bien esta plataforma no está pensada específicamente para PYMEs, cualquiera puede ofrecer sus productos en ella. Ofrecen un servicio gratuito, pero tiene un límite de ventas anuales, además de un límite de tiempo durante el cual una publicación se encuentra disponible. Para eliminar estas restricciones, se debe utilizar algunas de las opciones pagadas, que cobran un gran porcentaje de comisión por cada venta[6].

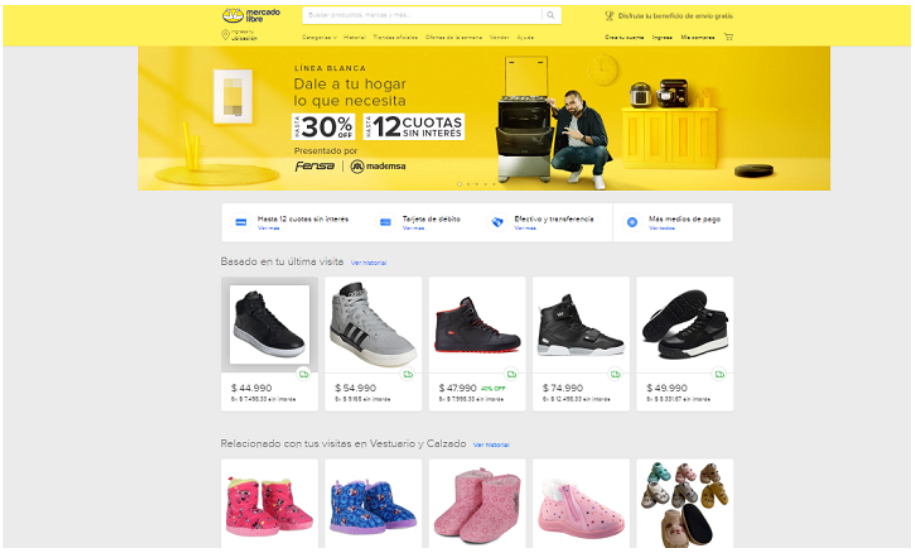


Fig. 2.3: Mercado Libre

2.4.3. *Apaño Tu Pyme*

Como el nombre le dice, esta plataforma si está pensada en PYMEs, y se requiere tener un RUT de empresa para poder ofrecer productos. Si bien dicen no cobrar comisiones, hay comisiones por cada venta dependiendo del medio de pago utilizado[7].

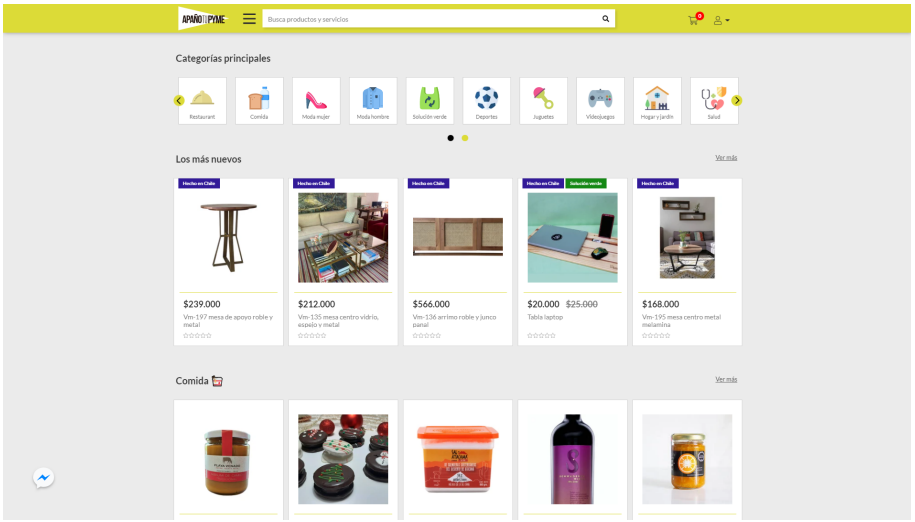


Fig. 2.4: Apaño Tu Pyme

Ninguno de estos sitios tiene a su disposición algo similar a la base de datos de InforedChile, ni integración a alguna plataforma similar a su directorio.

2.5. Alternativas para el Desarrollo

Existe un número importante de softwares open-source que ofrecen la creación sitios de comercio electrónico, las que podrían ser usadas como base para resolver este desafío. Los más importantes, categorizados por tecnología, son:

2.5.1. PHP

WooCommerce

WooCommerce [8]: Es un sistema basado en WordPress que tiene una gran cantidad de extensiones gratuitas, entre las que existen las que se acercan a lo requerido por el problema a resolver [9][10]. Estas extensiones, sin embargo, tienen gran cantidad de características no necesarias lo que, sumado a los agregados que habría que realizar que son específicos a este desafío, resultaría en un sistema demasiado complejo. De todas formas, sería posible crear una extensión con las características necesarias para este sistema, lo que requeriría un trabajo en PHP usando las APIs de WordPress.

Magento

Magento [11]: Es otro sistema con gran cantidad de extensiones. En este caso, sin embargo, la que se adecúa a lo necesitado es de pago [12]. Para tener una solución basada en este software, habría que desarrollar una extensión en PHP que modifique el funcionamiento interno.

PrestaShop

PrestaShop [13]: Solución open-source de e-commerce, que dispone de un módulo de pago para la creación de marketplaces multi-vendedor [14]. Es posible crear un módulo con las características necesarias para este sistema usando el lenguaje PHP.

OpenCart

OpenCart [15]: Este sistema puede manejar distintas tiendas en simultáneo [16] las que, sin embargo deben tener subdominios distintos, y no pueden navegarse bajo un mismo sitio. Para lograr tener distintos vendedores en un mismo sitio se necesita una extensión de pago [17], o desarrollar una alternativa en PHP.

2.5.2. Rails (Ruby)

Spree

Spree [18]: Solución que cuenta con una extensión multi-vendedor [19] que no cuenta con documentación. Para usar este software como base, habría que desarrollar extensiones escritas en Ruby.

2.5.3. NodeJS (JavaScript)

Reaction

Reaction [20]: Sistema moderno basado en Node.js, React, y GraphQL, que cuenta con el concepto de múltiples tiendas [21]. Este software cuenta además con un sistema de plug-ins escritos en JavaScript [22].

2.5.4. Django (Python)

Saleor

Saleor [23]: Sistema modular de e-commerce que utiliza Python, GraphQL, Django, y ReactJS. Lamentablemente, no cuenta con soporte para múltiples vendedores ni sistema de extensiones, por lo que habría que modificar el código interno para tener los resultados deseados.

La mayoría de los softwares mencionados no están pensados para tener múltiples vendedores en el mismo sitio, por lo que sería necesario modificar de forma importante el funcionamiento interno. Las excepciones a esto son WooComerce (que cuenta con extensiones gratuitas que ya se encargan de tener múltiples vendedores) y Reaction.

Además de estos sistemas, existe un framework para la construcción de sitios de e-commerce, basado en Django (Python). Este framework es Django-Oscar [24], que cuenta con el concepto de ”Partners”[25], quienes pueden administrar productos, pero no cuentan con su catálogo separado. Dado que Oscar es un framework para la creación de sistemas de e-commerce, cuenta con una detallada documentación del funcionamiento interno, y es altamente extensible [26].

2.6. Alternativas para la implementación

2.6.1. Comparación de las plataformas de Cloud Computing

Proveedor	aws	Azure		IBM SOFTLAYER	vmware	HEROKU
Categoría informática	IaaS	PaaS	PaaS	IaaS	IaaS	PaaS
Aplicación basada en Web, Panel de control	✓	X	X	✓	X	X
Control de líneas de comandos	✓	✓	X	X	X	✓
Soporte técnico gratis	X	✓	✓	✓	-	✓
Foros de ayuda	✓	✓	✓	✓	X	X
Servicio de apoyo 24/7	✓	✓	X	X	X	X
Herramienta de diagnóstico para soporte	✓	X	X	✓	X	✓
Escalabilidad automática	Sí, a través de Amazon CloudWatch	Auto-scaling application block y Windows Azure Fabric Controller	BigTable y GFS	IBM SmartCloud Application Workload Services	VCloud Director	No, manual a través de Dynos
Blueprints para acelerar el aprovisionamiento	Sí. Imagen de máquina de Amazon	Sí. En una galería y como imágenes propias guardadas	No	Sí	Imágenes propias guardadas de máquinas virtuales VMWare	No

Proveedor	aws	Azure		IBM SOFTLAYER	vmware	HEROKU
Soporte para lenguajes	• C++ • C# • Java • Perl • Python • Ruby	• .NET • Java • Node.js • Python	• Python • Java • Go (experimental)	• Java • PHP	• Java • C# • C++	• Node.js • Ruby • Java • PHP • Python • Go • Scala • Clojure
Soporte almacenamiento de datos	• Amazon SSS, Relational DB Service, SimpleDB • SQL Server Express, Server Standard y Web	SQL Relacional Almacenes de tablas NoSQL Blob no estructurado	• Bases de datos no relacionales ”Big Table” • No soporta bases de datos relacionales	• DB2 • Oracle • MS SQL • MySQL • Informix • Sybase	• Oracle • SQL Server • VMWare vFabric • Posgres • Múltiples distribuciones de Haddop	• Heroku Postgres • Heroku Redis • Apache Kafka
Soporte para colas	Amazon Simple Queue Service	Azure Services Bus. Colas FIFO con protocoles Rest, AMQP, WS	App Engine Task Queue	WebSphere Message Broker V8.00	RabbitMQ Protocolos AMQP, MQTT y STOMP	IronMQ, CloudMQTT, RabbitMQ y otros Add-ons
Servidor Web	• Apache • IIS • Otros	• IIS V7.5	Jetty Web Server	WebSphere Application Server V7.0 y 8.0	• Apache • IIS • Otros	• Apache • Nginx
Alternativas de hipervisores	Sen y LXC (Linux Containers)	Azure Hipervisor (custom Hyper-V)	Xen/KVM	• VMWare • Hyper-V • Otros	VMWare	LXC (Linux Containers)
Caché InMemory distribuidor DataGrid	Copen: VMWare, GemFire, Oracle, Coherence, Gigaspaces SAP, etc	Windows Azure Caching/-Memcached	Memcached	WebSphere eXtreme Scale	GemFire	MemCachier

Proveedor	aws	Azure		IBM SOFTLAYER	vmware	HEROKU
Soporte sistema operativo Windows	• Windows Server 2003 R2 • Windows Server 2008 • Windows Server 2008 R2 • Windows Server 2012	• Windows Server 2012 Datacenter • Windows Server 2008 R2 SP1	No	• Windows Server 2003 • Windows Server 2008	Sí, todas las distribuciones virtualizables	No
Soporte sistema operativo Linux	• Suse Linux Enterprise Server • Red Hat Enterprise Linux	• OpenSUSE 12.3 • SUSE Linux Enterprise Server 11 Services Pack 2 • Ubuntu Server 12.04 LTS • Ubuntu Serve 12.10 • Ubuntu Server 13.04 • OpenLogic CentOS 6.3 • Ubuntu Server 12.10 DAILY	Sí, pero las aplicaciones corren en un sandbox y Google provee acceso limitado al sistema operativo, el cual, no puede ser alterado	• Red Hat Enterprise Linux • SUSE Linux Enterprise Server	Sí, todas las distribuciones virtualizables	• Ubuntu 16.04 • Ubuntu 18.04 • Ubuntu 20.04

Tab. 2.1: Tabla comparativa plataformas Cloud Computing [27]

2.6.2. *Principales plataformas de Cloud Computing*

Amazon Web Services (AWS)

Amazon Web Services (AWS) se trata de una de las plataforma de cloud computing que ofrece: almacenamiento en la nube, gestión de instancias, hosting web, desarrollo de aplicaciones móviles, entre otras.[28]

Ventajas

- Flexibilidad: AWS permite seleccionar el sistema operativo, el lenguaje de programación, la plataforma de aplicaciones web, la base de datos u otros servicios que necesites. Esta flexibilidad permite que puedas centrarte en la innovación y no en la infraestructura.

- Rentabilidad: AWS ofrece precios bajos por uso, sin gastos anticipados ni compromisos a largo plazo. Únicamente tienes que afrontar el costo de la potencia de cómputo, el almacenamiento y demás tipos de recursos que vayas a utilizar.

- Velocidad: Gracias AWS se puede reducir el tiempo en lo que tardan los recursos en estar disponibles para los desarrolladores, pudiendo pasar de semanas a minutos.

- Escalabilidad: Gracias a herramientas como AWS, Auto Scaling y Elastic Load Balancing, se tiene acceso a los recursos informáticos y de almacenamiento siempre que sea necesario.

- Seguridad: Cuenta con certificaciones y acreditaciones para administrar la infraestructura de TI de las empresas de manera segura y duradera.

Microsoft Azure

Microsoft Azure es un conjunto de servicios en la nube basados en el almacenamiento y la virtualización, con el fin de ayudar a tu organización a satisfacer tus necesidades comerciales. Con un servicio de pago por uso y totalmente escalable permite disponer de una herramienta personalizable y adaptada al 100 % a tus necesidades. Actualmente disponen de 50 centros de datos repartidos por todo el mundo pudiendo asegurar al cliente una disponibilidad total y una menor latencia con respecto a otros fabricantes. [28]

Ventajas

- Reducción de costes: El capital destinado para hardware lo ahorras trabajando en cloud.
- Seguridad: Microsoft te garantiza que tus datos siempre van a estar seguros, redundados y con una SLA del 99,9 % de disponibilidad.
- Uso operativo: Su flexibilidad permite compartir la misma plataforma que utiliza Skype, Office 365, Bing y Xbox, ofreciendo soporte técnico continuo y una supervisión puntual del estado del servicio.
- Flexible: Azure admite cualquier sistema operativo, lenguaje, herramienta y marco, permitiendo crear aplicaciones y servicios que puedan funcionar con cualquier dispositivo.
- Almacenamiento en la nube: Azure te ofrece la opción de alojamiento en espacios virtualizados, permitiéndote adquirir la capacidad necesaria bajo un esquema flexible y práctico.

Google Cloud

Google Cloud Platform es una suite que contiene diferentes servicios que funcionan en la misma infraestructura que utiliza Google de manera interna, por ejemplo como Youtube o Google Search Console. El conjunto de herramientas que proporciona la suite abarca Cloud Computing, Networking, Data Storage, Data Analytics, Machine learning, etc.

Éste provee los productos, servicios y herramientas para poder diseñar, realizar testing y lanzar las aplicaciones en la plataforma garantizando una gran escalabilidad y seguridad gracias al diseño de la infraestructura proporcionada por Google. [28]

Ventajas

- Red de Fibra Global Privada + Red Gradual: Su red es una de las más grandes del mundo. Para hacernos una idea, en 2018 anunció que estaba poniendo sus propios cables de fibra óptica bajo el Océano Pacífico.
- Migración en vivo de máquinas virtuales: Esto significa que las máquinas virtuales siempre están activas sin degradación notable en el rendimiento cuando están migrando en vivo máquinas virtuales entre máquinas host. Las migraciones en vivo permiten a los ingenieros de Google abordar mejor los problemas como el parcheo, la reparación y la actualización de software y hardware.
- Mejor rendimiento: Las máquinas de Google Cloud manejan fácilmente más de 60k visitantes concurrentes sin ningún contratiempo.
- Seguridad: Datos codificados en tránsito entre Google, los clientes y los centros de datos, autenticación y autorización del las solicitudes procedentes de otros

componentes, menos saltos a través del Internet Público, realización de auditorías regulares, etc.

- Expansión continúa: Cada vez son más los países de ubicación y no paran de aumentar continuamente.

- Respaldos de Seguridad Redundantes: El almacenamiento de Google Cloud Hosting (GC Storage) está diseñado para una durabilidad de 99.999 % y cuenta con 4 tipos diferentes de almacenamiento.

- Datacenter en Chile: Google Cloud se instaló en Chile hace siete años al levantar su centro de datos en Quilicura. Hoy, además, cuenta con Google Cloud Platform (GCP), uno de sus servicios principales. Dentro de los beneficios principales de esto es la competitividad y mejora de servicios Cloud que agrega al mercado, tanto para los clientes perdidos en GCP como para los que están evaluando la posibilidad de mover cargas de trabajo a la nube, lo cual acelera aún más la transformación digital de las empresas que pueden desarrollar nuevas soluciones de manera segura. Además, al tratarse de un nodo local, la latencia específicamente actualizada, logra mejoras en rendimiento y eficiencia de los servicios para sus procesos.

2.6.3. Comparación de precios

Para crear una comparación precisa con respecto a los precios de los diferentes servicios de Cloud Computing, se seleccionó la misma región, CPUs y sistema operativo para la configuración de computación:

- **Región:** Este de Estados Unidos – Norte de Virginia
- **Sistema Operativo:** Linux
- **vCPUs/Núcleos:** 4

A continuación, se seleccionaron instancias de máquinas virtuales con especificaciones de RAM comparables en los siguientes tipos de uso de máquinas:

- De uso general
- Optimizado por ordenador
- Memoria optimizada

Para la comparación de precios, se utilizaron los siguientes casos:

Tipo de instancia	Amazon EC2	AWS RAM (GB)	Azure VM	Azure RAM (GB)	Compute Engine	Google RAM (GB)
De uso general	t4g.xlarge	16	B4MS	16	n1-standard-4	15
Optimizado por ordenador	c6g.xlarge	8	F4	8	c2-standard-4	16
Memoria optimizada	r6g.xlarge	32	E4a v4	32	n2-highmem-4	32

Pago por hora

Los precios de pago por uso ofrecen un enfoque flexible para el consumo de recursos en la nube. Esta opción, ideal para un uso intermitente de la nube, permite añadir y eliminar recursos de la nube en función de la demanda. Sin embargo, esta flexibilidad tiene un coste, ya que los modelos de precios de pago por uso tienen el precio más alto por hora:

Tipo de instancia	Amazon EC2	AWS Price (USD/h)	Azure VM	Azure Price (USD/h)	Compute Engine	Google Price (USD/h)
De uso general	t4g.xlarge	**$0.134**	B4MS	$0.166	n1-standard-4	$0.150
Optimizado por ordenador	c6g.xlarge	**$0.136**	F4	$0.199	c2-standard-4	$0.188
Memoria optimizada	r6g.xlarge	**$0.201**	E4a v4	$0.252	n2-highmem-4	$0.295

Al comparar los precios de las máquinas virtuales de AWS con los de Azure y Google Cloud, Amazon EC2 es el claro vencedor en los tipos de instancias de propósito general, optimizadas para el cálculo y optimizadas para la memoria. AWS es como mínimo un 20 % más barato en las tres categorías.

Planes a Largo Plazo

Un plan a largo plazo permite obtener ahorros significativos en comparación con el modelo de pago por uso.

Los servicios de Cloud Computing frecen un modelo de precios a largo plazo con opciones de compromiso por adelantado de 1 o 3 años. Las instancias reservadas suponen un ahorro de costes de hasta el 72 % con respecto a los precios de pago bajo demanda.

1 año

Tipo de instancia	Amazon EC2	AWS Price (USD/h)	Azure VM	Azure Price (USD/h)	Compute Engine	Google Price (USD/h)
De uso general	t4g.xlarge	**$0.079**	B4MS	$0.097	n1-standard-4	$0.125
Optimizado por ordenador	c6g.xlarge	**$0.080**	F4	$0.124	c2-standard-4	$0.141
Memoria optimizada	r6g.xlarge	**$0.118**	E4a v4	$0.148	n2-highmem-4	$0.177

AWS es un 20 % más barato que Azure en los tipos de instancias de propósito general y optimizadas para memoria, y un impresionante 40 % más barato en los tipos de instancias optimizadas para computación. Diferencia similar se registra en comparación con los precios de Compute Engine de Google.

3 años

Tipo de instancia	Amazon EC2	AWS Price (USD/h)	Azure VM	Azure Price (USD/h)	Compute Engine	Google Price (USD/h)
De uso general	t4g.xlarge	$0.050	B4MS	$0.062	n1-standard-4	**$0.046**
Optimizado por ordenador	c6g.xlarge	**$0.051**	F4	$0.078	c2-standard-4	$0.094
Memoria optimizada	r6g.xlarge	**$0.075**	E4a v4	$0.099	n2-highmem-4	$0.126

Amazon EC2 continúa siendo más barato en general al hacer un compromiso de 3 años frente a los otros 2 servicios de Cloud Computing. Sin embargo, existe una excepción en la categoría de Propósito General, Compute Engine se opone a la tendencia y es más barato al realizar un compromiso de 3 años con la plataforma.

3. PROPUESTA DE SOLUCIÓN

3.1. Objetivo Principal

El objetivo principal del proyecto es crear una plataforma e-Commerce. Esta plataforma pretende ser un Marketplace del tipo B2C, que permita la comunicación entre las empresas del segmento MiPymes, que es el sector al que está orientado el proyecto, con sus potenciales compradores, y la generación de transacciones de compra desde el catálogo de venta ofrecido por cada empresa. Junto con esto, el sitio web debe ser confiable y escalable, para ello se evaluarán los distintos servicios de Cloud Computing para montar los servidores de la plataforma, y se diseñará un entorno de red que permita asegurar la fiabilidad, escalabilidad, mantenibilidad y seguridad del sistema. A nivel de negocio, esta es una gran oportunidad para el sitio de InfoRed de generar ingresos por motivo de comisión por ventas y publicidad, permitiendo a las empresas colocar anuncios dentro de la página para promocionar sus productos, esto también, con un costo adicional asociado.

3.2. Objetivos Específicos

1. Obtener información del segmento MiPymes, sobre la recepción respecto a la implementación de una plataforma web e-Commerce en InforedChile. Además del método de obtención preferencial de este servicio por parte de los encuestados, la posibilidad de obtención de otros servicios de asesoría e ideas adicionales que puedan añadir al servicio ofrecido.

2. Determinar la mejor visión general de negocio, que contenga una propuesta de venta como de valor, atributos, plataformas de uso y dispositivos, diferenciación, descripción y tendencias de la startup.

3. Elaborar la mejor opción de modelo de negocio, según propuestas de valor y venta de la plataforma e-commerce a implementar, y conectar su flujo con los respectivos stakeholders implicados.

4. Desarrollar un sistema de e-commerce multi-vendedor. El sistema debe separar las tiendas de las empresas, cada una con su propio carro de compras, y catalogo visible de forma separada.

5. Actualizar el indexador del motor de búsqueda de la página principal al agregar y remover productos del marketplace, considerando palabras claves de los productos en cuestión.

6. Generar y utilizar una API para obtener información de productos de forma externa, y poder buscarlos por nombre o vendedor. La utilización de la API se refiere a la creación de un script que pueda incrustar resultados en la página principal.

7. Asociar la vista de detalles de empresas en el sitio principal, con el catalogo de dicha empresa en el marketplace (de tener productos). Desplegar junto a los detalles una lista de productos de su catalogo, utilizando la API, con un enlace al marketplace.

8. Integrar y validar un sistema de pago con tarjetas de crédito y débito chilenas. Este pago debe ser ofrecido para los productos solo luego de que haya una confirmación del vendedor.

9. Evaluar el rendimiento general de la empresa a través de la medición separada del rendimiento digital de esta y el rendimiento de captura de valor.

10. Con el desarrollo de herramientas de marketing identificar mejoras del modelo de negocio creado, y analizar los próximos desafíos de la empresa.

3.2.1. Requisitos del Sistema

Requisitos Funcionales

RF1 El sistema debe ofrecer un marketplace de productos. Esto es: se presentan distintos productos a un precio determinado.

RF2 El sistema debe ser accesible para todas las empresas registradas en el sitio principal.

RF3 Cada empresa debe poder administrar su y solo su catálogo de productos. Estos catálogos pueden verse en la página de forma independiente.

RF4 El sistema debe permitir el registro de usuarios, los que pueden explorar y buscar productos en los catálogos.

RF5 El sistema debe diferenciar entre usuarios y empresas, con privilegios distintos en el sistema.

RF6 El sistema debe funcionar de manera independiente a la página principal de InfoRedChile.

RF7 El sistema debe poder interactuar con el indexador del motor de búsqueda del sitio principal, pudiendo realizar actualizaciones en este de acuerdo a los productos que se agreguen.

RF8 El sistema debe ser capaz de incrustar resultados de búsqueda de productos en el buscador del sitio principal.

RF9 El sistema debe ser capaz de incrustar una lista de productos del catálogo de cierta empresa, al ver detalles de dicha empresa en el sitio principal.

RF10 Los usuarios deben poder agregar productos de un catálogo a un carro de compras, y solicitar los productos o servicios.

RF11 Estos carros de compra de los usuarios deben ser independientes para cada catálogo de empresas distintas.

RF12 El sistema debe permitir al usuario pagar por los productos y/o servicios, una vez que la empresa haya confirmado la disponibilidad.

Requisitos no Funcionales

RNF1 El diseño estético del sistema debe concordar con el diseño de la página principal.

RNF2 El sistema debe funcionar 24/7.

RNF3 El sistema debe funcionar con el orden de miles de usuarios diarios.

RNF4 Fácil de usar por los clientes y de administrar.

RNF5 Los pagos en el sistema deben realizarse de forma segura.

3.3. Arquitectura del Sistema

3.3.1. Diagrama de Contexto

En el diagrama de contexto (Figura 3.1), el sistema está graficado como una caja negra, junto a los principales entes externos al sistema que interactúan con él. Las flechas que unen los entes externos con el sistema indican flujos de datos desde o hacia el sistema.

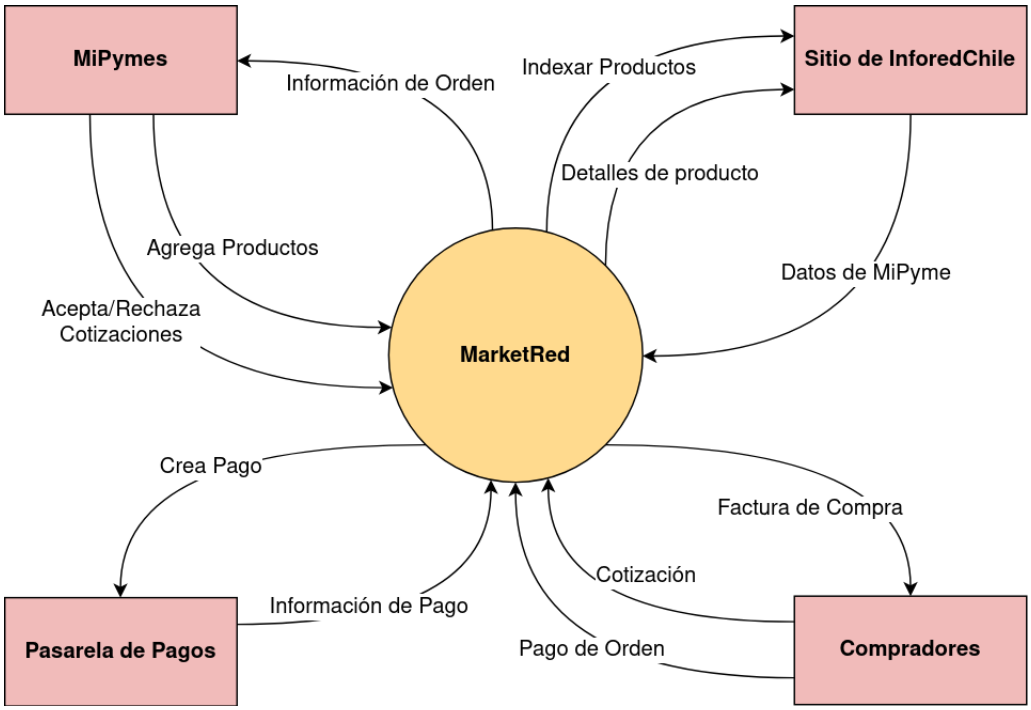


Fig. 3.1: Diagrama de contexto del sistema

3.3.2. *Diagrama de Arquitectura*

La Figura 3.2 muestra los componentes internos del sistema y sus principales interacciones.

Dado que 'MarketRed API' está compuesto por una gran cantidad de complementos con diversas interacciones, se listan los que deberán ser creados específicamente para esta solución (con borde grueso), junto a los que interaccionan fuertemente con estos (con borde delgado).

Además, hay 2 interfaces o 'FrontEnds'. Por un lado, la interfaz denominada 'Tiendas' es por la cual se muestran públicamente los productos, y donde se pueden realizar compras. Por otro lado, 'Admin' es la interfaz donde solo pueden acceder las MiPymes, y donde se publican los productos.

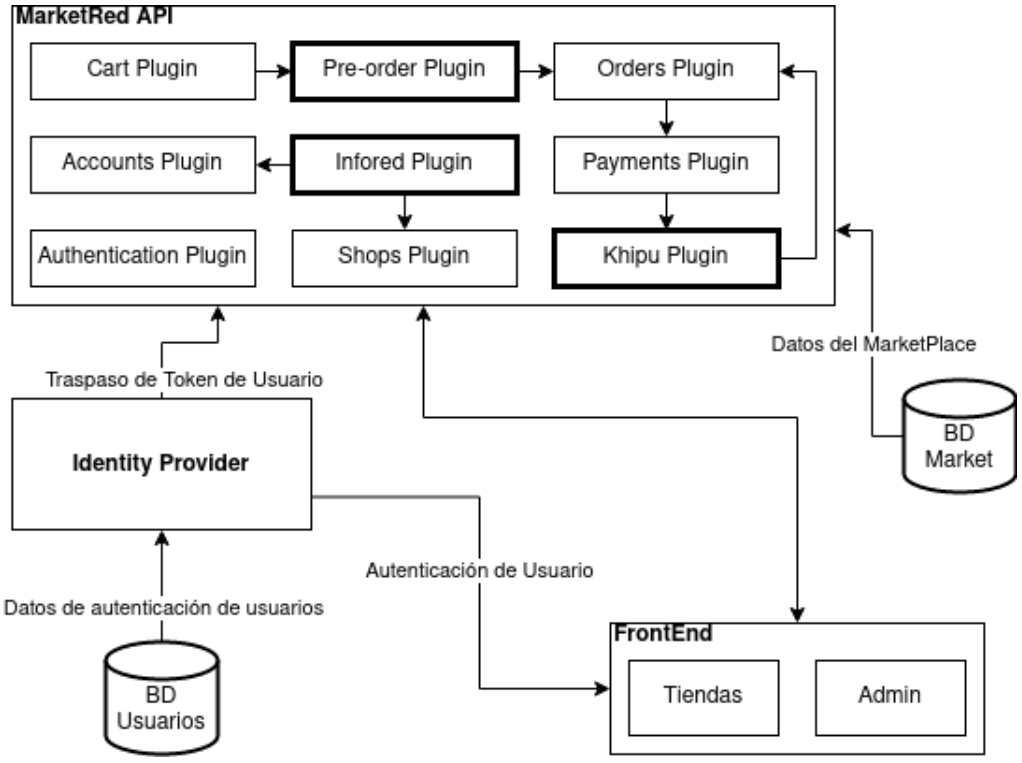


Fig. 3.2: Diagrama de arquitectura del sistema

3.4. Descripción de Componentes.

A continuación se describen los componentes del sistema. Por cada componente se entrega un breve párrafo descriptivo de su propósito además de la sección en donde se especifica el componente en detalle.

3.4.1. MarketRed API

Propósito	Manejar los datos y la lógica de negocios, exponiendo una API a ser consumida por el resto de los módulos.
Alcance	El componente principal que presenta y muta los datos de acuerdo a las solicitudes recibidas.
Dependencias	Es necesario el funcionamiento de la base de datos.
Supuestos	Se ejecuta sobre un kernel Linux para arquitectura x86_64.
Restricciones	Ninguna.
Estructura General	A pesar de estar representado como un solo componente, la funcionalidad interna está separada en múltiples complementos. Los datos son expuestos a través de una API GraphQL, mediante la cual se pueden recibir consultas o mutaciones. Los complementos pueden interactuar internamente entre ellos por medio de estas mismas mutaciones. En adición a los complementos por defecto del sistema Reaction, hay 3 principales que se deben adicionar para el funcionamiento de esta solución: **Pre-order Plugin**, **Infored Plugin** y **Khipu Plugin**.

Definición del Complementos

Pre-order Plugin

Este plugin es el encargado de generar y enviar la cotización a la empresa a la que se encuentran asociados los productos del carrito de compra. Aquellos productos que sean exclusivos de la tienda en la que se está realizando la cotización, se recogen los datos de identificación y los comentarios que pueda realizar el comprador. Se envía la cotización a la empresa que es la encargada de aceptarla y generar la orden respectiva de venta, o de lo contrario, rechazarla. En ambos casos el comprador recibe una notificación con la orden y la posibilidad de pagar o una eventual cancelación de la cotización con el comentario de las razones por parte de la empresa.

Infored Plugin

Dado que hay ciertos datos relativos a las MiPymes que son manejados en el sistema de InforedChile, como nombres y datos de contacto, y que son necesarios conocer en MarketRed, se diseña este complemento que tiene el propósito de mantener dichos datos sincronizados. Para esto, se planea ofrecer una mutación GraphQL, la cual será utilizada por el sistema InforedChile cada vez que haya cambios en los datos de cierta MiPyme.

Khipu Plugin

El plugin para interactuar con la pasarela de pagos de Khipu, para crear cobros y recibir pagos. Después de que el cliente elige sus productos y opta por pagar, comienza el proceso de pago. La mayoría de los software de comercio electrónico funcionan de manera similar. Lo que se debe hacer es relacionar la orden de compra con un pago utilizando la API REST de Khipu. [29] Dado que los pagos no son inmediatos, y tienen un plazo para completarse, este complemento además tiene la función de cambiar el estado de la orden una vez la operación de pago haya terminado.

3.4.2. Front End

Este componente está compuesto por dos partes: por un lado, una aplicación web NextJs, y por otra una aplicación Meteor. Su descripción se muestra a continuación.

Propósito	Exponer a los usuarios las interfaces gráficas para interactuar con el sistema
Alcance	Procesa las distintas solicitudes de los usuarios, llamando las mutaciones correspondientes de la API. Recibe de parte de IP la información de autenticación en caso que corresponda.
Dependencias	Depende de que la MA exponga la API para consumirla, y de IP para realizar acciones que requieran autenticación.
Supuestos	Ninguno
Restricciones	Ninguna
Estructura General	Este componente está separado en 2 'frontends' distintos: uno para la visualización de las tiendas, y otro para la administración de estas (que requiere autenticación). Ambos usan apollo-client para la interacción con la API de MA y el manejo de estado, usando React para el renderizado de los elementos visuales.

3.4.3. Identity Provider

Este componente es una aplicación web desarrollada en Meteor.

Propósito	Manejar los datos de autenticación de los usuarios y MiPypmes.
Alcance	Ante la petición de inicio de sesión de FE, autentica el usuario, retornando a FE un token. FE utiliza este token con MA, quien finalmente comprueba la validez con IP.
Dependencias	Es necesario el funcionamiento de la base de datos con los datos de ingreso de los usuarios.
Supuestos	El canal de comunicación IP-MA es seguro.
Restricciones	Ninguno
Estructura General	Este componente se divide en: 1) Ory Hydra, un servidor que gestiona el flujo de autenticación OpenID y 2) Un proveedor de identidad, que maneja los datos de inicio de sesión de los usuarios comunes, o interactúa con el sitio de InforedChile para autenticar a las MiPymes.

Este módulo utiliza el sitio actual de InforedChile como proveedor de usuarios para los clientes de InforedChile que podrán administrar sus empresas. Debe manejarse de forma distinta el acceso cuando el cliente es el ”frontend”de administración, donde solo MiPymes pueden acceder, a cuando es el cliente de la tienda.

Esto se logra detectando en el proveedor de usuarios MarketRed cuando la solicitud proviene del “cliente de administración”, y redirigiendo en ese caso al agente de usuario al sitio de InforedChile. Adicionalmente, se puede ofrecer una opción de “Inicio de sesión con cuenta InforedChile” en el “cliente de la tienda”, para que realice igualmente esta redirección.

3.5. Diseño del Sistema.

3.5.1. Modelos de Navegación.

Un modelo de navegación permite entender el contexto en el que se va a crear la estructura, de esta forma se puede observar el flujo de interacciones que debe realizar un usuario para acceder a las distintas funcionalidades de la plataforma.

La Figura 3.3 muestra los pasos que debe realizar un usuario que desea comprar un producto dentro del marketplace. Desde realizar la cotización, la espera de la confirmación y la emisión de la orden, hasta la posibilidad de pagar dicha orden o cancelarla.

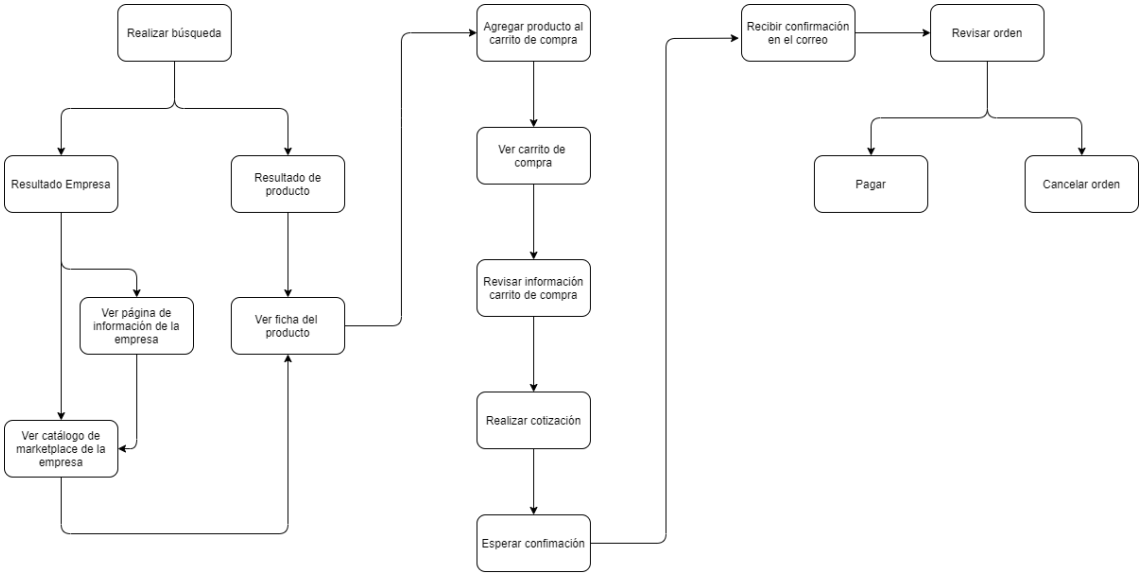


Fig. 3.3: Modelo de navegación de compra

En caso de que el usuario no se encuentre logueado en el sistema cuando reciba la confirmación de la orden o desee ver las cotizaciones que ha enviado y se encuentran pendientes de confirmación, puede iniciar sesión en la página y acceder a estas opciones desde su cuenta de usuario, dichas interacciones se encuentran diagramadas en la Figura 3.4

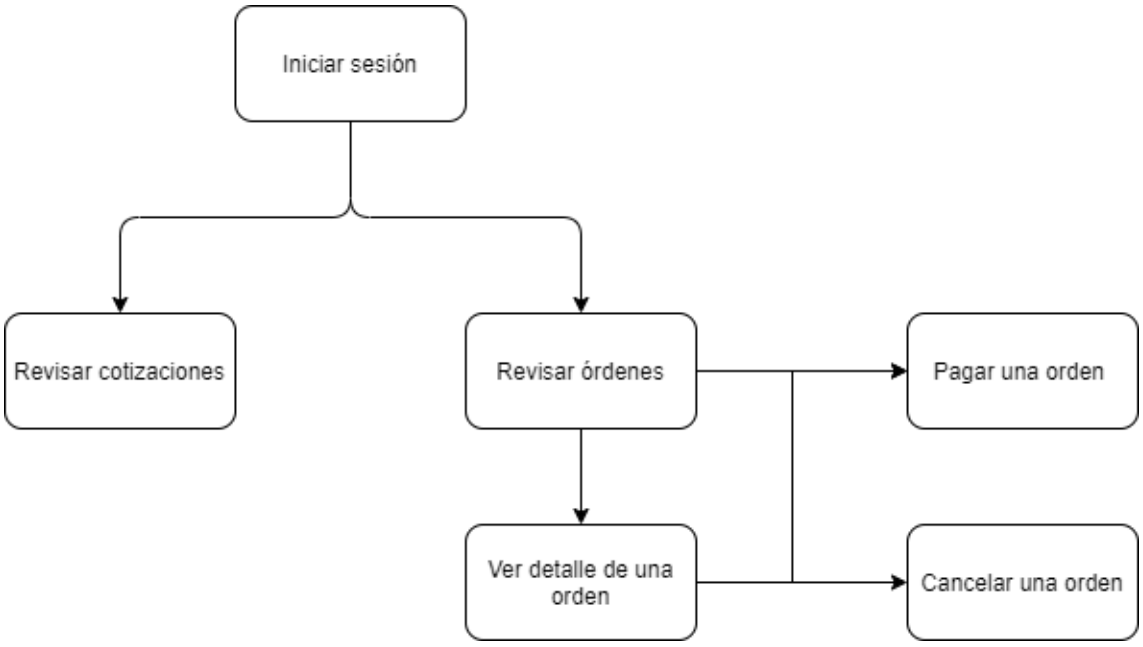


Fig. 3.4: Modelo de navegación de usuario para revisar cotizaciones y órdenes de Infored

Finalmente, podemos ver del lado de la empresa, en la Figura 3.5, las opciones que tiene posterior a iniciar sesión dentro de la plataforma.

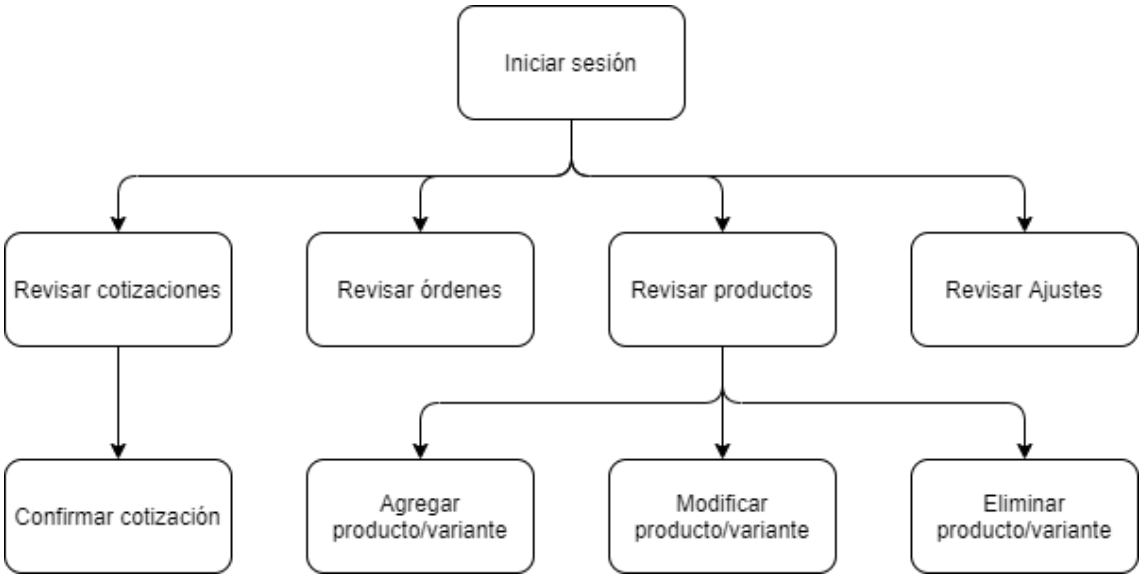


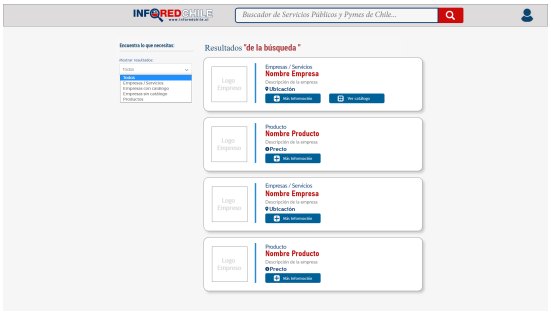
Fig. 3.5: Modelo de navegación de una empresa registrada en Infored

3.5.2. Diseño de Interfaces Usuarias

El diseño de interfaces usuarias permite realizar un enfoque de la plataforma basada en la experiencia de usuario y su interacción, de esta forma, ayuda a que las aplicaciones sean más atractivas y además, a hacer que la interacción con el usuario sea lo más intuitiva posible.

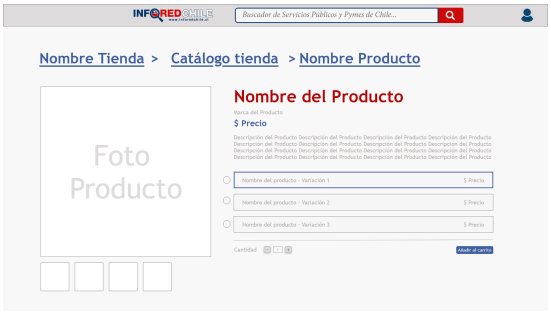
A continuación se presenta el diseño de las interfaces usuarias propuesto para el desarrollo de la plataforma de marketplace del sitio InforedChile, creado con AdobeXD.

(a) Página de inicio InforedChile [30]

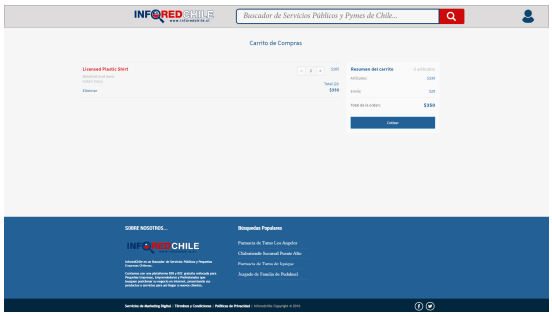


(b) Página de resultados de búsqueda

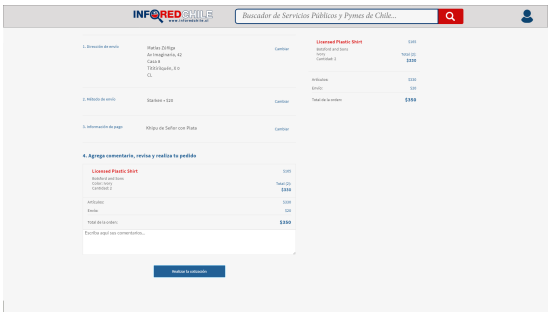
(c) Catálogo de empresa / servicio



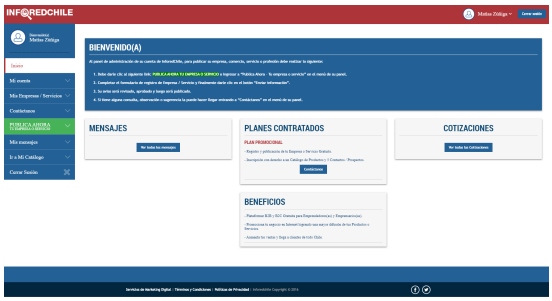
(d) Ficha de producto



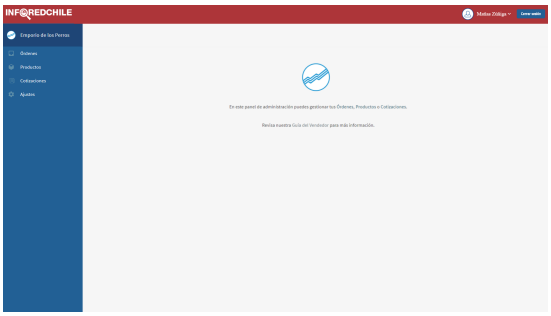
(e) Carrito de compra



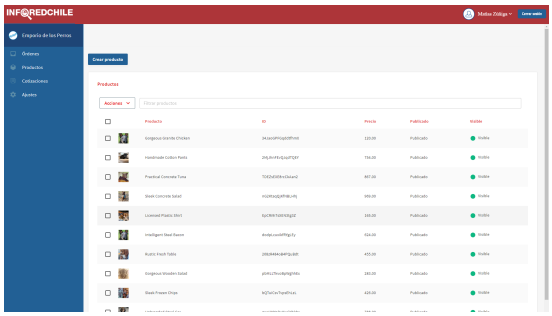
(f) Detalle de la compra, previo a la cotización



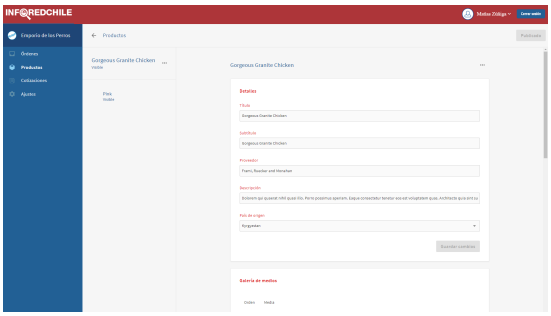
(g) Perfil de empresa al iniciar sesión (sitio actual de Infored)



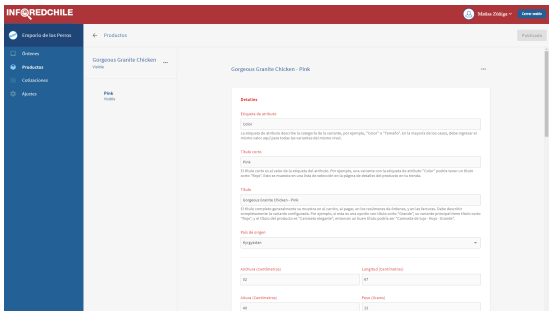
(h) Perfil de empresa al iniciar sesión (propuesta)



(i) Página de productos de la empresa



(j) Página para agregar producto



(k) Página para agregar variante de un producto

3.6. Propuesta para asegurar la confiabilidad del sistema

3.6.1. Confiabilidad del sistema

Conceptos claves para la confiabilidad de un sistema:

- **Fiabilidad:** Significa que incluso si hay una falla, el sistema aún puede funcionar normalmente. Los fallos pueden ser de hardware (normalmente aleatorios y no relacionados), software (los defectos suelen ser sistemáticos y difíciles de tratar) y humanos (que inevitablemente cometen errores de vez en cuando). La tecnología tolerante a fallas puede ocultar ciertos tipos de fallas a los usuarios finales.

- **Escalabilidad:** Esto significa que existen estrategias para mantener el rendimiento incluso cuando aumenta la carga. Es una propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

- **Mantenibilidad:** El término mantenibilidad se refiere a la facilidad o dificultad que se debe enfrentar para corregir fallos o introducir nuevas características a una aplicación luego de su puesta en producción. Actualmente la mantenibilidad o facilidad de mantenimiento es una de las características más importantes que se busca al desarrollar una aplicación y muchas veces se le asigna un valor mucho mayor que a otras.

- **Seguridad:** seguridad web son las medidas aplicadas para proteger una página web y garantizar que los datos no están expuestos ante los cibercriminales. En este sentido, la seguridad web es un proceso continuo y una parte esencial de administrar un sitio web.

3.6.2. *Diagrama de red interna del sistema*

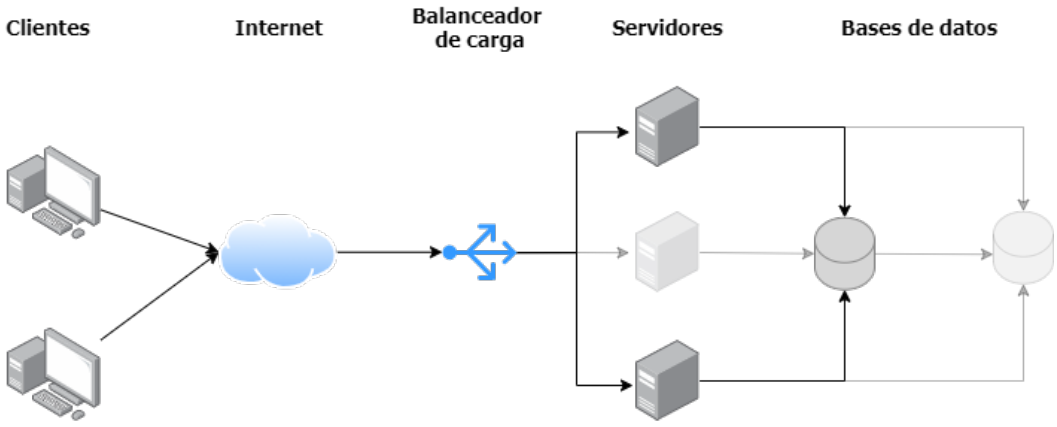


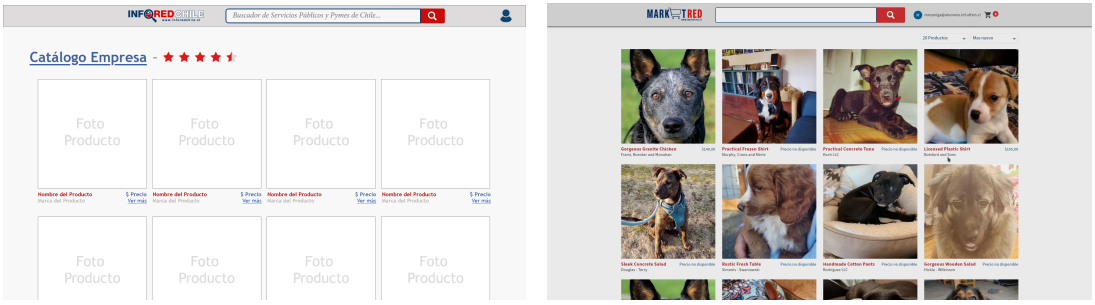
Fig. 3.8: Propuesta de red interna del sistema.

Lo que se propone para la estructura interna de la red es colocar un balanceador de carga con acceso a internet que sea el encargado de distribuir el flujo de visitas en primera instancia, entre dos servidores que se encontrarán activos. Además de eso se pretende implementar un servidor, pero que se encuentre inactivo, pasando a estarlo de acuerdo a la necesidad, por ejemplo, cuando ocurra un evento como Black Friday o Cyber Day en que la cantidad de visitas al sitio aumente.

Respecto a las bases de datos, la idea es similar: Tener una base de datos principal y otra tipo respaldo, la cual cada cierto tiempo se vaya actualizando con la información de la base de datos principal, en caso de que exista algún problema o error con la base de datos principal, poder hacer una restauración utilizando la información de la base de datos inactiva. Se maneja la posibilidad de utilizar un balanceador de carga interno que maneje los requerimientos que se hagan a las bases de datos, pero dicha posibilidad se evaluará al momento de la implementación.

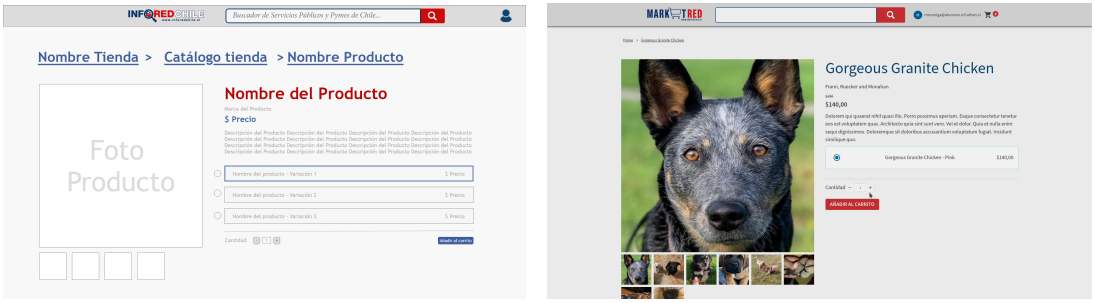
4. RESULTADOS

4.1. Interfaces Implementadas en el Prototipo.



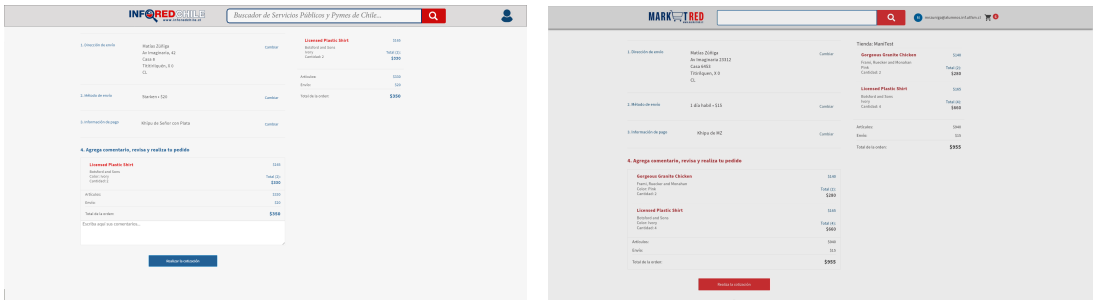
(a) Diseño propuesto (b) Interfaz implementada

Fig. 4.1: Catálogo de tienda



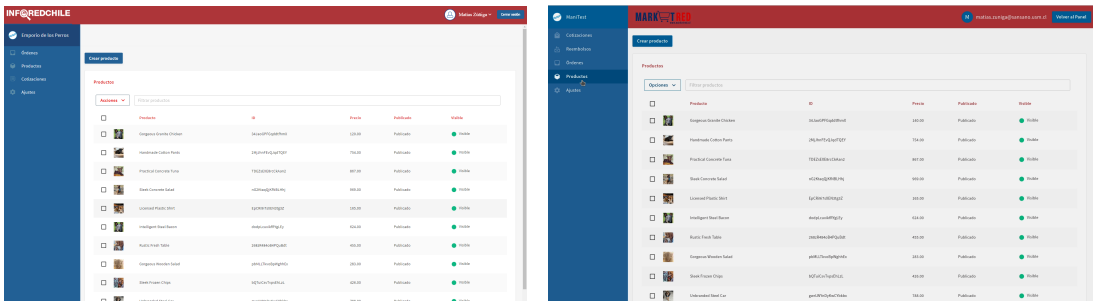
(a) Diseño propuesto (b) Interfaz implementada

Fig. 4.2: Página del producto



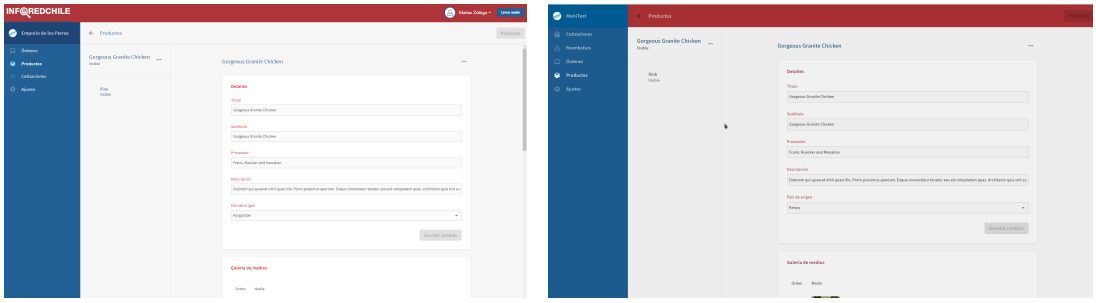
(a) Diseño propuesto (b) Interfaz implementada

Fig. 4.3: Carrito de compra



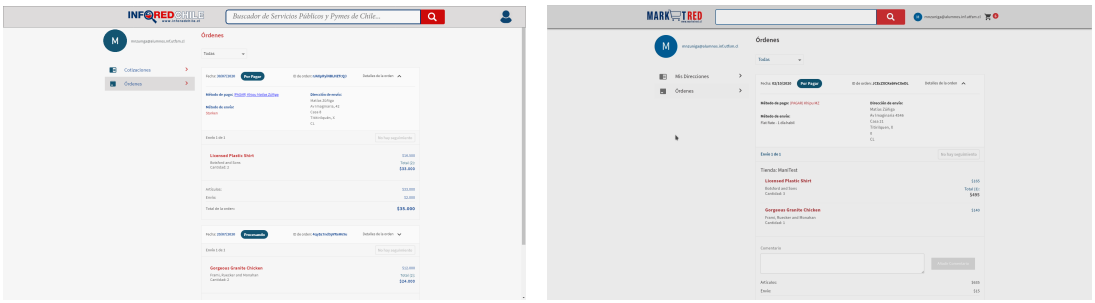
(a) Diseño propuesto (b) Interfaz implementada

Fig. 4.4: Página de productos



(a) Diseño propuesto (b) Interfaz implementada

Fig. 4.5: Agregar/editar producto



(a) Diseño propuesto (b) Interfaz implementada

Fig. 4.6: Página de órdenes

4.2. Prototipo en plataforma de Cloud Computing.

Para el levantamiento del sitio web de e-commerce se utilizó la plataforma de Cloud Computing de Amazon: Amazon Web Service, principalmente por las herramientas que proporciona, como por ejemplo el Auto Escalado (Auto Scaling) que permite optimizar los costos y el nivel de desempeño monitoreando las aplicaciones y ajustando automáticamente su capacidad o el Elastic Load Balancing que permite distribuir automáticamente el tráfico de aplicaciones entrantes entre varios destinos y dispositivos virtuales en una o varias zonas de disponibilidad, características que se busca al momento de implementar esta aplicación.

Adicionalmente lo flexible que es al permitir seleccionar el sistema operativo, el lenguaje de programación, la plataforma de aplicaciones web, la base de datos y otros servicios que sean necesarios.

Otra razón para la elección de AWS, fue el hecho de ofrecer precios bajos por uso, así como se pudo comprobar en la sección 2.6.3, además ofrece un nivel gratuito que no caduca y está disponible para todos los clientes, con el cual se pueden realizar pruebas y crear prototipos con recursos limitados antes de desarrollar un proyecto a gran escala.

Una vez decidida la plataforma de Cloud Computing que se iba a utilizar, se buscaron referencias para la implementación y se realizaron dos cursos online, que ayudaron como base para la realización de esta parte.[31] [32]

4.2.1. Consideraciones generales

Nuestra infraestructura cuenta con los tres servicios estándar de Reaction Commerce:

- El backend de Reaction Commerce (API y la interfaz de usuario de administración de Catalyst)
- El storefront
- Hydra y su base de datos PostgreSQL

Hydra es un proveedor de OAuth 2.0 y OpenID Connect. En otras palabras, una implementación del marco de autorización de OAuth 2.0, así como del marco OpenID Connect Core 1.0. Como tal, emite tokens de identificación, actualización y acceso de OAuth 2.0 que permiten a terceros acceder a sus API en nombre de sus usuarios, esto permitirá la gestión y autenticación de los usuarios que accedan al marketplace. Hydea requiere de la base de datos PostgreSQL para funcionar. El también llamado Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto

Para almacenar los datos de los productos que se publiquen en el marketplace se utilizará una base de datos MongoDB que será alojada en MongoDB Atlas. MongoDB es una base de datos NoSQL que almacena datos en documentos tipo JSON con esquemas dinámicos, simplifica la integración de datos y ofrece mejor escalabilidad que las bases de datos relacionales tradicionales. Los sitios de comercio electrónico de hoy en día necesitan modelos de datos enriquecidos y consultas dinámicas. MongoDB proporciona esto, por lo que es una opción popular para sitios de ecommerce.

Utilizaremos la función de emparejamiento de VPC para asegurarnos de que nuestras conexiones permanezcan dentro de nuestro centro de datos de AWS sin pasar por Internet, mejorando la seguridad y el rendimiento al mismo tiempo.

Para esta implementación es necesario además tener un dominio registrado, para que se pueda posteriormente acceder a los servidores de nuestra aplicación desde internet. En este caso se registró un dominio gratuito mediante la página `https://www.freenom.com`. El dominio registrado es `https://marketred.ml`

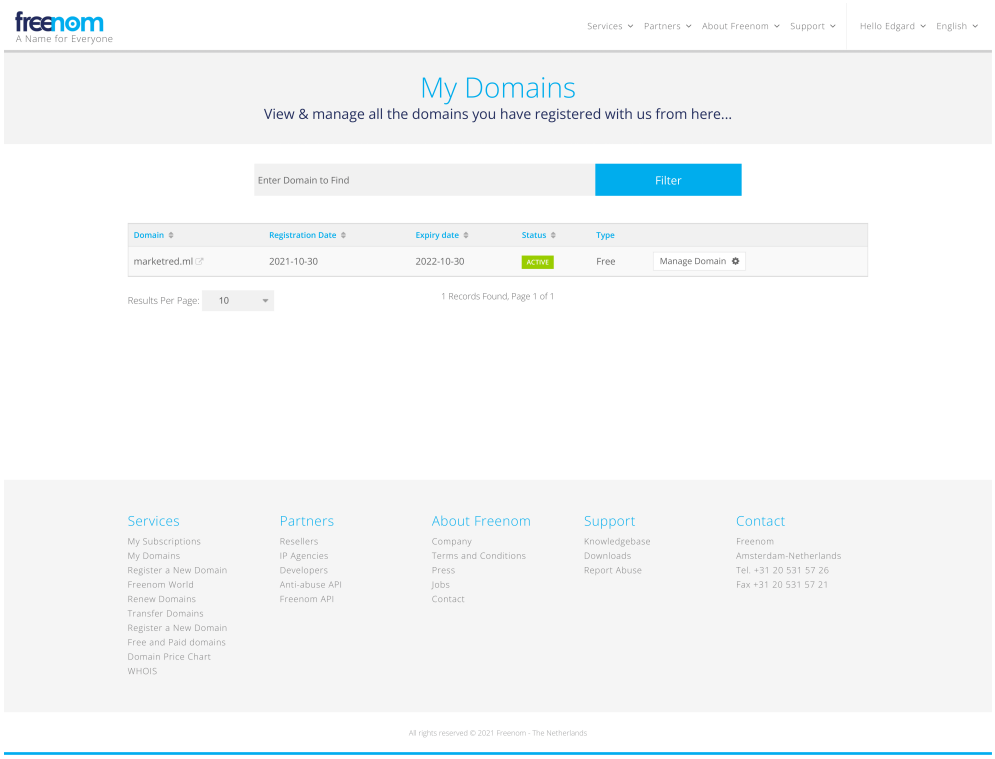


Fig. 4.7: Dominio en Freenom

Dicho dominio lo vamos a administrar con AWS para que podamos usarlo en nuestra implementación. Para eso se utilizó AWS Route 53, en el que básicamente se registra una nueva red hospedada y se copian los valores de DNS provistos en AWS a nuestro proveedor de dominio.

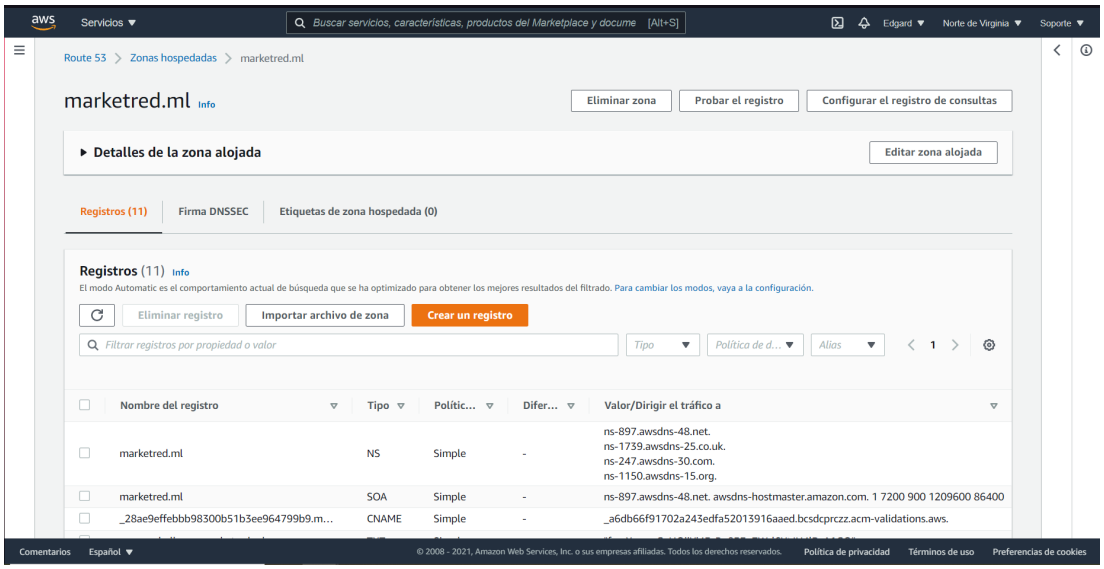


Fig. 4.8: AWS Route 53 - Zonas Hospedadas

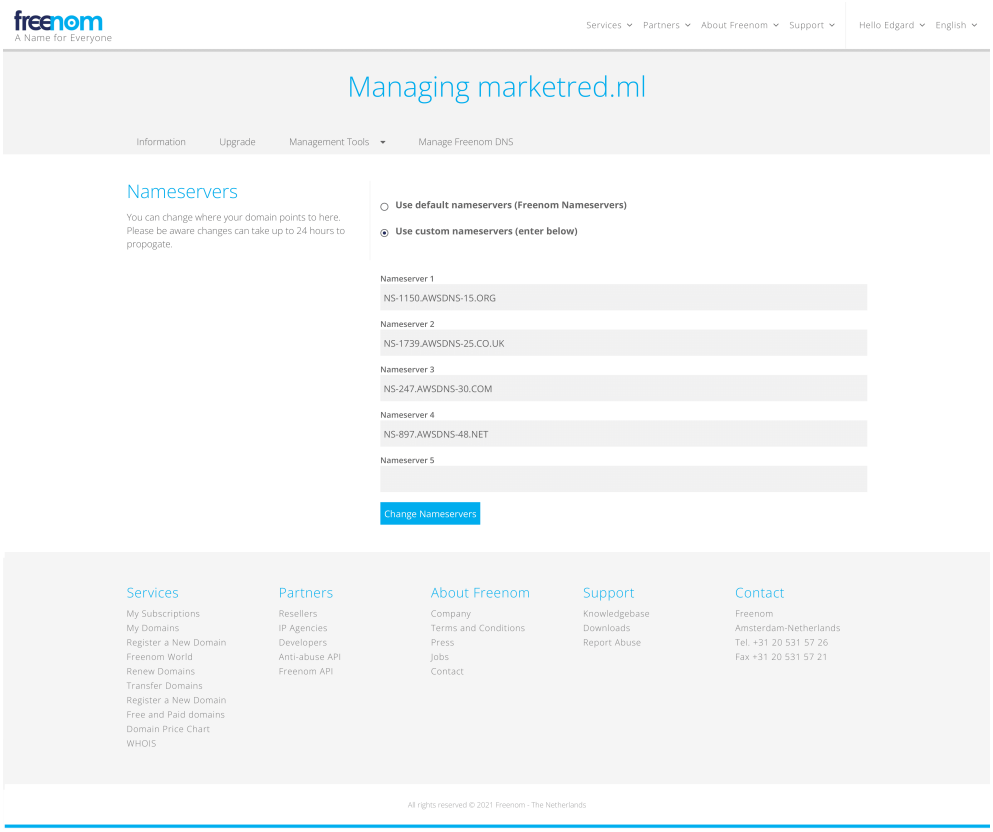


Fig. 4.9: Freenom - Managing domain

4.2.2. Creación de cluster en MongoDB y emparejamiento con AWS

En primera instancia se configura un clúster de MongoDB en Atlas. En el panel de MongoDB Atlas, se crea un nuevo clúster.

Se selecciona AWS como proveedor de nube y es necesarios seleccionar la misma región de AWS en la que desea implementar su clúster de ECS, en este caso, N. Virginia (us-east-1). De esta manera, el backend de Reaction se podrá comunicar directamente con la base de datos mediante el emparejamiento de VPC, utilizando solo la red interna de AWS, sin pasar por Internet.

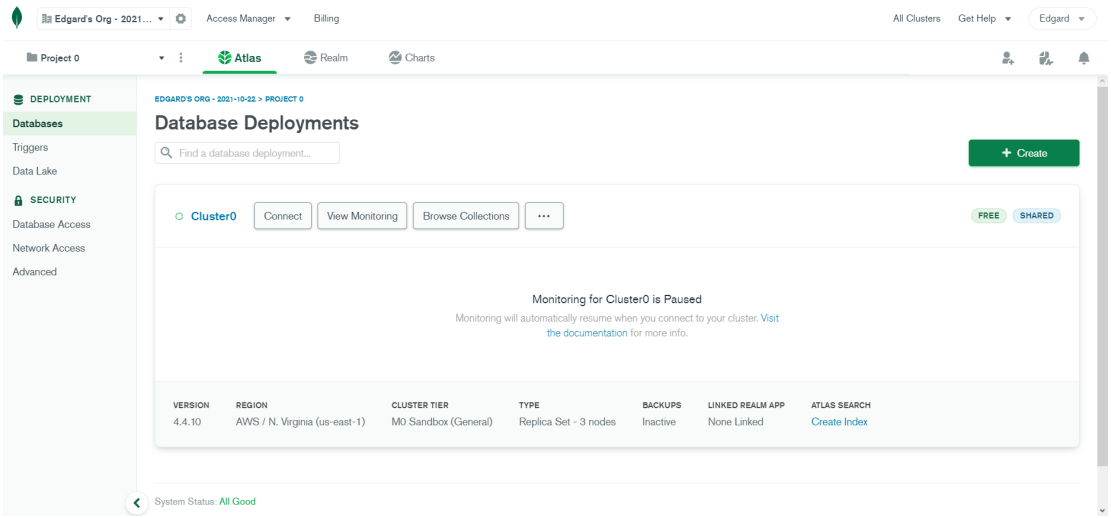


Fig. 4.10: Base de datos implementada

Ahora es necesario vincular el clúster Atlas a la cuenta de AWS mediante la función VPC Peering. Esto permitirá que el clúster de ECS se comunique con su clúster de Atlas directamente utilizando la red de AWS, ya que estarán en el mismo centro de datos. De esa manera, evitamos completamente las conexiones con el mundo exterior, lo cual agrega una capa adicional de seguridad.

Para ello en el panel de MongoDB Atlas, en la sección de “Acceso a la red”, en la pestaña de “Peering” se establece una “Nueva Conexión de Emparejamiento” (New Peering Connection). Se selecciona AWS como proveedor de nube y escribir los siguientes datos asociados a la cuenta de AWS:

ID de la cuenta

En “Mi cuenta”, el ID de cuenta es la primera credencial de la lista.

ID de VPC y CIDR

En el panel de VPC, en “Your VPCs” en la barra lateral izquierda. En la lista, se selecciona la VPC predeterminada. Se copia y pega su ID y CIDR IPv4 para completar el formulario de emparejamiento Atlas.

Región de VPC

Simplemente se selecciona la región de AWS en la que está implementando. Como habíamos definido antes, en este caso, estamos utilizando us-east-1.

Con toda la información requerida completa, se “Iniciar intercambio de tráfico”. La conexión de emparejamiento ahora se muestra como “Esperando aprobación”.

Dicha aprobación se hace en el panel de VPC de AWS en “Peering Connections” donde se debe visualizar la opción y dentro de las acciones la posibilidad de aceptarla.

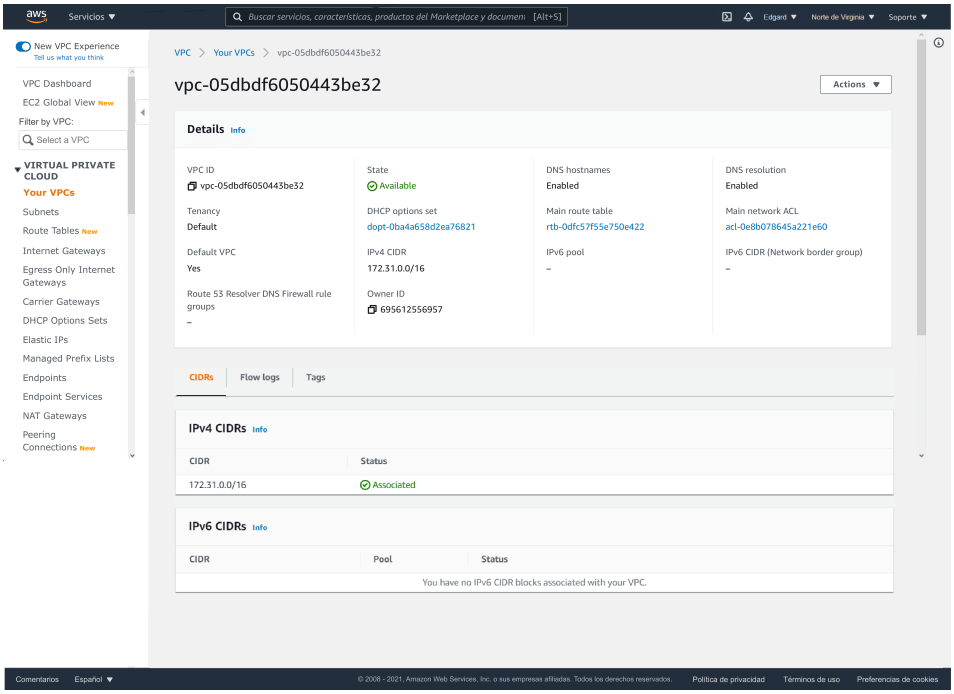


Fig. 4.11: Dashboard VPC

Ahora es necesario modificar la tabla de rutas, lo cual se puede realizar accediento a “Tablas de rutas” en la barra lateral del Panel de VPC en AWS. Hay que editar las rutas de la tabla predeterminada de enrutamiento. A la cual se le debe agregar una nueva ruta con el Atlas CIDR que se encuentra en panel de intercambio de tráfico en la consola Atlas y en el “Destino”, se selecciona “Conexión de intercambio de tráfico” y el ID de su conexión de intercambio de tráfico.

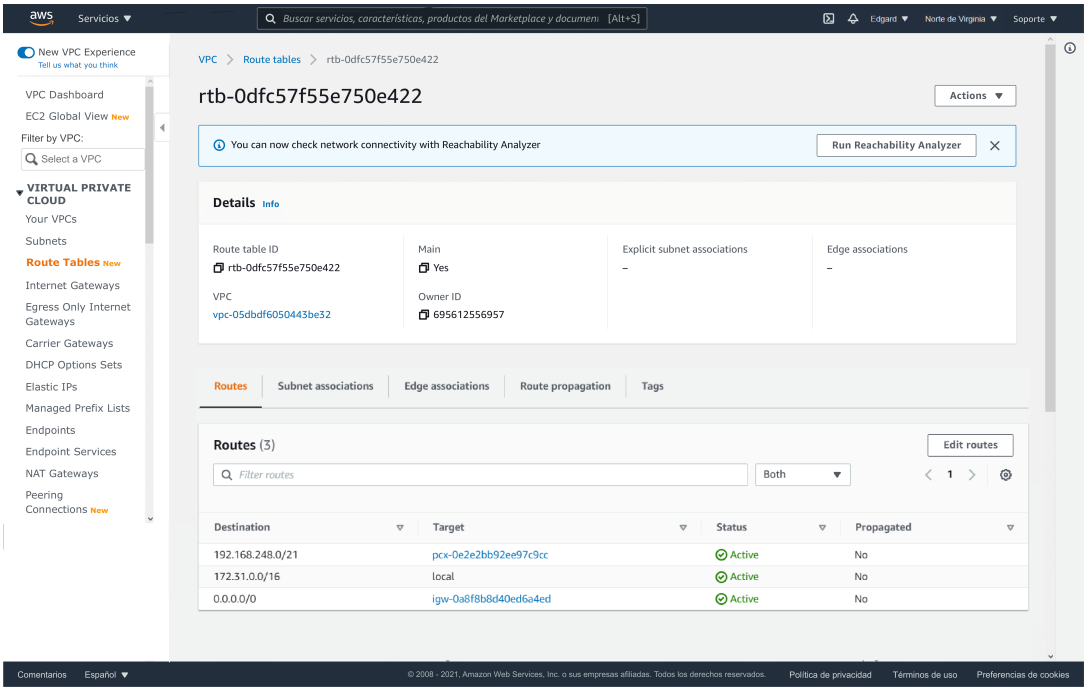


Fig. 4.12: VPC Routes

4.2.3. Creación de la base de datos PostgreSQL de Hydra en RDS

Hydra necesita una base de datos PostgreSQL para funcionar correctamente, por lo que se creó una con AWS RDS.

Buscamos “RDS” dentro de los servicios de AWS y luego el el panel de RDS, se busca la opción “Crear base de datos” y se selecciona PostgreSQL como motor de base de datos. La última versión de PostgreSQL siempre debería funcionar.

Se utilizó “hydra” como identificador de instancia de la base de datos, “postgres” como nombre de usuario maestro y se creó una contraseña maestra.

En “Tamaño de instancia de base de datos”, se seleccionó una micro instancia T2. Esto funciona bien para implementaciones pequeñas y medianas como en este caso. En “Almacenamiento” se usó “SSD de uso general” en lugar del tipo de almacenamiento predeterminado “IOPS aprovisionadas”.

Para asegurarse de que la base de datos sea accesible desde el exterior, hay que utilizar el panel “Configuración de conectividad adicional” cerca de la parte inferior de la página y habilitar el acceso público. Esto es importante ya que necesitamos conectarnos a la instancia con un shell de PostgreSQL para crear un usuario para que Hydra lo use. Y aunque no podemos crear el usuario específico de Hydra desde este asistente, podemos crear la base de datos que usaremos para Hydra. Esto se hace en el panel “Configuración adicional”. Llamaremos a nuestra base de datos inicial “hydra” por simplicidad.

Después de verificar todas las configuraciones, damos click en “Crear base de datos“ en la parte inferior de la página.

Sin embargo, antes de que podamos continuar, necesitaremos crear un usuario específico para Hydra. El usuario “postgres“ que hemos creado junto con la base de datos es un usuario “maestro“. Tiene todos los permisos y definitivamente no queremos que este usuario esté expuesto a ninguna aplicación.

En su lugar, usaremos este usuario maestro para iniciar sesión en nuestra instancia con un shell de PostgreSQL y crear otro usuario con menos permisos solo para Hydra.

Pero antes de conectarnos a la instancia, necesitamos abrir el puerto PostgreSQL en nuestro grupo de seguridad predeterminado. Si no hacemos esto, AWS no permitirá conexiones a nuestra instancia desde el exterior.

En el panel de RDS, la base de datos debería aparecer como activa. En la configuración de los Grupos de Seguridad de la base de datos en Inbound hay que asegurarse que el puerto TCP de PostgreSQL esté abierto para todas las direcciones IP.

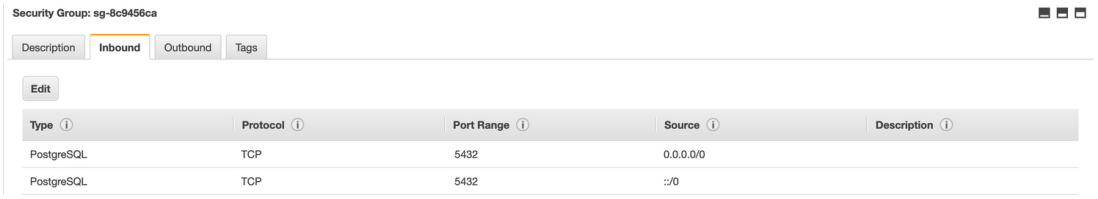


Fig. 4.13: Puerto TCP PostgreSQL

Para conectarnos a la base de datos utilizaremos DBeaver y el endpoint que se encuentra en la parte de abajo de los detalles de la base de datos, en el panel de RDS. Y nos conectamos utilizando el usuario maestro y la contraseña creados durante la inicialización de la base de datos.

Conectado a la base de datos, en Editor SQL se crea un nuevo Script en el que se escriben y ejecutan los comandos que se muestran en la imagen:

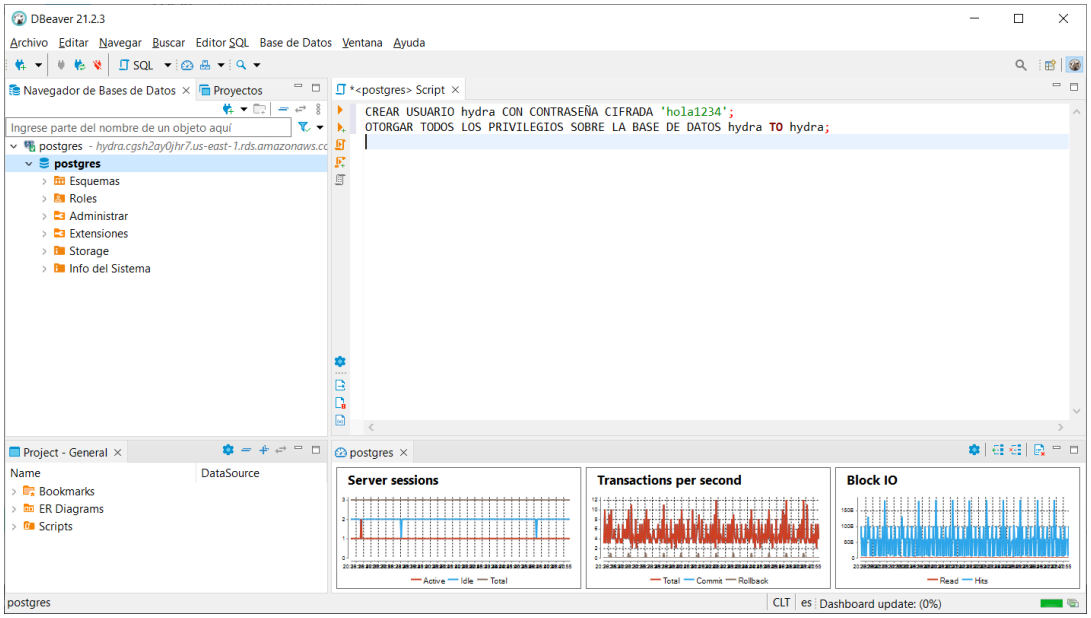


Fig. 4.14: Script SQL - DBeaver

Esto creará un usuario con el nombre hydra y le asignará la contraseña “hola1234” y le asignará todos los privilegios sobre la base de datos hydra, muy descriptivo.

Con esto, la base de datos PostgreSQL ya está lista para que Hydra la utilice.

4.2.4. Creación repositorios y pusheo de imágenes a ECR

Ahora es necesario crear los repositorios para alojar las imágenes tanto del backend como del storefront que se utilizarán para el proyecto, dichas imágenes son las que fueron desarrolladas con anterioridad y se encuentran almacenadas en imágenes de docker. Para ello dentro del servicio de ECS (Elastic Container Service) se busca la opción de Repositorios que se encuentra en Amazon ECR (Elastic Container Registry), para ambos casos se utilizó la configuración por defecto con respecto a la mutabilidad.

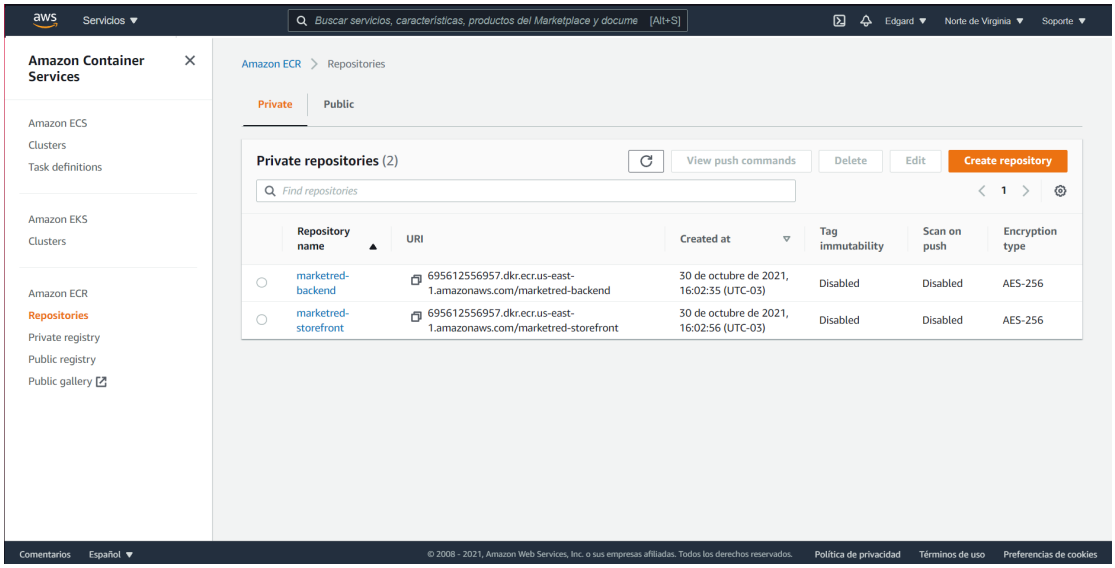


Fig. 4.15: Respositorios ECR

Ahora es necesario buildear y pushear las imágenes de Docker tanto para el backend como para el storefront. Para ello hay que tener instalada y configurada la herramienta de línea de comandos de AWS y en los mismos repositorios que acabamos de crear en el panel ECR de la consola de AWS, hay una opción para “Ver comandos push”, ya basta solamente copiar y pegar los comandos generados desde el modal.

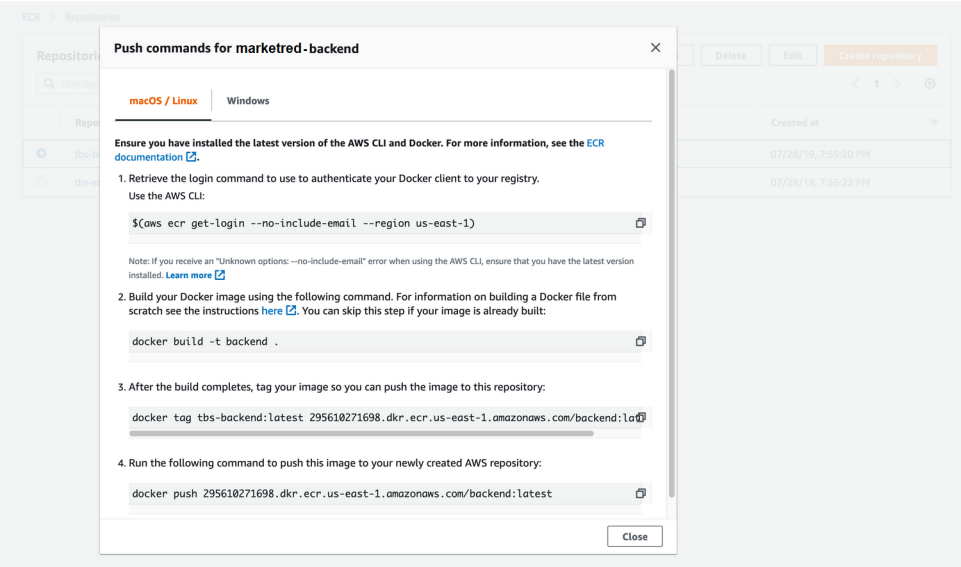


Fig. 4.16: Comandos Push

Siguiendo estos pasos ya tendremos las imágenes del backend y storefront en los contenedores de Amazon ECR que acabamos de crear.

4.2.5. Estableciendo los balanceadores de carga públicos.

Los balanceadores de carga son fundamentales para enrutar solicitudes entre varios contenedores del mismo servicio.

Una configuración básica de Reaction Commerce necesita un balanceador de carga para el backend, uno para el storefront y dos para Hydra (uno orientado a Internet para la API pública y otro interno para la API de administración). Cada balanceador de carga requerirá su propio dominio o subdominio para apuntarlo. Una configuración típica sería:

- Backend `reaction.dominio`, en este caso: `reaction.marketred.ml`
- API pública de Hydra en `auth.dominio`, en este caso sería auth.marketred.ml. La API de administración no necesita estar asociado con un dominio ya que es interna y no accesible desde Internet)
- Storefront en `marketred.ml`

Antes de crear los balanceadores de carga es necesario obtener el certificado SSL para los dominios y subdominios para los que apuntarán los balanceadores de carga. En este caso se hizo la certificación del dominio *.marketred.ml de forma manual, utilizando certbot, pero también hay un asistente en AWS que permite emitir certificados gratuitos a través de ACM (AWS Certificate Manager), con éste se hizo la certificación de los subdominios que se utilizarán para los distintos balanceadores de carga.

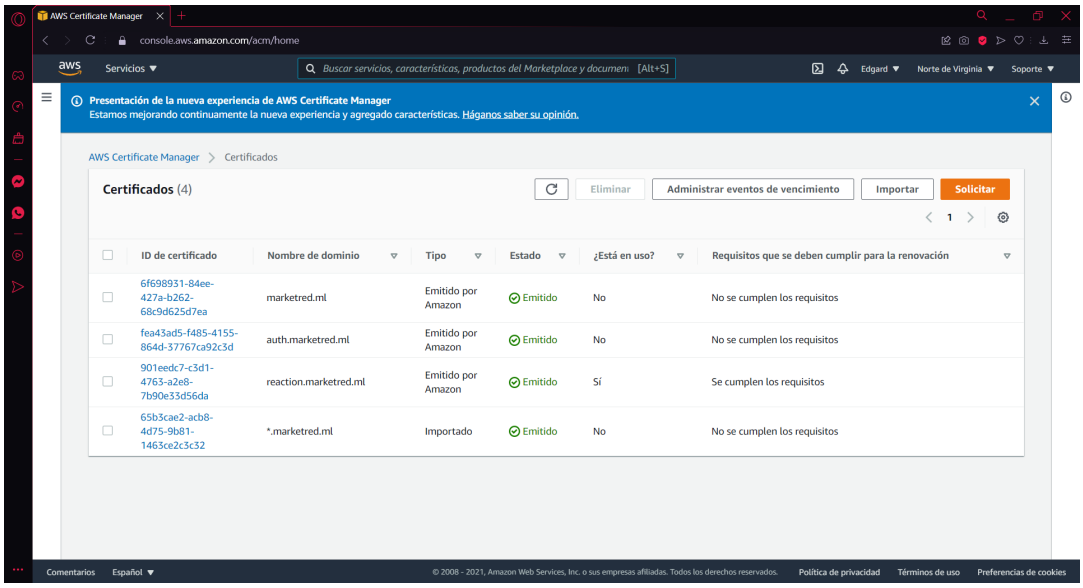


Fig. 4.17: Certificados SSL

Ahora, se crean los tres balanceadores de carga de acceso público con los que tendremos tres direcciones IP diferentes para apuntar a los registros A / CNAME del dominio y subdominios, además de eso se configura del balanceador de carga interno de la API de administración de Hydra. En cada caso en hay que establecer los puertos correspondientes:

- 3000 para el backend
- 4444 para la API pública de hydra, 4443 para la interna
- 4000 para el storefront

Cada uno además hay que asociarlo a su dominio correspondiente y al certificado/dominio de cada uno.

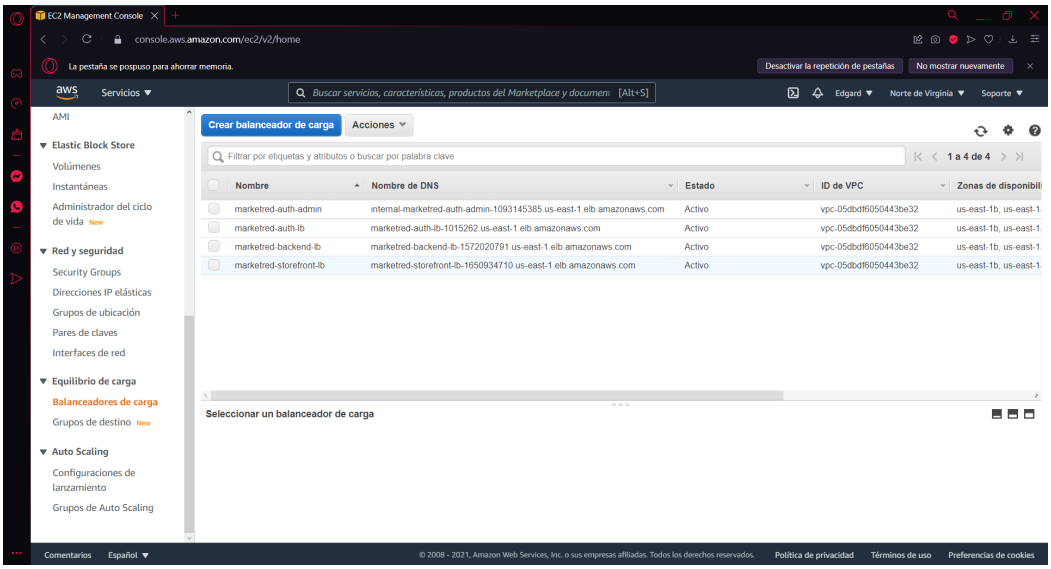


Fig. 4.18: Balanceadores de carga

4.2.6. *Creación de cluster*

Para alojar las imágenes del proyecto se crea un cluster con la configuración predeterminada de **EC2 Linux + Networking**.

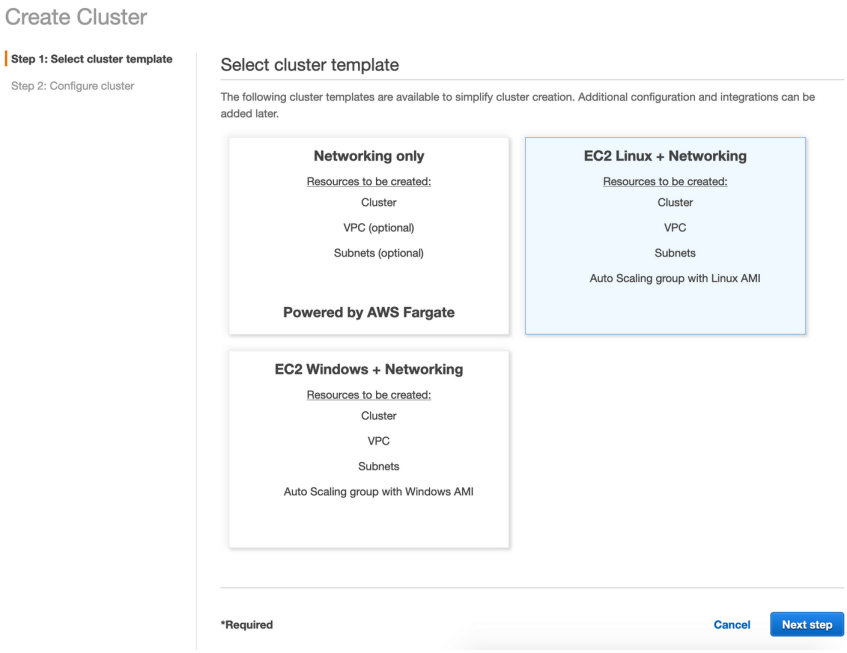


Fig. 4.19: Creación del cluster

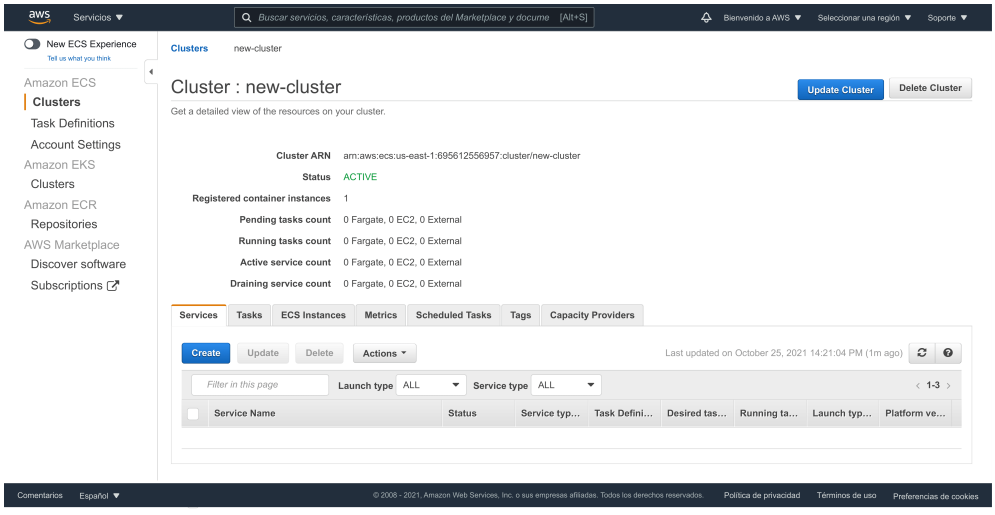


Fig. 4.20: Cluster creado

Con el cluster creado es necesario crear las tareas y servicios que éste va a alojar.

Implementación del backend

En primera instancia crearemos la tarea asociada al backend el cual es requisito para el funcionamiento de las demás.

Para esto tenemos que identificar el URI del repositorio Docker del backend que se encuentra alojado en el ECR en nuestro AWS. Este URI lo vamos a necesitar al momento de crear nuestra tarea.

Para crear la tarea nos vamos al dashboard de nuestro ECS, y en la parte de Task Definitions buscamos la opción “Create new Task Definition” y en el Launch Type Compatibility se selecciona EC2. Le damos un nombre a la Task Definition y en la parte de Container Definition agregamos uno nuevo, le ponemos un nombre y en la parte en la que debemos colocar la imagen, pegamos la URI que habíamos copiado de nuestro repositorio Docker del backend. Hay que configurar el puerto, en este caso el 3000 y luego definir las variables y configuraciones de entorno. Para esto es necesario haber creado un nuevo usuario en la base de datos que tenga permisos de lectura y escritura.

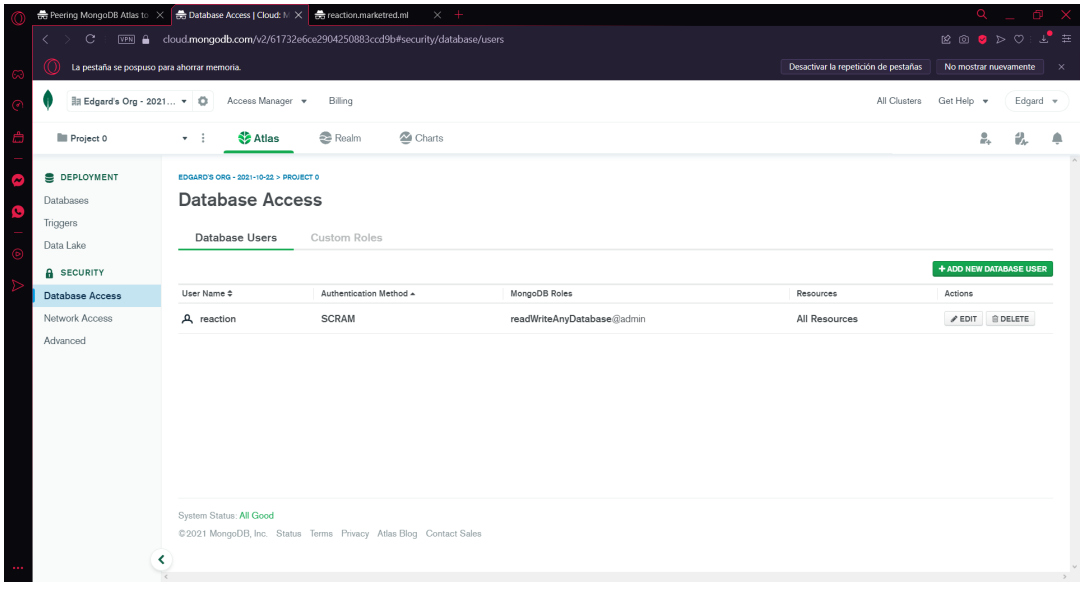


Fig. 4.21: Acceso a la base de datos

Los detalles de las configuraciones de entorno se pueden ver en la lectura asociada a la implementación del Backend en el curso antes mencionado. [33]

Por último, en “Almacenamiento y registro”, se habilita la opción para “Configurar automáticamente los registros de CloudWatch”. De esta forma, los registros de este contenedor estarán disponibles en CloudWatch, lo cual es bastante útil. Con eso listo, se agrega la definición de contenedor y luego en “Crear” en la parte inferior de la página para guardar la definición de tarea.

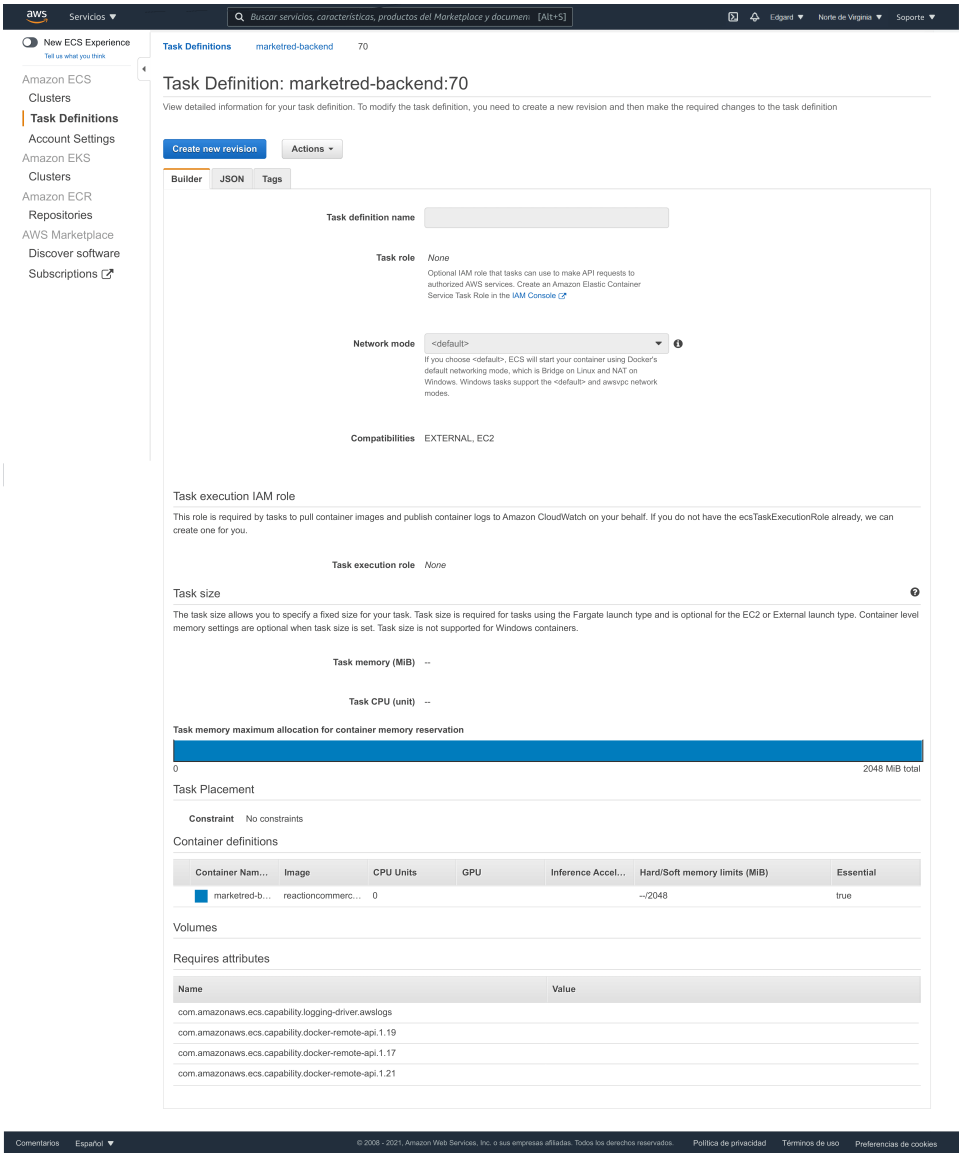


Fig. 4.22: Amazon ECS - backend

Posterior a esto se crea un servicio que utilice esta definición de tarea. En el que se selecciona EC2 como el tipo de lanzamiento, se puede usar el mismo nombre que se usó en la definición de la tarea y 1 es la cantidad de tareas a la que estará asociado el servicio.

Entre 30 segundos y 1 minuto sería más o menos recomendable como período de gracia de verificación de estado. Igual esto depende del tiempo que tarde en comenzar la compilación de producción. Si bien nuestra construcción de producción no toma más de 1 minuto, se puede colocar un poco más en el período de gracia por si acaso.

En la sección de Equilibrio de carga, se utiliza Application Load Balancer y se selecciona el balanceador de carga asociado al backend que se creó anteriormente.

Luego, en la sección "Contenedor para balancear la carga", se selecciona el grupo objetivo del backend existente. Todo lo demás se autocompletará.

Al final de la página se desmarca la "integración de descubrimiento de servicios", ya que usaremos los nombres DNS del balanceador de carga interno. Con esto podemos dar por finalizado el asistente para la creación del servicio del backend, el cual podemos ver ejecutándose en el cluster.

Antes de poder probar nuestro backend es necesario modificar la lista de seguridad de éste. En el panel de EC2 de la parte de Servicios. Nos dirigimos a "Grupos de seguridad".

En la lista de grupos de seguridad, debería tener uno llamado "EC2ContainerService-≪cluster-name≫", el cual hay que editar y agregarlas siguientes reglas:

- Puerto TCP 3000, accesible desde cualquier lugar
- Puerto TCP 4000, accesible desde cualquier lugar
- Puerto TCP 4444, accesible desde cualquier lugar

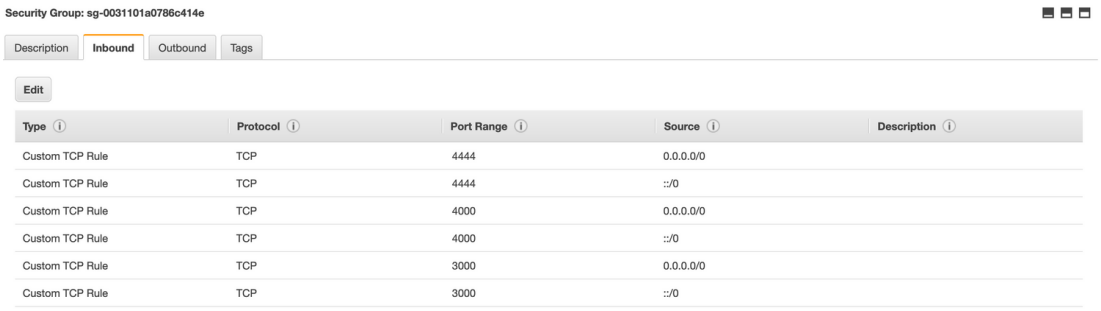


Fig. 4.23: Grupos de seguridad

Estos aseguran que las instancias de nuestro clúster sean accesibles en nuestros puertos de aplicación. Con el grupo de seguridad configurado, basta abrir el dominio asociado al balanceador de carga del backend.

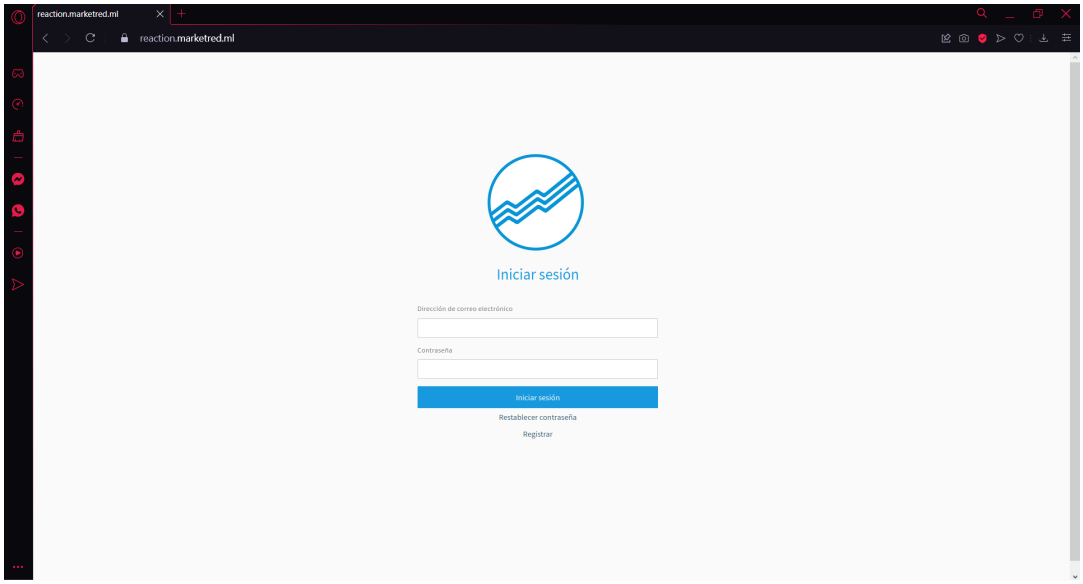


Fig. 4.24: Backend implementado en el dominio especificado

Implementación del storefront

La implementación del storefront será bastante similar a la del backend. En el panel de Task Definitions, creamos una nueva tarea. El tipo de lanzamiento, igual que para el backend, es EC2. Se le asigna un nombre a la tarea, y en la sección Contenedores agregamos un nuevo contenedor.

Ahora tenemos que identificar el URI del repositorio Docker del storefront que se encuentra alojado en el ECR en nuestro AWS. Este URI lo agregamos como la imagen de nuestro contenedor. Asignamos el puerto TCP 4000 tanto en el host como en el contenedor. Luego se establecen las variables y la configuración de entorno.

Una vez hecho esto, en la sección de Almacenamiento y registro se habilita la opción para "Configurar automáticamente los registros de CloudWatch" y finalmente damos click en Crear. Con esto el contenedor del storefront se encontrará creado.

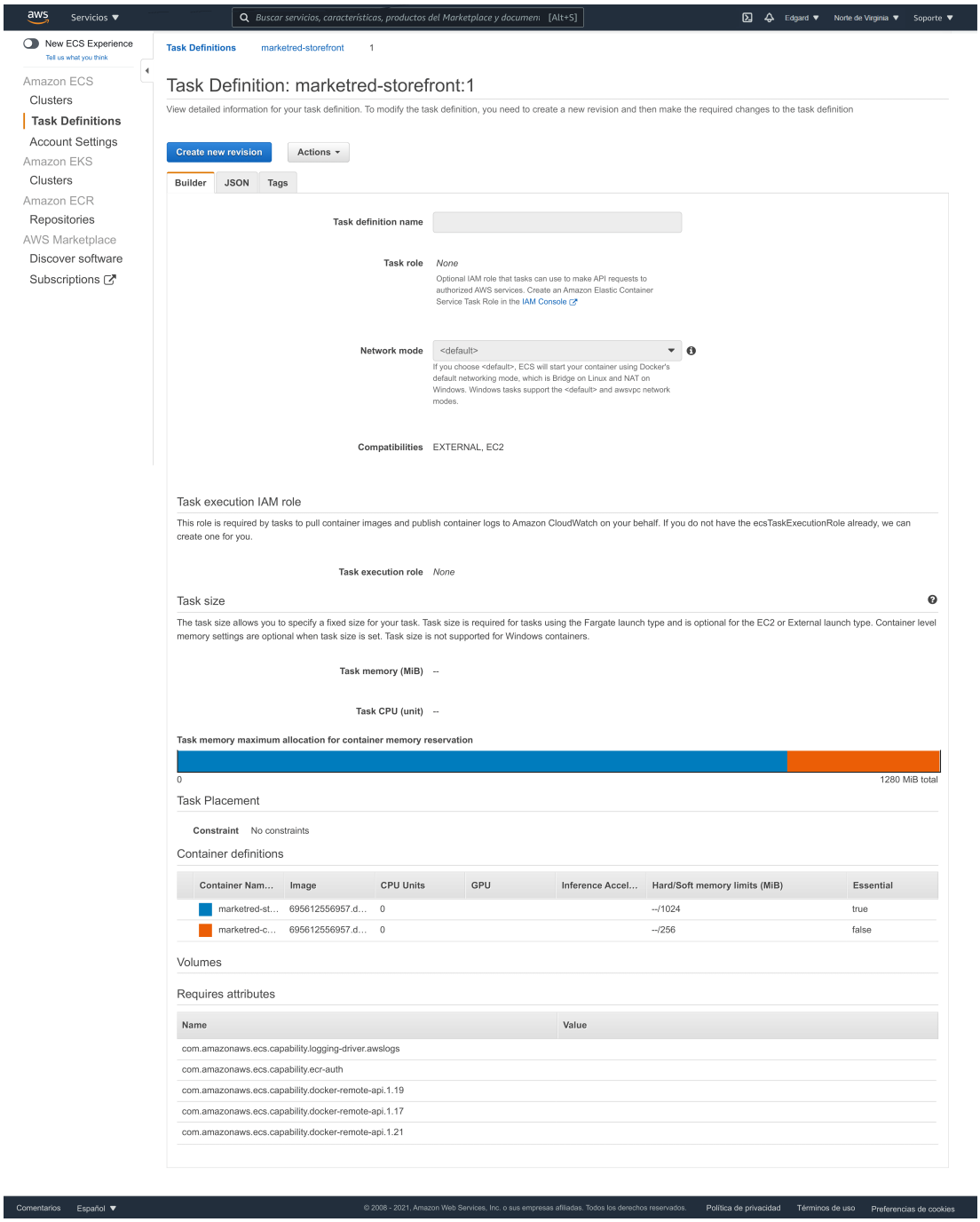


Fig. 4.25: Amazon ECS - storefront

Al igual que con el backend, en la parte superior de la página de definición de la tarea guardada en las Acciones se crea un Servicio para la nueva definición de tarea que se acaba de crear.

EC2 es el tipo de lanzamiento y 1 es el número de tareas al que estará asociado el servicio. Se utilizaron 30 segundos como período de gracia de verificación de estado.

Se selecciona “Application Load Balancer”, y luego el balanceador de carga asociado al storefront en el menú desplegable que dice “Nombre del balanceador de carga”.

En “Container to load balance”, se hace click en “Add to load balancer” y se elije el target group del balanceador de carga del storefront en el menú desplegable, el resto se completará automáticamente. Antes ir al siguiente paso hay que asegurarse de desmarcar la integración de descubrimiento de servicios, el cual utiliza las acciones de la API de AWS Cloud Map para administrar los espacios de nombres HTTP y DNS para los servicios de Amazon ECS y aquí se utilizaron los nombres DNS del balanceador de carga interno. Después de completar los requisitos de escalado automático, se crea el servicio.

Ahora tanto el servicio del storefront como el servicio de backend se están ejecutando en conjunto.

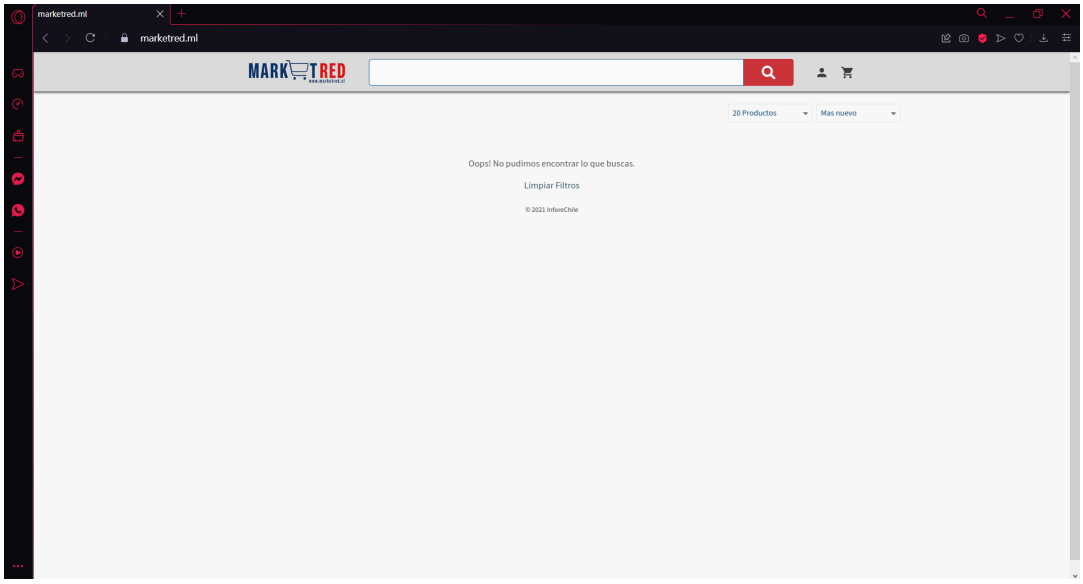


Fig. 4.26: Storefront implementado en el dominio especificado

Implementación de la API pública de Hydra

Ahora que el backend y la tienda funcionan, solo nos falta un componente clave del ecosistema Reaction: Hydra. A diferencia de todos los demás servicios que hemos creado hasta ahora, Hydra requerirá dos definiciones de tareas diferentes que se ejecuten en dos servicios diferentes. Eso es porque Hydra tiene una API pública (que debe estar expuesta a Internet) y una API privada (que debe permanecer dentro de nuestra red privada). Para separarlos, los estamos ejecutando en sus propios servicios y balanceadores de carga.

En este caso no se creó una imagen de Docker de producción para Hydra. En su lugar, usaremos la URL del repositorio de Docker Hub de Hydra.

En este caso hay que tener dos contenedores para la definición de tarea. Antes de que Hydra pueda iniciarse, es necesario ejecutar algunos scripts de migración con un comando especial. Entonces se creó un contenedor que fue el que se encargó de eso, y se configuró el contenedor principal de Hydra para que se inicie solo después de que el primero esté terminado.

Como es costumbre en la parte de Task Definitions, creamos una nueva tarea, EC2 el tipo de ejecución, le asignamos un nombre a la tarea y se agregó un nuevo contenedor.

Al contenedor le ponemos como nombre “hydra-migrate” y en la URL imagen, se colocó lo siguiente: *oryd / hydra: v1.0.0-beta.9-alpine*. ECS entiende que esta es una referencia a Docker Hub y extrae la imagen de allí.

En la sección de Entorno (Environment) se agrega el siguiente comando:
migrate,sql,postgres://hydra:<password>@<endpoint>:5432/hydra?sslmode=disable

Una vez hecho esto, en Almacenamiento y registro, se habilita la opción para “Configurar automáticamente los registros de CloudWatch” y se agrega el contenedor a la tarea.

Con nuestro contenedor “hydra-migrate” agregado a la definición de la tarea, todo lo que queda por hacer es crear nuestro contenedor Hydra principal.

En la definición de la tarea, se vuelve “Agregar contenedor”. Se nombró “hydra” y se configura la imagen como *“oryd / hydra: v1.0.0-beta.9-alpine”*.

Se asigna el puerto 4444 y se establecen y configura el entorno y sus variables. Con las variables de entorno configuradas, debemos decirle a ECS que inicie nuestro contenedor Hydra solo después de que se haya ejecutado hydra-migrate. Esto se hace en “Startup Dependency Ordering”. Se selecciona hydra-migrate, que es el contenedor anterior que habíamos agregado y se establece la condición en “SUCCESS”.

Y finalmente, se habilita para que AWS pueda “Configurar automáticamente los registros de CloudWatch” en Almacenamiento y registro.

Se agregua el contenedor Hydra y se guarda la definición de la tarea.

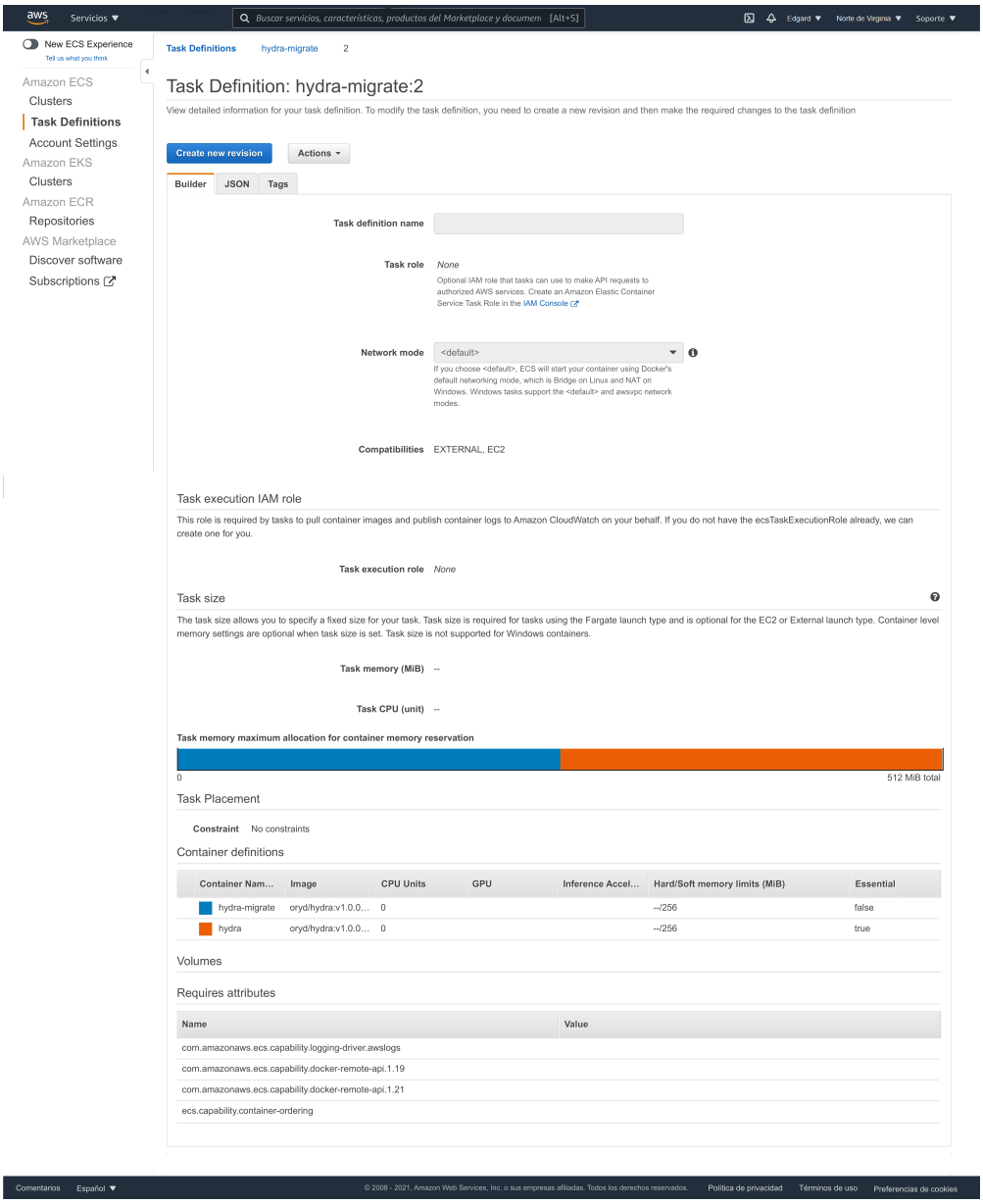


Fig. 4.27: Amazon ECS - hydra-migrate

Ahora hay que crear el servicio que ejecute la tarea. Se selecciona EC2 como el tipo de lanzamiento, se coloca 1 como el número de tareas y se avanza al siguiente paso.

El período de gracia de verificación de estado se estableció en 30 segundos, se seleccionó Application Load Balancer y se asignó el balanceador de carga de Hydra / auth creado anteriormente.

Se hace click en “Agregar al balanceador de carga” y se selecciona el grupo objetivo en el menú desplegable “Nombre del grupo objetivo”.

Como siempre, se desmarca la integración de descubrimiento de servicios, pasamos al siguiente paso, se configura el escalamiento automático si es necesario y se implementa el servicio.

Ahora la API pública de Hydra se encuentra en ejecución. El último paso es crear otra definición de tarea para la API de administración, y nuestro entorno estará completo.

Ahora se debe configurar la API privada de Hydra. Es muy similar a la API pública, solo que el comando del contenedor es diferente y estamos vinculando nuestro servicio a nuestro balanceador de carga interno.

En el panel de ECS, en Definiciones de tareas creamos una nueva tarea. Se le da un nombre a la tarea y nos vamos hasta la definición de contenedores. Agregamos uno nuevo al cual llamamos "hydra-admin" y usamos *"oryd/hydra: v1.0.0-beta.9-alpine"* como imagen (igual que la API pública). Y asignamos el puerto 4445 tanto en el host como en el contenedor.

Luego se establece el entorno y sus variables, activamos el CloudWatch logs, guardamos el contenedor y creamos la tarea.

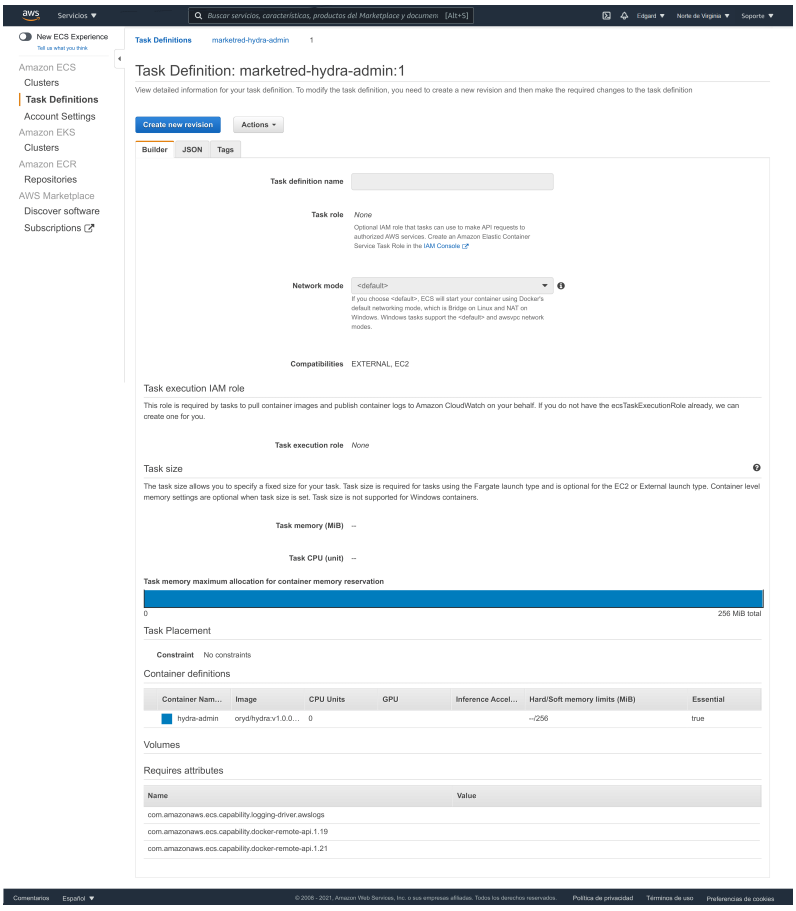


Fig. 4.28: Amazon ECS - hydra

Una vez que se crea la definición de la tarea, es necesario crear un servicio que se encargue con ella, esto lo hacemos utilizando el menú desplegable Acciones.

Al igual que para los otros servicios, utilizamos EC2, le asignamos un nombre y 1 será el número de tareas.

En el siguiente paso, se establece un período de gracia de verificación de estado de 30 segundos, se selecciona Application Load Balancer y asignamos el balanceador de carga interno Hydra creado anteriormente, además de agregar el target group correspondiente.

Como siempre, deshabilitamos la integración de descubrimiento de servicios, en el siguiente paso, se configura el escalamiento automático si es necesario y finalmente se implementa el servicio.

5. CONCLUSIONES

El desarrollo de la solución se encuentra prácticamente completo, así como se ha podido apreciar en el informe. Los principales hitos del proyecto han sido alcanzados y se encuentra registro de ello en el repositorio de MarketRed. Se definieron los distintos actores que estarán en interacción con el sitio web y se han implementado las formas en que interactúan. De manera interna, el sitio web se encuentra estructurado así como se había descrito en el esquema general, los componentes del prototipo, además de sus respectivas estructuras internas. Muestra de ello también, son las interfaces implementadas y funcionales que se pudieron observar en el informe, las cuales siguen lo que se había propuesto inicialmente en el diseño de éstas.

Como este proyecto en primera instancia estuvo pensado para ser implementado en la página web de Infored Chile, dentro del trabajo pendiente de la aplicación está precisamente la integración con dicho sitio web, situación que no se produjo por incumplimiento de parte de la contraparte con respecto a los tiempos. Más allá de eso, como la plataforma es escalable, modular e independiente se pudo realizar el levantamiento de ésta de forma externa haciendo uso de Cloud Computing, utilizando en este caso los servicios entregados por parte de AWS los cuales además permitieron agregarle robustez y mayor seguridad a la plataforma, aprovechando sus ventajas, agregando balanceadores de carga y administrando las bases de datos internamente en la red de AWS para así evitar el flujo de información a través de internet.

Siguiendo la linea de lo último, en lo que respecta a los riesgos, la mayoría han sido abordados, principalmente los que están relacionados con el tema de seguridad, esto por el uso del Identity Provider que permite gestionar el flujo de autenticación, además de lo que se mencionaba anteriormente de la base de datos.

Como trabajo futuro queda pendiente el testeo de la plataforma en un entorno más real, con tráfico real, para así poder establecer la cantidad de servidores necesarios para atender cierto flujo de visitas. De la misma forma esto permitiría verificar el correcto funcionamiento de los balanceadores de carga.

Junto con esto quedó pendiente también la utilización del método CI/CD (Continous Integration/Continous Deployment) que incorpora la automatización continua y el control permanente en todo el ciclo de vida de las aplicaciones, desde las etapas de integración y prueba hasta las de distribución e implementación, la cual es una práctica recomendada para maximizar el retorno del uso de metodologías ágiles, permitiendo realizar cambios en la aplicación en ”tiempo real”, mejorando así el uptime y reduciendo los tiempos de mantención de los sitios. La ausencia de este método se debe principalmente al modo de implementación que se realizó del prototipo, por utilizar las imágenes de docker.

Para utilizar dicho método hubiese sido necesario trabajar directamente con el código, crear una canalización con AWS CodePipeline, un servicio que compila, prueba e implementa el código cada vez que se produce un cambio en este. Utilizar una cuenta de GitHub, un bucket de Amazon Simple Storage Service (S3) o un repositorio de AWS CodeCommit, como la ubicación de origen del código de la aplicación de muestra. También se utilizaría AWS Elastic Beanstalk como destino de la implementación para la aplicación. La canalización permite detectar cambios realizados al repositorio de origen que contiene la aplicación de muestra y actualizar la que se encuentra activa.

Aparte de lo que respecta a la implementación, dentro de la aplicación quedó pendiente automatizar la función de pago a las MiPymes. El sistema puede generar informes de pago, con todas las transacciones aún no rendidas, para que estas sean pagadas, considerando el descuento de comisión, sin embargo, en estos momentos dicho proceso debe iniciarse de forma manual.

Bibliografía

[1] Comercio online se triplica, pero tiendas físicas, turismo y entretención extienden profunda crisis. (2020, 28 mayo). T13.cl. https://www.t13.cl/noticia/negocios/comercio-online-triplica-tiendas-fisicas-turismo-entretencion-crisis-28-05-20

[2] Ministerio de Economía, Fomento y Turismo: Quinta Encuesta Longitudinal de Empresas (ELE5) https://www.economia.gob.cl/2019/03/12/quinta-encuesta-longitudinal-de-empresas-ele5.htm/

[3] Montoya, A. Evolución del e-commerce en Chile. (2021, 17 mayo). Enrique Ortega Burgos. https://enriqueortegaburgos.com/evolucion-del-e-commerce-en-chile/

[4] Evolución del comercio electrónico: fases y futuro. (2021, 26 enero). Beetrack. https://www.beetrack.com/es/blog/evolucion-del-comercio-electronico

[5] eCommerce en Chile: crecimiento e influencia del Covid-19. (2020, 7 agosto). Expande Online. https://expandeonline.cl/blog/ecommerce-chile/

[6] MercadoLibre, Costos de vender un producto https://www.mercadolibre.cl/ayuda/Costos-de-vender-un-producto_870.

[7] ApañoTuPyme, Pagos — ApañoTuPyme: Manual de Inscripción https://unete.apanotupyme.cl/docs/pages/payments.html

[8] WooCommerce, WooCommerce - Sell Online With The eCommerce Platform for WordPress https://woocommerce.com/

[9] Lovers,WCFM Marketplace - Best Multivendor Marketplace for WooCommerce - Word-Press plugin
https://wordpress.org/plugins/wc-multivendor-marketplace/.

[10] Marketplace, The Grey Parrots, WC Marketplace - WordPress plugin
https://wordpress.org/plugins/dc-woocommerce-multi-vendor/

[11] Magento eCommerce Platforms — Best eCommerce Software for Selling Online — Magento
https://magento.com/es

[12] ebKul SoftWare Private Limited, Multi Vendor Marketplace
https://marketplace.magento.com/webkul-module-marketplace.html

[13] PrestaShop, Crear Tienda Online Gratis con Prestashop — Venta Online
https://www.prestashop.com/es

[14] WebKul Software Private Limited, Advanced Multi Vendor Marketplace
https://addons.prestashop.com/es/creacion-marketplace/8057-advanced-multi-vendor-marketplace.html

[15] OpenCart, OpenCart - Open Source Shopping Cart Solution
https://www.opencart.com/

[16] OpenCart, Multi-Store - OpenCart Documentation
http://docs.opencart.com/en-gb/administration/multi-store/

[17] PurpleTree Software LLP, No. 1 BestSeller Multi Vendor Marketplace Opencart Extension —50 % OFF
https://www.purpletreesoftware.com/multi-vendor-marketplace-opencart.html

[18] Spark Solutions Sp. z o.o., Spree Commerce - an ecommerce platform with over 1 million downloads
https://spreecommerce.org/

[19] GitHub, Spree Multi Vendor marketplace extension
https://github.com/spree-contrib/spree_multi_vendor

[20] Reaction Commerce, Reaction Commerce — Connecting the world through commerce
https://reactioncommerce.com/

[21] Docs Reaction Commerce, Shops Reaction Docs
https://docs.reactioncommerce.com/docs/concepts-shops.

[22] Docs Reaction Commerce, Understanding API Plugins · Reaction Docs
https://docs.reactioncommerce.com/docs/core-plugins-intro.

[23] Saleor, Saleor – A headless, GraphQL-first, open-source e-commerce platform
https://saleor.io/

[24] Oscar, Oscar - Domain-driven e-commerce for Django
http://oscarcommerce.com/

[25] Django-Oscar, Partner - django-oscar 2.0 documentation
https://django-oscar.readthedocs.io/en/2.0.4/ref/apps/partner.html

[26] Django-Oscar, Oscar - django-oscar 2.0 documentation
https://django-oscar.readthedocs.io/en/2.0.4/

[27] M. Análisis comparativo entre plataformas Cloud Computing — Grupo Fractalia. (2014, 17 junio) Fractalia.
https://fractaliasystems.com/analisis-comparativo-entre-plataformas-cloud-computing/

[28] Bardají, E. Comparativa Cloud: Amazon Web Services (AWS), Microsoft Azure y Google Cloud. (2021, abril). Esed.
https://www.esedsl.com/blog/comparativa-cloud-amazon-web-services-aws-microsoft-azure-y-google-cloud

[29] API de Khipu para crear cobro. (2021, 12 junio). Khipu. https://www.khipu.com/api/

[30] InforedChile - Buscador de servicios públicos y de pequeñas empresas de Chile. InforedChile. https://inforedchile.cl

[31] Laux, L. Deploying Reaction Commerce 2 on AWS ECS. out:grow university. https://university.outgrow.io/p/deploying-reaction-commerce-on-aws-ecs

[32] Laux, L. Deploying Reaction Commerce 3 on AWS ECS. out:grow university. https://university.outgrow.io/p/deploying-reaction-commerce-3-on-aws-ecs

[33] Deploying the backend. out:grow university. https://university.outgrow.io/courses/622670/lectures/11153960