



TECNOLÓGICO NACIONAL DE  
MÉXICO EN CELAYA



## INFORME TÉCNICO DE RESIDENCIAS PROFESIONALES

Mezfer Insider: Renovación digital

13 de julio de 2024 – 13 de diciembre de 2024

IBT Innova Business Technology

Presenta:

Cristhian Alberto Ortega Hernández

20030191

Ingeniería en Sistemas Computacionales

Asesor Externo

Jorge Octavio Luna Medrano

Asesor Interno

Claudia Mayela Alcaraz Avendaño

Celaya Guanajuato, a 13 de enero de 2025

# Índice

	Pág.
<b>1 Introducción</b>	<b>1</b>
<b>2 Justificación</b>	<b>1</b>
<b>3 Objetivos</b>	<b>2</b>
3.1 Objetivo General . . . . .	2
3.2 Objetivos específicos . . . . .	2
<b>4 Problemas a resolver</b>	<b>2</b>
<b>5 Marco Teórico</b>	<b>3</b>
5.1 Desarrollo Web . . . . .	3
5.2 Frameworks . . . . .	4
5.2.1 Frameworks para Frontend . . . . .	4
5.2.1.1 Next.js . . . . .	5
5.2.1.2 React.js . . . . .	5
5.2.1.3 Tailwind CSS . . . . .	5
5.2.2 Frameworks para Backend . . . . .	6
5.2.2.1 Node.js . . . . .	6
5.2.2.2 Express.js . . . . .	6
5.3 Servicios en la nube . . . . .	7
5.3.1 Beneficios de los servicios en la nube . . . . .	7
5.3.2 Tipos de servicios en la nube . . . . .	8
5.3.2.1 Software como servicio (SaaS) . . . . .	8
5.3.2.2 Plataforma como servicio (PaaS) . . . . .	8
5.3.2.3 Infraestructura como servicio . . . . .	8
5.4 Proveedores de servicios en la nube . . . . .	9
5.4.1 Amazon Web Services (AWS) . . . . .	9
5.4.1.1 Servicios de AWS . . . . .	9
5.5 Metodología para la gestión de proyectos . . . . .	11
5.5.1 Metodología Kanban . . . . .	11

<b>6</b>	<b>Procedimiento y descripción de las actividades realizadas</b>	<b>11</b>
6.1	Conociendo Mezfer Rewards . . . . .	12
6.1.1	Aplicación web de Mezfer Rewards . . . . .	12
6.1.2	Aplicación móvil de Mezfer Rewards . . . . .	13
6.2	Análisis de la base de datos de Mezfer Rewards . . . . .	14
6.2.1	Creación de la base de datos de Mezfer Insider . . . . .	14
6.3	Selección de tecnologías a utilizar . . . . .	15
6.3.1	Servicios en la nube . . . . .	15
6.3.2	Tecnologías de frontend . . . . .	15
6.3.3	Tecnologías de backend . . . . .	16
6.4	Inicialización y despliegue de los proyectos de Frontend y Backend . . . . .	16
6.4.1	Inicialización del proyecto de Frontend . . . . .	16
6.4.2	Despliegue del proyecto de Frontend en servidor de producción . . . . .	17
6.4.3	Inicialización del proyecto de Backend . . . . .	19
6.4.4	Despliegue del proyecto de Backend en servidor de producción . . . . .	21
6.5	Creación de balanceadores de carga y asignación de los dominios . . . . .	22
6.6	Integración de Mezfer Insider con Control Manager Siso . . . . .	24
6.6.1	Creación de endpoints en la API de Mezfer Insider . . . . .	24
6.6.2	Creación de la empresa, vinculación del usuario e ingreso . . . . .	25
6.7	Creación de conexiones a las bases de datos . . . . .	26
6.7.1	Instalación de librería . . . . .	27
6.7.2	Configuración de conexiones . . . . .	27
6.8	Inicio de sesión para Mezfer Insider . . . . .	28
6.8.1	Inicio de sesión de administradores . . . . .	28
6.8.2	Inicio de sesión de clientes . . . . .	29
6.8.2.1	Endpoint de inicio de sesión . . . . .	29
6.8.2.2	Formulario de inicio de sesión . . . . .	29
6.9	Dashboard del cliente . . . . .	30
6.9.1	Endpoints del backend para el dashboard . . . . .	30
6.9.1.1	Endpoint para obtener las campañas activas . . . . .	31
6.9.1.2	Endpoint para obtener la información del usuario . . . . .	31
6.9.2	Interfaz del Dashboard . . . . .	31

6.10 Sección “Mis campañas” . . . . .	32
6.10.1 Endpoints del backend de la sección “Mis campañas” . . . . .	32
6.10.1.1 Endpoint para obtener las campañas activas . . . . .	32
6.10.1.2 Endpoint para obtener las campañas en las que se encuentra inscrito el usuario . . . . .	33
6.10.2 Interfaz de la sección “Mis campañas” . . . . .	33
6.11 Detalles de las campañas . . . . .	34
6.11.1 Endpoints del backend de la sección “Detalles de las campañas” . . . .	34
6.11.1.1 Endpoint para obtener la información de la campaña . . . . .	34
6.11.1.2 Endpoint para obtener las campañas en las que se encuentra inscrito el usuario . . . . .	35
6.11.1.3 Endpoint para obtener los premios de las campañas . . . . .	35
6.11.1.4 Endpoint para obtener los productos participantes de la campaña . . . . .	35
6.11.1.5 Endpoint para obtener las solicitudes de puntos de un usuario	36
6.11.1.6 Endpoint para obtener los canjes del usuario . . . . .	36
6.11.1.7 Endpoint para obtener los puntos del usuario . . . . .	37
6.11.1.8 Endpoint para realizar una solicitud de puntos . . . . .	37
6.11.2 Interfaz de la sección “Detalles de las campañas” . . . . .	41
6.11.2.1 Encabezado de la sección “Detalles de las campañas” . . . .	41
6.11.2.2 Contenido de la sección “Detalles de la campaña” . . . . .	42
6.12 Cambios generales en el Frontend de los clientes . . . . .	50
6.13 Dashboard de los administradores . . . . .	50
6.13.1 Endpoints del Backend del Dashboard de administradores . . . . .	50
6.13.1.1 Endpoint para obtener las ventas que se han realizado con el tiempo . . . . .	50
6.13.1.2 Endpoint para obtener los usuarios que se han registrado con el tiempo . . . . .	51
6.13.1.3 Endpoint para obtener el Top 10 de productos . . . . .	51
6.13.1.4 Endpoint para obtener el Top 10 de distribuidores . . . . .	51
6.13.1.5 Endpoint para obtener una comparativa de ventas entre el año actual y el año anterior . . . . .	52

6.13.2 Interfaz del Dashboard de administradores . . . . .	52
6.13.2.1 Card con los ingresos de la empresa . . . . .	52
6.13.2.2 Card con los tops de la empresa . . . . .	55
6.13.2.3 Card con los usuarios que se han registrado a Mezfer Rewards	56
6.14 Cambios generales en el Frontend de los administradores . . . . .	57
6.15 Prueba . . . . .	58

# 1. Introducción

En este informe se documenta el desarrollo y los resultados obtenidos durante el proyecto de residencias profesionales, el cual permitió aplicar y consolidar los conocimientos adquiridos en el ámbito académico, enfrentando problemas reales del entorno profesional y ofreciendo soluciones prácticas y efectivas.

El contenido de este informe abarca desde los fundamentos que justifican el proyecto hasta los resultados y conclusiones finales. Se describen los objetivos específicos, las problemáticas detectadas y el procedimiento que se siguió para resolverlas. Además, se incluyen los resultados obtenidos mediante planos, prototipos, gráficas u otros elementos que permitan mostrar el alcance del trabajo realizado. Por último, se destacan las competencias desarrolladas y se presentan recomendaciones para el futuro del proyecto.

Este trabajo representa una valiosa oportunidad para aplicar habilidades técnicas y profesionales, proporcionando un aprendizaje significativo que contribuye al crecimiento profesional y la solución de problemas.

## 2. Justificación

Mezfer es una empresa mexicana dedicada a solucionar los problemas del campo mediante la elaboración de productos dirigidos al sector agroalimentario. El creciente número de clientes llevó a Mezfer a crear “Mezfer Rewards”, un programa de recompensas mediante el cual los clientes se inscriben a los diferentes ‘Programas de lealtad’ para canjear puntos que reciben con la compra de productos.

Actualmente, Mezfer Rewards es gestionado mediante 2 aplicaciones diferentes:

- Una aplicación web para que los administradores puedan realizar la gestión del programa.
- Una aplicación móvil para que los clientes puedan realizar todo el proceso para obtener recompensas.

Ahora, Mezfer busca que tanto clientes como administradores puedan realizar sus respecti-

vas actividades en una misma plataforma para así mantener un mejor control. Con este objetivo establecido, comienza el desarrollo de “Mezfer Insider”, un portal para administradores y clientes que unifica las funcionalidades de las aplicaciones anteriores en un solo lugar.

## **3. Objetivos**

### **3.1 Objetivo General**

- Desarrollar e implementar la aplicación “Mezfer Insider” para centralizar y controlar las actividades de administradores y clientes.

### **3.2 Objetivos específicos**

- Analizar la base de datos anterior para identificar áreas de mejora.
- Diseñar la nueva base de datos para Mezfer Insider.
- Integrar Mezfer Insider a Control Manager Siso (CMS) para que los administradores puedan iniciar sesión.
- Crear el inicio de sesión para los clientes de Mezfer.
- Crear el dashboard para los clientes.
- Integrar la funcionalidad de canje de puntos para los clientes.

## **4. Problemas a resolver**

- Fragmentación de plataformas: Actualmente, el programa Mezfer Rewards es gestionado mediante dos aplicaciones separadas: una aplicación web para los administradores y una aplicación móvil para los clientes. Esta separación implica un manejo dividido e ineficiente de las funcionalidades.
- Descentralización: El hecho de que las actividades estén dispersas en diferentes plataformas dificulta la capacidad de mantener una visión centralizada y controlada de las operaciones.
- Falta de integración: Las actividades de los clientes y los administradores están divididas en distintos sistemas, lo que hace que la gestión de tareas sea poco eficiente.
- Falta de adaptación de funcionalidades: Se requiere unificar las funcionalidades de

las aplicaciones web y móvil en una misma plataforma. Es importante que estas funcionalidades mantengan las características clave que ofrecen, como la gestión del programa y el acceso a las recompensas.

- Mejora de la experiencia del usuario: El usuario es una pieza fundamental en esta nueva plataforma, por lo que es importante asegurar que tanto administradores como clientes puedan realizar sus actividades eficientemente y sin problemas para ofrecer una experiencia agradable.

## **5. Marco Teórico**

### **5.1 Desarrollo Web**

El desarrollo web es el proceso de crear y mantener sitios web y abarca una gran variedad de acciones que van desde la creación de códigos y diseños, hasta la administración de contenidos y servidores (Ortega, 2022). Para el desarrollo web se utilizan diferentes lenguajes de programación que ayudan a crear el código que hace que las páginas funcionen.

Entre los lenguajes de programación más comunes para el desarrollo web se encuentran:

- HyperText Markup Language (HTML): Este lenguaje es el componente más básico de la web, y ayuda a definir el significado y la estructura del contenido web. (MDN Web Docs, s.f.)
- Cascading Style Sheet (CSS): Este lenguaje de estilos es utilizado para describir la presentación de documentos HTML o XML. (MDN Web Docs, s.f.)
- Hypertext Preprocessor (PHP): Es un lenguaje de programación del lado del servidor que puede integrarse en HTML para crear aplicaciones y sitios web dinámicos. (MDN Web Docs, s.f.)
- JavaScript: Es un lenguaje de programación que se ejecuta del lado del cliente de la web y es utilizado para programar cómo se comportan las páginas web cuando ocurre un evento. También puede ser ejecutado de lado del servidor, lo que permite que se puedan generar páginas web dinámicas. (MDN Web Docs, s.f.)



El desarrollo web se puede dividir en tres categorías: Frontend, Backend y Full Stack (TripleTen, s.f.).

El desarrollo Frontend es el responsable de la parte del cliente, es decir, lo que el usuario ve en pantalla cuando ingresa al sitio.

El desarrollo Backend se encarga de aspectos como servidores, bases de datos y lenguajes de programación. En esta parte se procesan las solicitudes del Frontend que contienen los datos para la base de datos u otros sistemas.

El desarrollo Full Stack es la combinación del Frontend y Backend, por lo que esta categoría es responsable tanto de la parte del cliente como de la parte del servidor.

## **5.2 Frameworks**

En el mundo del desarrollo (y en muchas otras áreas) existen marcos de trabajo o frameworks que ofrecen una estructura base para elaborar un proyecto con objetivos específicos; es una especie de plantilla que sirve como punto de partida. El uso de un framework puede ayudar a realizar un proyecto en menos tiempo gracias a la reutilización de herramientas y módulos, y en la programación, también ayuda a tener un código más limpio y consistente. (Unir FP, 2022)

Existen una gran diversidad de frameworks que son utilizados tanto para el desarrollo Frontend como el desarrollo Backend.

### **5.2.1 Frameworks para Frontend**

Estos frameworks proporcionan una base para el desarrollo de la interfaz de usuario de las aplicaciones o páginas web. Hay una gran diversidad de frameworks frontend disponibles, lo que permite que se puedan crear soluciones de muy diferentes escalas y objetivos (Valencia, 2023).

A continuación, se mencionan algunos de los que han sido utilizados para este proyecto:

### **5.2.1.1 Next.js**

Next.js es un framework JavaScript ligero y de código abierto creado sobre React que permite construir sitios y aplicaciones web rápidos y sencillos de usar. Se utilizan los componentes de React para construir la interfaz de usuario mientras que Next.js se encarga de algunas características adicionales y optimizaciones (Vercel, s.f.).

Next.js también abstrae y configura automáticamente las herramientas que se requieren para React, como agrupación, compilación, entre otras cosas. Esto permite que los desarrolladores se enfoquen más en el desarrollo de la aplicación en lugar de gastar tiempo en configuraciones.

### **5.2.1.2 React.js**

Para algunas personas, React es un framework pero en realidad, de acuerdo al sitio oficial (<https://es.react.dev/>), es una librería de JavaScript. Aunque no es un framework, se describirá en esta sección por ser una herramienta de gran importancia para el proyecto.

React.js es una librería de JavaScript de código abierto desarrollada por Facebook cuyo objetivo es simplificar el proceso de construir interfaces de usuario interactivas (Herbert, 2022). Permite desarrollar las aplicaciones con la creación de componentes reutilizables; estos componentes son piezas individuales de una interfaz final.

El rol principal de React es encargarse de la capa de vista de una aplicación proveyendo la mejor y más eficiente ejecución de renderizado. Además, motiva a los desarrolladores a separar las interfaces en componentes individuales y reutilizables, de esta manera, React combina la velocidad y eficiencia de JavaScript con un método más eficiente de manipulación del DOM para renderizar las páginas web de manera más rápida.

### **5.2.1.3 Tailwind CSS**

Tailwind CSS es un framework de CSS de utilidad que permite a los desarrolladores aplicar estilos directamente en el HTML utilizando clases predefinidas (Axarnet, s.f.).

A diferencia de otros frameworks, Tailwind CSS proporciona una colección de clases CSS de bajo nivel que se pueden combinar para construir cualquier diseño sin tener que escribir hojas de estilo personalizadas.

Este es un framework altamente personalizable. Aunque viene con una configuración predeterminada, se puede sobrescribir modificando su archivo de configuración, el cual permite la personalización de paletas de colores, estilizado, espaciado, temas, entre otros. Otra ventaja importante es que este framework optimiza el CSS del proyecto gracias a su herramienta PurgeCSS, la cual se encarga de analizar el HTML y eliminar las clases que no se utilizan, lo cual ayuda a reducir el tamaño de los archivos.

## **5.2.2 Frameworks para Backend**

Estos frameworks permiten simplificar algunos aspectos del proceso de desarrollo web, haciéndolo más fácil y rápido; ayudan al desarrollador a construir la arquitectura de su sitio web (García, 2022).

A continuación, se mencionan algunos de los que han sido utilizados para este proyecto:

### **5.2.2.1 Node.js**

Node.js no es un framework, si no un entorno de ejecución para JavaScript, A pesar de esto, se describirá en esta sección debido a su importancia para el desarrollo del proyecto.

Node.js es un entorno de ejecución de JavaScript de código abierto y multiplataforma que permite a los desarrolladores crear servidores, aplicaciones web, herramientas para la línea de comandos y scripts. (OpenJS Foundation, s.f.)

Está basado en el motor de código abierto V8 de Google, el cual se actualiza constantemente y ofrece una gran rapidez, es por ello que se recomienda para el desarrollo de aplicaciones en tiempo real, las cuales se caracterizan por proporcionar información de manera instantánea.

### **5.2.2.2 Express.js**

Express.js es un framework de backend para Node.js que proporciona características y herramientas robustas para desarrollar aplicaciones de backend escalables (OpenJS Foundation, s.f.).

Este framework proporciona un conjunto de herramientas para aplicaciones web, peticiones y respuestas HTTP, enrutamiento y middleware para construir y desplegar aplicaciones a

gran escala. Es utilizado para una gran variedad de cosas en el ecosistema JavaScript/Node.js; se pueden desarrollar aplicaciones, endpoints de APIs, sistemas de enrutamiento y frameworks (Kinsta, 2022)

Express.js es un framework no dogmático, es decir, un framework que no impone demasiadas restricciones sobre la mejor manera de unir componentes para alcanzar un objetivo. Es por esto que permite insertar casi cualquier middleware compatible dentro de la cadena del manejo de la petición, en cualquier orden; además, permite definir una estructura de directorios propia, por lo que se pueden crear tantas carpetas y archivos como se desee (MDN Web Docs, s.f.).

### **5.3 Servicios en la nube**

Los servicios en la nube son recursos para aplicaciones e infraestructura que existen en internet, y permiten a los clientes aprovechar recursos de computación potentes sin la necesidad de adquirir o mantener hardware o software. (Hewlett Packard Enterprise, s.f.)

Cuando se utilizan los servicios en la nube, se puede delegar la gestión de la infraestructura y centrarse en utilizarla. El proveedor que se elija podrá proporcionar una amplia gama de actividades que mantendrán un negocio en funcionamiento. Cuando se emplean estos servicios, los usuarios autorizados podrán comunicarse, colaborar y gestionar proyectos, así como analizar, procesar, compartir y almacenar datos sin la necesidad de que alguien más tenga que supervisar, mantener o realizar copias de seguridad de la actividad.

#### **5.3.1 Beneficios de los servicios en la nube**

Los servicios en la nube ofrecen una gran variedad de beneficios para las empresas, entre los cuales se encuentran:

- Escalabilidad: Proporcionan la capacidad para escalar los recursos vertical y horizontalmente de forma instantánea en respuesta a la demanda.
- Flexibilidad: Se puede acceder a recursos informáticos y modificarlos con más agilidad, aumentando su capacidad para responder al uso de las unidades de negocio, los cambios en las circunstancias del mercado y las demandas de los clientes.
- Rentabilidad: Los mecanismos de precios con pago por consumo y la ausencia de

inversiones iniciales en hardware e infraestructura mejoran la eficiencia de los costes de capital.

- Seguridad: Para ofrecer protección frente amenazas e infracciones, los proveedores de nube emplean fuertes características de seguridad como cifrado, límites al acceso y supervisión.

### **5.3.2 Tipos de servicios en la nube**

Existen diferentes tipos de servicios que proporciona la nube, y en cada caso, los proveedores mantienen la infraestructura de la nube subyacente (Hewlett Packard Enterprise, s.f.).

#### **5.3.2.1 Software como servicio (SaaS)**

En este servicio, los proveedores a los suscriptores el uso de su software ejecutándose sobre la infraestructura de la nube, lo que significa que la aplicación permite una amplia distribución y acceso. El proveedor se ocupa de cuestiones como la gestión y el control de la red, los servidores, sistemas operativos, almacenamiento, virtualización, datos, middleware e incluso capacidades de aplicaciones individuales. Las aplicaciones de SaaS se suelen diseñar para resultar fáciles de usar por un público más amplio.

#### **5.3.2.2 Plataforma como servicio (PaaS)**

Con PaaS, los usuarios tienen más control que con SaaS, porque obtienen acceso a un marco que empieza en el sistema operativo. Permite a los usuarios ubicar sus propias aplicaciones en la infraestructura de la nube con lenguajes de programación, bibliotecas, servicios y herramientas que admita el proveedor. El suscriptor dispone de control sobre las aplicaciones implementadas, los datos y, posiblemente, los ajustes de configuración del entorno de hospedaje. Sin embargo, la gestión y control de la red, los servidores, los sistemas operativos y el almacenamiento recaen en el proveedor.

#### **5.3.2.3 Infraestructura como servicio**

Con IaaS, los suscriptores pueden diseñar un entorno completo configurando una red virtual separada de otras redes. Los usuarios ejecutan un sistema operativo y aprovisionan el procesamiento, el almacenamiento, las redes y otros recursos informáticos fundamentales para ejecutar software en la infraestructura de la nube. IaaS también proporciona a los suscriptores un control limitado de determinados de la red, y en ocasiones, también

ofrecerán servicios como supervisión, automatización, seguridad, equilibrio de carga y resiliencia del almacenamiento.

## **5.4 Proveedores de servicios en la nube**

Un proveedor de servicios en la nube (CSP) es una empresa de Ti que proporciona recursos de computación bajo demanda y escalables, como potencia de computación, almacenamiento de datos o aplicaciones por internet (Google Cloud, s.f.).

El mercado de los proveedores de servicios de comunicaciones incluye una gran variedad de proveedores de servicios en la nube. Los que se consideran líderes consolidados son Google Cloud, Microsoft Azure y Amazon Web Services (AWS), sin embargo, hay muchas otras empresas pequeñas que también ofrecen servicios en la nube como IBM, Alibaba, Oracle, Red Hat, DigitalOcean y Rackspace.

Para el desarrollo de este proyecto se utilizó Amazon Web Services, por lo que, a continuación, se hablará únicamente de esta plataforma y los servicios utilizados.

### **5.4.1 Amazon Web Services (AWS)**

AWS es un proveedor de servicios en la nube que ofrece una gran variedad de servicios y características que van desde tecnologías de infraestructura como cómputo, almacenamiento y bases de datos hasta tecnologías emergentes como aprendizaje automático e inteligencia artificial, lagos de datos y análisis e internet de las cosas. Esto hace que llevar las aplicaciones existentes a la nube sea más rápido, fácil y rentable y permite crear casi cualquier cosa que se pueda imaginar. (Amazon Web Services, Inc., s.f.)

La infraestructura de AWS está basada en centros de datos distribuidos en todo el mundo, lo que permite a los usuarios implementar sus aplicaciones y servicios en ubicaciones geográficas específicas según sus necesidades. Además, posee una amplia gama de herramientas de gestión y automatización para ayudar a los usuarios a administrar y controlar sus recursos en la nube de manera eficiente.

#### **5.4.1.1 Servicios de AWS**

A continuación, se describen algunos de los servicios de AWS que se utilizaron para el desarrollo del proyecto:

- Amazon Elastic Compute Cloud (Amazon EC2): Amazon EC2 proporciona capacidad de computación escalable bajo demanda en la nube de AWS. El uso de este servicio reduce los costos de hardware para que se pueda desarrollar e implementar aplicaciones con mayor rapidez. Permite lanzar cuantos servidores virtuales se necesiten, configurar la seguridad y las redes, y administrar el almacenamiento.
- Amazon Relational Database Service (Amazon RDS): Es un servicio de base de datos relacional fácil de administrar, optimizada para el costo total de propiedad. Es fácil de configurar, operar y escalar según la demanda. Automatiza las tareas de administración de bases de datos como el aprovisionamiento, la configuración, las copias de seguridad y la aplicación de revisiones. Permite a los clientes crear una nueva base de datos en cuestión de minutos y ofrece flexibilidad para personalizar las bases de datos a fin de satisfacer las necesidades en ocho motores.
- Amazon Simple Storage Service (Amazon S3): Es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Almacena datos como objetos dentro de buckets. Un objeto es un archivo y cualquier metadato que describa ese archivo. Un bucket es un contenedor de objetos. Todos los clientes pueden usar este servicio para almacenar y proteger cualquier cantidad de datos para diversos casos de uso como lagos de datos, sitios web, aplicaciones móviles, dispositivos IoT, entre muchas otras cosas.
- Elastic Load Balancing: Es un servicio que distribuye automáticamente el tráfico entrante entre varios destinos, como instancias EC2, contenedores y direcciones IP, en una o varias zonas de disponibilidad. Supervisa el estado de los destinos registrados y dirige el tráfico únicamente a los destinos en buen estado. Este servicio escala automáticamente la capacidad del balanceador de carga en respuesta a los cambios en el tráfico entrante.
- Amazon Route 53: Es un servicio web de sistema de nombres de dominio (DNS) escalable y de alta disponibilidad. Se puede utilizar Route 53 para realizar tres funciones principales en cualquier combinación: registro de dominios, enrutamiento DNS y comprobación de estado.

## **5.5 Metodología para la gestión de proyectos**

Las metodologías para la gestión de proyectos son conjuntos de principios, prácticas y procedimientos que son utilizados para planificar, ejecutar y supervisar un proyecto (APD, 2024). Existen una gran variedad de metodologías, por lo que se debe analizar adecuadamente cada una de estas para saber cual es la que se podría adaptar mejor al proyecto.

A continuación, se explica la metodología que se utilizó para el desarrollo de este proyecto.

### **5.5.1 Metodología Kanban**

La metodología Kanban es un sistema de gestión de tareas que forma parte de las metodologías ágiles. Su propósito principal es supervisar y optimizar el flujo de trabajo desde el inicio hasta la finalización de las tareas, asegurando un proceso continuo y eficiente.

El primer paso para utilizar Kanban es crear un tablero visual que sea accesible para todo el equipo, donde se representen las diferentes fases del flujo de trabajo. El tablero debe tener columnas que indiquen el estado de las tareas, desde que inician hasta que terminan. Cada tarea es representada mediante tarjetas que se mueven a través del tablero a medida que avanza por las diferentes etapas del proceso.

La metodología Kanban es ideal para empresas que requieren flexibilidad y eficiencia en la gestión de sus proyectos. Al proporcionar una visión clara del flujo de trabajo y facilitar la priorización de tareas, Kanban no solo mejora la productividad, sino que también asegura un seguimiento adecuado y una capacidad de respuesta rápida ante cualquier cambio o desafío que surja.

## **6. Procedimiento y descripción de las actividades realizadas**

Las actividades que en esta sección son descritas están redactadas en orden cronológico y con base en el cronograma de actividades. Es importante destacar que el cronograma de



actividades fue elaborado de acuerdo al tablero de actividades Kanban que se realizó para dar seguimiento a todas las actividades que se han realizado para el desarrollo del proyecto.

## **6.1 Conociendo Mezfer Rewards**

Mezfer Rewards es un programa de recompensas creado por la compañía Mezfer para poder entregar a sus clientes diferentes premios mediante el canje de puntos que obtienen gracias a la compra de productos. Para poder llevar a cabo este programa, Mezfer decidió crear una aplicación web y una aplicación móvil.

Para poder comenzar con el nuevo proyecto “Mezfer Insider” fue de gran importancia conocer ambas aplicaciones con las que funciona el proyecto, pues estas sirven como antecedentes y guías para tener una idea más sólida sobre como se va a estructurar el nuevo proyecto y todas las funciones que se van a mantener o eliminar.

A continuación, se describen de manera general ambas aplicaciones.

### **6.1.1 Aplicación web de Mezfer Rewards**

La aplicación web fue desarrollada utilizando Laravel, un framework para el lenguaje de programación PHP. Esta aplicación es un panel de administración, por lo que sólo está pensada para ser utilizada por usuarios que sean administradores.

La aplicación está integrada a un sistema más grande llamado “Siso ERP”, un sistema integral de gestión empresarial que controla, facilita y optimiza los procesos operativos de las empresas. Por lo tanto, para poder acceder al panel de administración de Mezfer Rewards, es necesario tener una cuenta en Siso ERP.

Una vez que el administrador ha ingresado, podrá realizar todas las acciones necesarias para gestionar el programa. Entre las acciones más importantes se encuentran:

- **Gestión de usuarios:** El administrador es capaz de activar o desactivar la cuenta de un usuario, modificar su rol y suscribir o desuscribir al usuario de una campaña.
- **Gestión de campañas:** El administrador es capaz de modificar toda la información relevante de una campaña como su descripción, los productos participantes y los premios disponibles.

- Gestión de solicitudes de puntos: El administrador es capaz de aceptar o rechazar una solicitud de puntos que haya enviado un cliente, de acuerdo a si la evidencia que envió es válida o no.

Hay muchas más acciones que puede realizar el administrador y que complementan al programa de recompensas. Si bien son acciones que también son importantes, no es fundamental describirlas para poder comprender mejor la aplicación.

### **6.1.2 Aplicación móvil de Mezfer Rewards**

La aplicación móvil fue desarrollada con Flutter, un framework que permite desarrollar aplicaciones nativas para iOS y Android, y cuyo lenguaje de programación es Dart. Esta fue desarrollada específicamente para los clientes.

Al descargar e iniciar la aplicación, los clientes deberán ingresar su número telefónico y su contraseña para acceder; si no tienen cuenta, tendrán que registrarse.

Cuando el cliente ha iniciado sesión, accederá a la aplicación y podrá ver las campañas disponibles, así como otro contenido que podría ser de su interés. Es importante mencionar que la aplicación no cuenta con una función para inscribirse a la campaña que desee, pero esto no es una falla en el diseño, sino una decisión de la propia empresa; para que el cliente pueda inscribirse tendrá que comunicarse con un encargado de Mezfer y realizar la solicitud.

Si el cliente no está inscrito a ninguna campaña, sólo podrá revisar la información general como la descripción, los productos participantes y los premios disponibles; si el cliente está inscrito a alguna campaña, además de poder ver la información general, podrá realizar dos acciones más:

- Realizar solicitud de puntos: El cliente es capaz de enviar una solicitud de puntos, en donde especifica los productos que compró y la cantidad, así como la evidencia de la compra.
- Canjear puntos disponibles: El cliente es capaz de canjear los puntos que obtuvo con la compra de productos para poder obtener premios que sean de su interés.

Hay otras acciones que puede realizar el cliente y que complementan su experiencia en el

uso de la aplicación. Si bien son acciones que también son importantes, no es fundamental describirlas para poder comprender mejor la aplicación.

## **6.2 Análisis de la base de datos de Mezfer Rewards**

Para las aplicaciones anteriores de Mezfer Rewards se utilizó una base de datos relacional creada en PostgreSQL, un Sistema Gestor de Bases de Datos. Esta base de datos fue creada gracias a los servicios en la nube de Amazon Web Services (AWS), específicamente con su servicio “Amazon Relational Database Service” (Amazon RDS), un servicio que web que facilita la configuración, operación y escala de una base de datos relacional.

Esta base de datos contiene un total de 60 tablas, cada una con los atributos necesarios para almacenar la información relevante. El análisis que se realizó sirvió para identificar las tablas y los atributos que debían conservarse o que se debían eliminar, además de identificar áreas de mejora para la normalización de la base de datos.

### **6.2.1 Creación de la base de datos de Mezfer Insider**

Para Mezfer Insider se decidió crear una nueva base de datos en la que se aplicarían los cambios que se identificaron en la base de datos anterior. Estos cambios no se realizaron en la antigua base de datos porque no se quería afectar el funcionamiento de las aplicaciones, ya que éstas debían seguir disponibles para los usuarios en lo que se termina el desarrollo de Mezfer Insider; realizar estos cambios implicaría también actualizar el código de las aplicaciones para que siguieran funcionando correctamente, y esto quitaría tiempo de desarrollo para el nuevo proyecto.

Esta nueva base de datos también fue desarrollada como una base de datos relacional utilizando PostgreSQL, y está almacenada en la nube gracias al servicio RDS de Amazon Web Services.

Además de los cambios, una nueva funcionalidad que se quiso añadir a la base de datos de Mezfer Insider fueron las bitácoras. El propósito de las bitácoras es permitir llevar un registro sobre qué usuarios han realizados cambios en la información de ciertas tablas, de esta manera se podrá rastrear más fácilmente a cualquier usuario que haya realizado una modificación indebida y se podrá recuperar la información que haya sido cambiada.

Hasta el momento no se ha utilizado esta nueva base de datos pues no se ha iniciado el proceso de migración, pero próximamente será la única en utilizarse.

## **6.3 Selección de tecnologías a utilizar**

Como último paso para comenzar con el desarrollo de Mezfer Insider, se analizaron diferentes tecnologías para usar en frontend y backend.

### **6.3.1 Servicios en la nube**

La plataforma que se utilizó para proveer los servicios en la nube fue Amazon Web Services. De esta plataforma se utilizaron servicios como:

- Amazon Elastic Compute Cloud (EC2): Es un servicio que proporciona capacidad de computación escalable bajo demanda en la nube. Permite lanzar cuantos servidores virtuales sean necesarios. Dos instancias de EC2 fueron utilizadas para lanzar el proyecto de Mezfer Insider y la API.
- Amazon Relational Database Service (RDS): Es un servicio web que facilita la configuración, la operación y la escala de una base de datos relacional en la nube. Se utilizó este servicio para crear la base de datos de Mezfer Insider.
- Amazon Simple Storage Service (S3): Es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento líderes del sector. Este servicio fue utilizado para almacenar recursos importantes de Mezfer Insider, como archivos e imágenes.

### **6.3.2 Tecnologías de frontend**

Para facilitar el desarrollo del frontend, se decidió comprar una plantilla de panel de usuario llamada DashTail, la cual proporciona vistas ya diseñadas o componentes para poder ir construyendo poco a poco la vista para el usuario; esta plantilla ofrece una alta personalización, por lo que es posible realizar cualquier cambio a los componentes.

DashTail fue desarrollada utilizando Next.js y Tailwind CSS.

Next.js es un framework basado en JavaScript y React que permite crear aplicaciones web modernas, dinámicas y escalables. Aunque es un framework full-stack, en DashTail sólo se utiliza para frontend.

Tailwind CSS es un framework de CSS para el diseño de páginas web. Su principal característica es que no genera una serie de clases predefinidas para elementos, en su lugar, crea una lista de clases “de utilidad” que son utilizadas para dar estilos individuales a cada elemento. Es gracias a este framework que los componentes de la plantilla son altamente personalizables.

### **6.3.3 Tecnologías de backend**

El backend consistió en desarrollar una API para que el frontend pudiera comunicarse con esta y realizar peticiones HTTP para interactuar con la base de datos.

La principal tecnología de backend es Node.js, un entorno de ejecución utilizado para poder ejecutar código de JavaScript directamente en los servidores, es decir, sin la necesidad de un navegador web.

Junto a Node.js se utiliza Express.js, un framework de backend para Node.js que proporciona características y herramientas robustas para desarrollar aplicaciones de backend escalables. Sus herramientas como peticiones y respuestas HTTP, enrutamiento y middlewares resultaron muy útiles para el desarrollo de la API.

## **6.4 Inicialización y despliegue de los proyectos de Frontend y Backend**

Para comenzar con el desarrollo de Mezfer Insider fue necesario crear los proyectos e instalar las dependencias necesarias para que estos funcionaran correctamente.

### **6.4.1 Inicialización del proyecto de Frontend**

Como se mencionó anteriormente, para el Frontend se utilizó una plantilla llamada DashTail, la cual proporciona un conjunto de páginas ya diseñadas. Esta plantilla es un proyecto de Next.js, por lo que, en este caso, no fue necesario crear el proyecto desde cero.

Para poder ejecutar el proyecto es necesario instalar todas las dependencias necesarias con el gestor de dependencias que se prefiera. En este proyecto se decidió utilizar el gestor de dependencias Yarn.

Cuando se instala Node.js, el gestor de dependencias predeterminado es NPM (Node

Package Manager), por lo que si se desea utilizar Yarn será necesario realizar una instalación extra. En la página oficial de Yarn se explica de una manera breve y sencilla como se realizar la instalación; utilizando NPM se ejecuta el siguiente comando:

***npm install --global yarn***

De esta manera Yarn quedará instalado en todo el sistema y podrá ser utilizado en cualquier proyecto.

Ahora, lo único que queda es instalar las dependencias del proyecto. Para esto, se ejecuta el comando:

***yarn install***

Y con esto comenzará la instalación.

Una vez instaladas todas las dependencias, se ejecuta el comando:

***yarn dev***

Esto iniciará el entorno de desarrollo del proyecto, que será ejecutado en un servidor local y para acceder a él simplemente en un navegador se escribe la URL *http://localhost:3000* (El puerto 3000 es el puerto predeterminado en el cual se ejecuta el servidor local de Next.js, sin embargo, este puede ser cambiado al puerto que se desee).

#### **6.4.2 Despliegue del proyecto de Frontend en servidor de producción**

Para desplegar el proyecto en el servidor de producción se deben seguir casi los mismos pasos que con la inicialización, pero hay algunos pasos extra que deben realizarse.

Principalmente, deben de estar instalados Node.js y Yarn para poder instalar las dependencias y ejecutar el proyecto. El proceso de instalación es el mismo, no hay ninguna diferencia.

Otra herramienta importante que se debe instalar es PM2 (Process Manager 2), un gestor de procesos que permite mantener proyectos de Node.js en ejecución, es decir, si el servidor llegase a ser detenido o reiniciado, PM2 se encargará de iniciar el proyecto automáticamente

sin la necesidad de que un administrador ingrese al servidor a iniciar todos los proyectos nuevamente. Para instalar PM2, se ejecuta el siguiente comando:

***yarn global add pm2***

El siguiente paso es compilar el proyecto para que Next.js realice todas las optimizaciones necesarias y así quede preparado para producción. Para compilar el proyecto, desde la carpeta principal se ejecuta el siguiente comando:

***yarn build***

De esta manera Next.js comenzará a compilar y optimizar todas las páginas que tenga la aplicación. Al finalizar el proceso, se creará una carpeta llamada “.next”, la cual contiene el proyecto optimizado y listo para producción.

Ahora, hay 5 archivos importantes que deben ser enviados al servidor para poder ejecutar correctamente el proyecto:

- .next: Esta es la carpeta que contiene todo el proyecto optimizado.
- .env: Este es el archivo donde se almacenan las variables de entorno.
- package.json: Este es el archivo que contiene todas las dependencias que han sido instaladas en el proyecto.
- yarn.lock: Un archivo generado por Yarn que lleva un registro de las versiones exactas de las dependencias que requiere un proyecto para que funcione adecuadamente.
- next.config.js: Este es el archivo que contiene la configuración del servidor de Next.js.

Para enviar estos archivos al servidor, se ejecuta el siguiente comando:

***scp -i ~/ruta/llave/PEM -r archivos usuario@ip\_publica\_servidor:~/carpeta/destino***

Una vez estén todos los archivos en el servidor de producción, se instalan las dependencias como se explicó anteriormente, y finalmente, se ejecuta el proyecto con PM2 para que este se encargue de gestionarlo. Para ejecutar el proyecto con PM2, se utiliza el siguiente comando:

***pm2 start "yarn start" --name nombre-app***

Posteriormente, se guarda la configuración con el siguiente comando:

***pm2 save***

Para hacer que la aplicación se ejecute automáticamente cuando se reinicie el servidor, se utiliza el siguiente comando:

***pm2 startup***

Al ejecutar este comando PM2 proporcionará otro comando que solamente debe ser copiado, pegado y ejecutado, y se habrá terminado de configurar. Ahora, con cada reinicio del servidor, la aplicación se ejecutará automáticamente.

Cuando se quieran subir cambios al proyectos, simplemente se repiten los pasos:

- Compilar el proyecto.
- Subir los archivos que hayan sufrido cambios al servidor.
- Reiniciar el proceso de PM2.

Para reiniciar el proceso de PM2, se ejecuta uno de los siguientes comandos:

***pm2 reload nombre-app***

***pm2 restart nombre-app***

Y con esto quedarán aplicados los cambios que se hayan subido.

### **6.4.3 Inicialización del proyecto de Backend**

El proyecto de Backend consiste en una API realizada en el framework Express.js. Este proyecto sí es creado desde cero, por lo que se deben seguir unos cuantos pasos extra. Es importante mencionar que también debe estar ya instalado Node.js, al igual que un gestor de dependencias.

Lo primero que se tiene que hacer es crear una carpeta donde va a estar guardado todo



el proyecto. Para crear una carpeta, se ejecuta el siguiente comando (puede cambiar de acuerdo al Sistema Operativo que se use):

***mkdir nombre-carpeta***

Posteriormente, se ingresa a la carpeta y desde ahí se ejecuta el siguiente comando:

***npm init -y***

Lo que hace ese comando es iniciar la configuración del proyecto, al usuario se le hacen ciertas preguntas para saber que cosas quiere incluir o usar. La opción -y hace que a todas las preguntas se les responda con un “sí”, por lo que se puede omitir si se desea pensar un poco más en la configuración.

Como esta es una API realizada con Express.js, es necesario instalar el paquete. Para instalar Express.js, se ejecuta el siguiente comando:

***npm install express***

Ahora sólo falta crear el servidor de Express, ya que ahí es donde se realizarán las diferentes solicitudes HTTP desde el Frontend. Para lograr esto, se crea un archivo que preferentemente se llame “index.js”, aunque puede llevar cualquier nombre; en este archivo es donde se inicializa el servidor y se colocan las rutas y middlewares que vayan a ser ocupados. Para inicializar el servidor de Express, se añaden las siguientes líneas de código (Ver figura 1):

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Figura 1: Aplicación básica de Express.

Y para ejecutar la aplicación, se utiliza el siguiente comando:

***node index.js***

Con ese comando se ejecuta el servidor de manera local y se pueden realizar algunas solicitudes HTTP a la URL *http://localhost:3000*.

#### **6.4.4 Despliegue del proyecto de Backend en servidor de producción**

La estrategia que se siguió para desplegar la API en el servidor de producción fue utilizar repositorios de GitHub, ya que estos es posible clonarlos en cualquier parte y esto facilita y simplifica el proceso de despliegue.

Lo primero que se debe hacer es crear un repositorio de GitHub. Esto se puede hacer desde la línea de comandos o desde la página web, pero para más facilidad se recomienda realizarlo desde la página web. Una vez creado el repositorio, se proporcionan una serie de comandos que se deben ejecutar para inicializar el repositorio local y subir el código ya existente al repositorio remoto.

Una vez que el código se encuentre en el repositorio remoto sigue clonarlo en el servidor de producción. Realizar la clonación de un repositorio es muy sencillo, sólo es necesario ingresar al servidor y ubicarse en el directorio donde va a estar el proyecto, luego se ejecuta el siguiente comando:

***git clone ruta-del-repositorio-remoto***

Y con esto todas las carpetas y archivos del proyecto estarán en el servidor de producción. Después, se deben instalar todas las dependencias mediante el siguiente comando:

***npm install***

El último paso es hacer que se ejecute automáticamente la aplicación, y esto se logra gracias a PM2. Los comandos que se utilizan son prácticamente los mismos, sólo hay una ligera variación en el primer comando que se utiliza, puesto que ejecutar una aplicación de Express es diferente a ejecutar una aplicación de Next.js. El comando que se debe usar para esta aplicación es el siguiente:

***pm2 start "node index.js" --name nombre-app***

Los siguientes comandos a ejecutar son exactamente los mismos que se explicaron en el despliegue de la aplicación del Frontend.

Para actualizar la aplicación, simplemente se suben los cambios realizados en el código al repositorio remoto y se descargan en el servidor de producción mediante el siguiente comando:

***git pull***

Una vez descargados los cambios, se reinicia el proceso de PM2 y la aplicación ya estará en su última versión.

## **6.5 Creación de balanceadores de carga y asignación de los dominios**

El uso de un balanceador de carga es necesario para poder distribuir equitativamente el flujo de tráfico que se piensa gestionar. AWS ofrece un servicio para implementar los balanceadores de carga, y para este proyecto se implementó uno en el servidor donde se encuentra alojado (una instancia de EC2). Existen diferentes opciones disponibles para un balanceador de carga, pero el que se seleccionó fue el Balanceador de Carga de Aplicaciones (ALB).

El ALB es ideal para aplicaciones web, soporta HTTP/HTTPS y enruta solicitudes basadas en reglas de nivel de aplicación.

La configuración más básica de un ALB consiste en asignarle un nombre, el esquema bajo el cual va a trabajar (que puede ser expuesto a internet o interno, y en este caso se usará el expuesto a internet.) y el tipo de dirección IP (que puede ser IPv4, Dualstack o Dualstack sin IPv4 pública, y en este caso se usará IPv4).

Lo siguiente que se tiene que configurar es el mapeo de la red, donde lo más importante es seleccionar las zonas de disponibilidad del balanceador ya que, si una zona llegara a fallar, el balanceador se encargará de dirigir el tráfico a una zona que esté disponible.

El siguiente aspecto a configurar es el agente de escucha y el direccionamiento. En el agente de escucha se asigna el puerto por el cual se van a estar escuchando las peticiones de la aplicación. Como se mencionó anteriormente, en el servidor la aplicación está siendo ejecutada mediante PM2, lo que hace que las peticiones que se realicen sean mediante el protocolo HTTP y por eso esté escuchando en el puerto 80; se debe limitar a que servidores se les aceptarán las peticiones del puerto 80, y es por eso que se crea un grupo de destino, donde se configuran las instancias a las cuales sí se les recibirán las peticiones del puerto 80. En este grupo de destino sólo estará involucrada la instancia en la cual se encuentra montada la aplicación Mezfer Insider.

Posteriormente, se configura el grupo de seguridad del balanceador, donde se definen las reglas de entrada. Las reglas de entrada que se configuraron fueron para los puertos 80 (HTTP) y 443 (HTTPS) y todas las direcciones IPv4, esto para que pueda ser accesible desde todos los navegadores, sin restricción de acceso. Estos dos puertos se seleccionaron debido a que el balanceador de carga actúa como un intermediario que distribuye el tráfico a los servidores Backend.

Aunque los puertos están abiertos, la seguridad se puede reforzar con la implementación de cifrado con SSL/TLS para HTTPS, y esto se va a lograr utilizando un dominio (el dominio fue implementado por el dueño de la cuenta de AWS y sólo se proporcionó la dirección). Gracias a este cifrado se crea una capa de seguridad adicional conocido como Firewall de Aplicaciones Web (WAF) que ayuda a proteger contra ataques y se asegura que los servidores sólo acepten tráfico proveniente del balanceador.

Se ha creado el dominio, pero todavía hace falta asignarlo al balanceador para que ya cuente con la certificación SSL. En los agentes de escucha ya estaba creado uno para el puerto 443, a este se le va a añadir el grupo de destino al cual se le van a redireccionar las peticiones a este puerto y se le va a asignar el dominio que fue proporcionado (<https://insider.mezfer.com>).

Con estos pasos realizados, el balanceador de carga queda correctamente configurado. Al acceder al DNS del balanceador de carga, este redirige a Mezfer Insider e ingresar directamente el dominio en el navegador permite el acceso a la aplicación web.

## **6.6 Integración de Mezfer Insider con Control Manager Siso**

Control Manager Siso (CMS) es el sistema de inicio de sesión de Siso ERP, un sistema ERP que permite realizar la gestión de distintos aspectos de una empresa. Cuando un cliente compra una licencia de Siso ERP, se le otorgan un correo y contraseña que podrán usar para iniciar sesión mediante CMS y acceder a sus empresas.

Quienes son administradores de CMS tienen acceso al panel de administración, donde se les permite gestionar a los usuarios y empresas del sistema Siso ERP.

Para poder crear empresas (que son tratadas como aplicaciones web), CMS realiza peticiones a diferentes endpoints de la API de la nueva empresa y, dependiendo de la respuesta que se reciba, se registrará correctamente a la base de datos o no. Lo mismo aplica para cuando se quiere asignar un usuario a una empresa y cuando se quiere ingresar a la empresa.

### **6.6.1 Creación de endpoints en la API de Mezfer Insider**

En la API de Mezfer Insider deben existir tres endpoints que permitirán a CMS realizar las acciones necesarias para registrar el proyecto en su base de datos.

El primer endpoint consiste en que se procese la información de la empresa. Mediante el Frontend, el administrador ingresa la información de la empresa en el formulario correspondiente y esta es enviada a la API; dependiendo de las necesidades del proyecto, esta información puede ser procesada o solamente recibida. En el caso de Mezfer Insider, esta información no requiere ser almacenada ni requiere ser procesada. Se debe retornar una respuesta con el código de estado de respuesta HTTP 202, indicando así que la petición se ha recibido, pero no se ha actuado.

El segundo endpoint consiste en vincular a un usuario con la empresa para indicar al sistema que ese usuario va a poder acceder a la empresa. En CMS, el administrador se encargará de asignarle la empresa al usuario, lo que enviará una nueva petición a la API para permitir la vinculación. La API recibirá el usuario y la empresa que están siendo vinculados, esta información será procesada de acuerdo a las necesidades del proyecto. La API enviará un código de estado de respuesta HTTP 200, indicando que la operación ha sido realizada con

éxito.

El tercer endpoint permitirá al usuario acceder a la empresa. Este endpoint es un poco más complejo, pues requiere el procesamiento de tokens JWT. Cuando el usuario ingresa en la empresa, CMS envía en la URL un token JWT con cierta información del usuario. Para Mezfer Insider, esta información es importante pues se considera que todos los usuarios que ingresen mediante CMS son administradores, así que debe de quedar registro de ellos. Cuando se recupera el token de la URL, este es decodificado y se lee cierta información que posteriormente es utilizada para realizar otra petición a la API de CMS y así obtener la información completa del usuario; con esto se crea un objeto con la información que se requiere del usuario para guardarla en la base de datos. Una vez guardado el usuario, se procede a crear un token JWT propio de Mezfer Insider que posteriormente será almacenado en una cookie, de esta manera se creará una cookie de sesión que será de utilidad para permitir al usuario realizar distintas acciones dentro de la aplicación. Este proceso se repite cada que un usuario ingresa a la empresa, por lo que debe validarse la existencia del usuario en la base de datos para evitar que se intente guardar nuevamente.

### 6.6.2 Creación de la empresa, vinculación del usuario e ingreso

Con los endpoints creados, ya es posible realizar todas las acciones necesarias para integrar Mezfer Insider con CMS.

Lo primero que se tiene que hacer es crear la empresa, y para eso se llena un formulario con todos los datos de la empresa (Ver Figura 2).

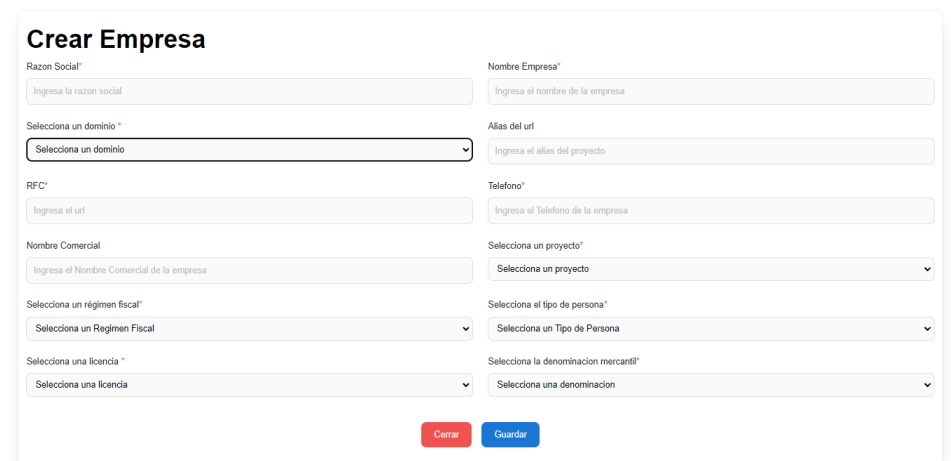
El formulario, titulado "Crear Empresa", está dividido en dos columnas. La columna izquierda contiene los campos: "Razon Social" (con un placeholder "Ingresa la razon social"), "Selecciona un dominio" (un menú desplegable con "Selecciona un dominio"), "RFC" (con un placeholder "Ingresa el rfc"), "Nombre Comercial" (con un placeholder "Ingresa el Nombre Comercial de la empresa"), "Selecciona un régimen fiscal" (un menú desplegable con "Selecciona un Regimen Fiscal") y "Selecciona una licencia" (un menú desplegable con "Selecciona una licencia"). La columna derecha contiene: "Nombre Empresa" (con un placeholder "Ingresa el nombre de la empresa"), "Alias del url" (con un placeholder "Ingresa el alias del proyecto"), "Telefono" (con un placeholder "Ingresa el Telefono de la empresa"), "Selecciona un proyecto" (un menú desplegable con "Selecciona un proyecto"), "Selecciona el tipo de persona" (un menú desplegable con "Selecciona un Tipo de Persona") y "Selecciona la denominación mercantil" (un menú desplegable con "Selecciona una denominación"). En la parte inferior del formulario hay dos botones: "Cerrar" (rojo) y "Guardar" (azul).

Figura 2: Formulario para la creación de empresa.

Posteriormente, el usuario debe ser vinculado con la empresa (Ver Figura 3).

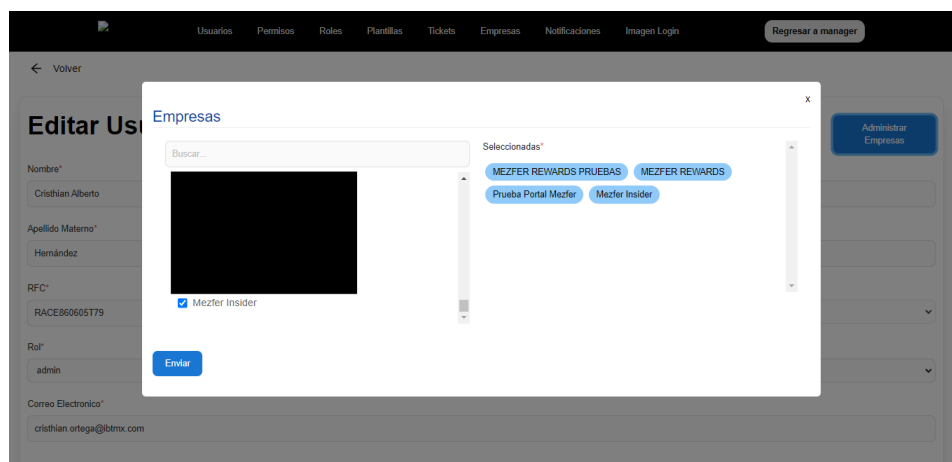


Figura 3: Vinculación de usuario con empresa.

Y con esto realizado, el usuario ya podrá tener acceso a la empresa (Ver Figura 4).

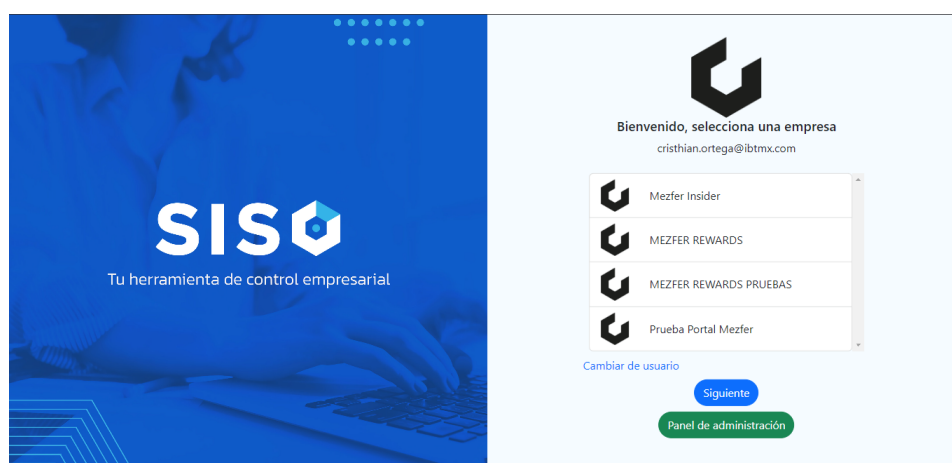


Figura 4: Selección de empresa.

## 6.7 Creación de conexiones a las bases de datos

Para poder empezar a codificar todas las acciones que se podrán realizar en Mezfer Insider, es necesario que en el backend estén realizadas las conexiones a las bases de datos para poder recuperar y guardar información.

Se realizaron dos conexiones puesto que, hasta el momento de la redacción de este reporte, Mezfer Insider ocupaba tanto la antigua base de datos como la nueva.

### 6.7.1 Instalación de librería

Para poder realizar las conexiones a las bases de datos es necesario instalar una librería que proporcione los módulos del Sistema Gestor de Bases de Datos.

La librería que se seleccionó para este proyecto fue Knex.js, un constructor de queries de código abierto para JavaScript. Knex.js tiene soporte para varios SGBD, incluido PostgreSQL, que es el que se utiliza.

Para instalar Knex.js, se ejecuta el siguiente comando dentro de la carpeta del proyecto:

***npm install knex***

### 6.7.2 Configuración de conexiones

Con Knex.js instalado, sólo falta configurar las conexiones.

Para realizar las configuraciones, se debe crear un archivo que lleve de preferencia el nombre “knexfile.js”. En este archivo, se crea un objeto con las siguientes propiedades:

- alias: El nombre que se le dará a la conexión.
- client: El SGBD que se utilizará para realizar la conexión.
- connection: En esta propiedad se debe configurar lo siguiente:
  - host: El servidor donde está almacenada la base de datos.
  - port: El puerto donde se está ejecutando el SGBD.
  - database: El nombre de la base de datos.
  - user: El nombre del usuario dueño de la base de datos.
  - password: La contraseña para acceder a la base de datos.
- pool: En esta propiedad se debe configurar lo siguiente:
  - min: El número mínimo de conexiones que se pueden realizar.
  - max: El número máximo de conexiones que se pueden realizar.
  - afterCreate: Una función que se ejecuta una vez realizada la conexión. Esta es opcional

La estructura del objeto queda de la siguiente manera (Ver Figura 5):



```

export default {
  development: {
    client: config.database.testing.dbms,
    connection: {
      host: config.database.testing.host,
      port: config.database.testing.port,
      database: config.database.testing.name,
      user: config.database.testing.user,
      password: config.database.testing.password,
      ssl: {
        rejectUnauthorized: false,
      },
    },
    pool: {
      min: 0,
      max: 10,
      afterCreate: function (conn, done) {
        conn.query("SELECT version()", function (err) {
          if (err) {
            done(err, conn);
          } else {
            console.log("Conexión exitosa a la base de datos");
            done(null, conn);
          }
        });
      },
    },
    migrations: {
      tableName: "knex_migrations",
    },
  },
};

```

Figura 5: Estructura del objeto de configuración de la conexión.

Por último, se crea una instancia de Knex y se le asigna la configuración que se desea utilizar, de esta manera se creará la conexión y se podrán realizar los queries que se requieran.

## 6.8 Inicio de sesión para Mezfer Insider

El ingreso a Mezfer Insider se puede hacer de dos maneras: como cliente o como administrador.

En esta sección se explicará cómo se desarrollo el inicio de sesión de ambos roles.

### 6.8.1 Inicio de sesión de administradores

Anteriormente se explicó una actividad donde se integraba Mezfer Insider a CMS. Esta integración, además de permitir la creación de la empresa en el sistema, también permite el inicio de sesión de los administradores pues todas las personas que ingresen a Mezfer Insider desde CMS son consideradas con ese rol.

Explicar como se realizó el inicio de sesión de administradores sería redundante, pues esto

ya se explicó anteriormente. Para conocer este proceso, puede leer la sección 6.6 de este reporte.

## **6.8.2 Inicio de sesión de clientes**

Para la creación del inicio de sesión de clientes se realizaron actividades en el Frontend y en el Backend.

### **6.8.2.1 Endpoint de inicio de sesión**

El primer paso para la creación del endpoint del Backend es construir la query que permitirá buscar al cliente en la base de datos. Esta query es una búsqueda en la tabla de clientes que recibe como parámetro ya sea el correo o el teléfono, y una vez encontrado, retorna toda la información.

Con la información del cliente recibida, se recupera la contraseña almacenada en la base de datos y se compara con la que se recibió desde el Frontend. Para realizar la comparación, es necesario descriptar la contraseña de la base de datos, de esta manera se podrá saber si lo que ingresó el cliente es lo mismo que está en la base de datos.

Si las credenciales del cliente son correctas, se crea un token JWT con información útil del cliente y se almacena en una cookie de sesión.

Con este proceso realizado, el cliente podrá ingresar a Mezfer Insider.

### **6.8.2.2 Formulario de inicio de sesión**

Para que un cliente pueda iniciar sesión necesita de un formulario donde pueda ingresar sus datos como correo y contraseña.

El formulario que se creó está conformado por dos campos: en el primer campo se le solicita al cliente que ingrese su correo electrónico o su número de teléfono y en el segundo campo se le solicita su contraseña (Ver Figura 6).

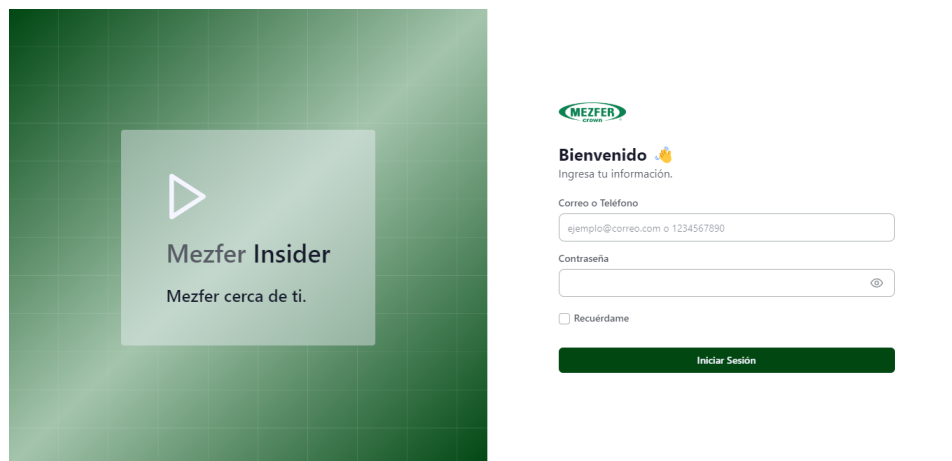
The image shows a login interface for 'Mezfer Insider'. On the left, there is a green square with a white play button icon and the text 'Mezfer Insider' and 'Mezfer cerca de ti.'. On the right, there is a login form. At the top right is the 'MEZFER' logo. Below it, the text 'Bienvenido' is followed by a smiley face emoji and 'Ingresa tu información.'. There are two input fields: 'Correo o Teléfono' with the placeholder text 'ejemplo@correo.com o 1234567890' and 'Contraseña' with a password icon. Below the password field is a checkbox labeled 'Recuérdame'. At the bottom right is a green button labeled 'Iniciar Sesión'.

Figura 6: Formulario de inicio de sesión de clientes.

El campo de correo electrónico o teléfono está validado mediante expresiones regulares, de esta manera si el cliente escribe algo que no tenga la estructura de alguna de esas dos opciones, no podrá avanzar en el envío de la información. Además, esta validación también permite que el sistema detecte que opción de inicio de sesión se va a usar. Si la validación se cumple para correo electrónico, el objeto que se crea con la información de inicio de sesión del usuario sólo enviará correo y contraseña, y en el Backend se realizará la búsqueda del cliente mediante correo; si la validación se cumple para teléfono, el objeto sólo enviará teléfono y contraseña y se realizará la búsqueda del cliente mediante teléfono.

El cliente no podrá iniciar sesión hasta que ambos campos tengan información y esa información tenga el formato esperado.

## 6.9 Dashboard del cliente

Hasta el momento de la redacción de este reporte, el dashboard del cliente sólo muestra un saludo personalizado y las campañas activas de Mezfer.

### 6.9.1 Endpoints del backend para el dashboard

Para poder mostrar el contenido correcto en el dashboard, primero es necesario obtenerlo de la base de datos. Para ello, se crearon dos endpoints en la API.

#### **6.9.1.1 Endpoint para obtener las campañas activas**

En la base de datos existe una tabla donde se almacenan todas las campañas activas y su información principal, y es necesario obtener esta información para poder mostrarla al usuario en el dashboard.

El endpoint que se debe crear es el que permite obtener las campañas disponibles. El query de este endpoint consiste en obtener todos los registros de la tabla de campañas

Cuando el Frontend realice la petición al Backend, este último le retornará un JSON con las campañas activas.

#### **6.9.1.2 Endpoint para obtener la información del usuario**

Este endpoint permitirá obtener la información de un cliente de la tabla correspondiente en la base de datos.

Como en endpoints anteriores, lo primero que se tiene que hacer es crear el query que permita obtener la información. Es importante enviarle como parámetro el ID del cliente para que sólo se obtenga la información del cliente que se requiere y no se obtengan todos los registros existentes.

La obtención del ID se realiza con la ayuda del Frontend. Cada que se realice la petición HTTP a este endpoint se envían las credenciales, es decir, la cookie de sesión y esta será procesada por la API. La cookie contiene un token JWT con información del usuario, y entre esta información se encuentra el ID por lo que se debe extraer el token de la cookie, decodificarlo y obtener el ID para pasarlo al query.

Una vez ejecutado el query, el endpoint retornará un JSON con la información del cliente.

### **6.9.2 Interfaz del Dashboard**

Al iniciar sesión, la primera página que ve el cliente es el dashboard. En este dashboard se muestra un mensaje personalizado para el cliente y las campañas activas de Mezfer (Ver Figura 7).

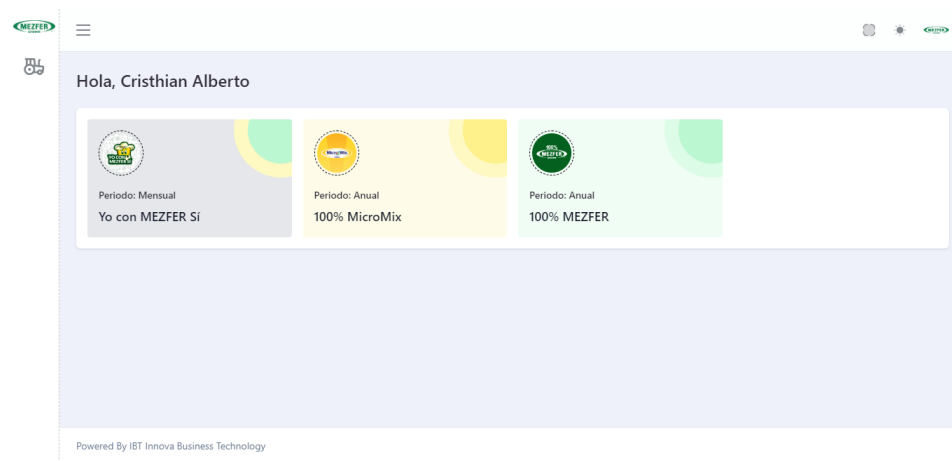


Figura 7: Dashboard de clientes.

Antes de cargar las campañas y el mensaje personalizado, se renderiza un componente llamado “Skeleton” que muestra la estructura de la página pero sin ningún contenido, sólo elementos grises que representan que la información se está cargando. Este componente se muestra en lo que se obtiene la información de la API.

Una vez se haya cargado la información, el Skeleton deja de renderizarse y ahora muestra el mensaje personalizado y las campañas activas en componentes llamados “Cards”. El cliente puede dar clic a cada una de estas Cards, lo que lo llevará a otra página donde se muestran todos los detalles de la campaña, pero esto se explicará más adelante.

## 6.10 Sección “Mis campañas”

La sección de campañas tiene un diseño muy similar al dashboard del cliente, lo que las hace diferentes es que la sección de campañas muestra en cuáles se encuentra inscrito el usuario.

### 6.10.1 Endpoints del backend de la sección “Mis campañas”

La sección de campañas sólo requiere de dos endpoints: uno para obtener las campañas activas y otro para obtener las campañas en las que está inscrito el usuario.

#### 6.10.1.1 Endpoint para obtener las campañas activas

Como se mencionó anteriormente, la interfaz de esta sección es muy similar a la del dashboard, por lo que comparten un mismo endpoint para obtener las campañas activas.

Para saber como funciona este endpoint, puede leer la sección 6.9.1.1 de este documento.

#### **6.10.1.2 Endpoint para obtener las campañas en las que se encuentra inscrito el usuario**

Para obtener la información que se requiere, se hará uso de la tabla que es producto de una relación muchos a muchos entre las tablas que contienen la información de los usuarios y de las campañas. Esta tabla intermedia permite relacionar un usuario con muchas campañas, permitiendo así que pueda estar inscrito en varias camapañas.

Se debe construir el query que permita obtener la información de la tabla correspondiente. Este query requiere del ID del usuario, para que sólo devuelva la información del usuario que se desea y no toda la información que contenga la tabla, además, también se debe verificar que el usuario siga inscrito en esta campaña; si no se valida esto, se obtendrán todas las campañas en las que ha estado el usuario, sin importar si sigue inscrito o no.

Con el query construido, al realizar la petición HTTP a este endpoint, se obtendrá un JSON con las campañas en las que está inscrito el usuario.

#### **6.10.2 Interfaz de la sección “Mis campañas”**

En la parte izquierda de la página web, el usuario podrá ver una barra de navegación que contiene un ícono de un tractor. Al dar clic en este tractor, el usuario será dirigido a la sección “Mis campañas”.

En esta sección también se utiliza el componente “Skeleton” para indicar que se está cargando la información.

Cuando se cargan la información, el Frontend recibe todas las campañas activas y las campañas en las que se encuentra inscrito el usuario. Antes de construir la interfaz, se realiza un filtrado de los resultados obtenidos. Dentro del arreglo que contiene las campañas activas se buscan coincidencias con el arreglo que contiene las campañas en las que está inscrito el usuario; las campañas que se encuentren en ambos arreglos se almacenan en un nuevo arreglo, y las que no coincidan se ignoran.

Este nuevo arreglo es el que se utiliza para renderizar las Cards de las campañas, y de esta

manera sólo se le muestran al usuario sus campañas correspondientes (Ver Figura 8).

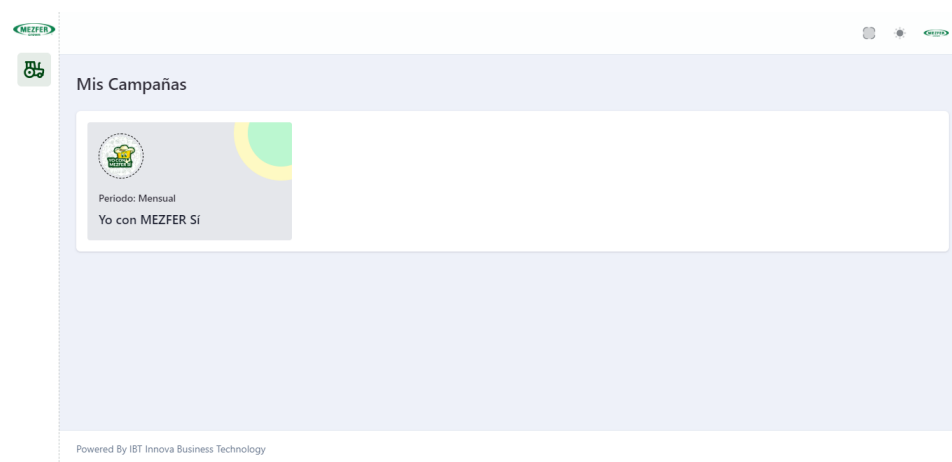


Figura 8: Sección “Mis campañas”.

## 6.11 Detalles de las campañas

Esta es una subsección de la sección de campañas. Aquí los usuarios podrán ver toda la información específica de una campaña.

### 6.11.1 Endpoints del backend de la sección “Detalles de las campañas”

Esta sección requiere de una cantidad mayor de endpoints debido a toda la información que se debe obtener para poder mostrarla a los usuarios.

#### 6.11.1.1 Endpoint para obtener la información de la campaña

La información de la campaña está distribuida en diferentes tablas de la base de datos.

Principalmente, se debe utilizar la tabla de campañas pues aquí está almacenada información como el nombre, imágenes, fechas para obtener puntos y canjearlos, entre otras cosas. Posteriormente, esta tabla se debe unir con la tabla de descripciones para recuperar el número de la descripción y su contenido, de esta manera todos los párrafos que conforman la descripción podrá ser ordenados adecuadamente. Finalmente, se realiza otra unión con otra tabla donde se almacenan los tipos de descripciones para identificar el tipo de descripción que es cada párrafo y así poder asignarle un color.

Este query requiere del ID de la campaña para evitar que se obtenga la información de todas las campañas, sólo se debe obtener la información de la campaña de la que se quieran ver

los detalles.

Ya que el query esté construido, se podrá realizar la petición HTTP a este endpoint y se recibirá como respuesta un JSON con la información solicitada.

#### **6.11.1.2 Endpoint para obtener las campañas en las que se encuentra inscrito el usuario**

Este endpoint se vuelve a utilizar debido a que cierta parte de los detalles de las campañas se le muestra solamente a los usuarios que estén inscritos a la campaña. Este renderizado condicional será posible gracias a la información que se recibe de este endpoint.

Para obtener una explicación más detallada de este endpoint, puede revisar la sección 6.10.1.2 de este documento.

#### **6.11.1.3 Endpoint para obtener los premios de las campañas**

Cada una de las campañas ofrece premios diferentes para los usuarios que deseen canjear sus puntos.

La tabla principal para comenzar a desarrollar este query es una tabla intermedia producto de una relación muchos a muchos en las tablas de las campañas y los premios. En esta tabla intermedia se guardan los premios y a qué campañas pertenecen, así como la cantidad de puntos que se requieren para poder ganar esos premios.

Esta tabla intermedia se debe unir con las tablas de campañas y premios para poder obtener correctamente los nombres.

Este query también debe recibir el ID de la campaña para sólo obtener los premios específicos de esa campaña.

Cuando el query esté construido, se podrán realizar peticiones HTTP a este endpoint para recibir la información solicitada.

#### **6.11.1.4 Endpoint para obtener los productos participantes de la campaña**

Cada campaña tiene ciertos productos que, al comprarlos, permiten ganar una cantidad de puntos.



Al igual que el endpoint anterior, la tabla principal es una tabla intermedia producto de la relación muchos a muchos de las tablas de campañas y productos. En esta tabla se guardan los productos y en qué campañas participan, así como la cantidad de puntos que un cliente puede recibir con su compra.

Esta tabla se debe unir con las tablas de campañas y productos para poder obtener correctamente los nombres.

Es importante que el query reciba el ID de la campaña para sólo obtener los productos de una determinada campaña.

Con el query construido, se puede realizar una petición HTTP para obtener la información correspondiente.

#### **6.11.1.5 Endpoint para obtener las solicitudes de puntos de un usuario**

Para poder obtener los puntos de la compra de productos, los usuarios realizan una solicitud de puntos la cual será revisada y aprobada o rechazada.

Mediante este endpoint, los usuarios pueden obtener un historial de todas las solicitudes que han realizado.

Para poder crear este query se utilizan cuatro tablas distintas. Principalmente se utiliza la tabla donde se almacenan todas las solicitudes para poder obtener la cantidad de puntos solicitados, la evidencia y la fecha; posteriormente, se realiza la unión con la tabla de estatus para poder obtener el nombre del estatus. Las otras dos tablas que se requieren son las de campañas y usuarios, de estas no se obtiene ninguna información, sólo son utilizadas para limitar, mediante los IDs del usuario y la campaña, la información que se va a obtener.

Ya que el query esté construido, se podrá realizar la petición HTTP.

#### **6.11.1.6 Endpoint para obtener los canjes del usuario**

Los puntos que cada uno de los usuarios acumule podrán ser canjeados por los premios que pueda ofrecer la campaña.

Gracias a este endpoint, los usuarios pueden obtener un historial de los canjes que han

realizado.

Para poder construir este query se requieren de 5 tablas. La tabla principal es aquella donde se almacenan los canjes, de aquí se van a obtener la cantidad de premios que se canjearon y la fecha en la que se canjearon; esta tabla se va a unir con las tablas de premios y estatus para poder obtener el premio que ha sido canjeado y el estatus del canje. Las últimas dos tablas son las de campañas y usuarios, y al igual que el endpoint anterior, estas sólo son utilizadas para limitar la información que se va a obtener.

Con esto queda terminado el query y se pueden realizar las peticiones HTTP para obtener la información.

#### **6.11.1.7 Endpoint para obtener los puntos del usuario**

Este endpoint permite obtener la cantidad de puntos que tiene cada usuario.

Para crear este query se requieren de tres tablas. La tabla principal es una tabla intermedia producto de la relación muchos a muchos entre las tablas de usuarios y campañas. En esta tabla se almacena el usuario y las campañas a las que está inscrito, así como la cantidad de puntos que tiene en cada campaña.

Para poder limitar los resultados de este query se deben cumplir tres condiciones: que el ID que se reciba del usuario exista, el ID que se reciba de la campaña exista y que la fecha de vencimiento de los puntos no haya llegado.

Es así como, al realizar la petición HTTP a este endpoint, el usuario recibirá sus puntos que sigan vigentes.

#### **6.11.1.8 Endpoint para realizar una solicitud de puntos**

Este es el último endpoint que utiliza esta sección.

Para poder realizar una solicitud de puntos, el usuario debe llenar un formulario donde se le solicita que indique que productos compró, la cantidad y suba una imagen que sea evidencia de esta compra.

Es en este endpoint donde se integra un nuevo servicio de Amazon: Amazon Simple Storage

Service (Amazon S3) que proporciona los llamados “buckets” para almacenar archivos. Para poder utilizar Amazon S3, es necesario instalar la librería para el lenguaje de programación que se esté utilizando (que en este caso es JavaScript) que proporciona Amazon. Una vez instalada, en un archivo específico para esta configuración, se importa esta librería para acceder a la clase necesaria para la configuración. Para una configuración básica sólo se requieren de tres datos: la región donde está ubicado el bucket, una llave de acceso y una llave de acceso secreta; esta información se obtiene a través de la cuenta de AWS.

El objeto que se obtiene como resultado de esta configuración es el siguiente (Ver Figura 9):

```
const s3 = new S3Client({
  credentials: {
    accessKeyId: accessKey,
    secretAccessKey: secretAccessKey,
  },
  region: bucketRegion,
});
```

Figura 9: Objeto de configuración de Amazon S3.

Una vez hecha la configuración de Amazon S3, se puede continuar con el procesamiento de los datos.

Como se mencionó anteriormente, el usuario llena un formulario para enviar la información de su solicitud de puntos. El Backend no recibe esta información en formato JSON, como se hizo en el endpoint de inicio de sesión, esta vez la recibe como FormData, y esto es porque el usuario debe subir un archivo, y este formato es el más adecuado para estas situaciones.

Este FormData está formado por dos pares llave/valor: uno con toda la información del formulario (que esta sí está en formato JSON) y otro con el archivo que se va a subir.

El controlador recibe la petición con el FormData y se extraen el ID del usuario, el archivo y la información del formulario. Todo esto, posteriormente, es enviado al servicio.

El servicio es el que se encarga de procesar toda esta información. Primero se trabaja con el archivo, se obtiene la extensión del archivo y se verifica si está entre las permitidas; si no es permitido, se detiene el proceso de subida del archivo y se manda un error,

si es permitido, avanza con el proceso. Lo siguiente que se hace es generar una ruta para almacenar la imagen en el bucket, esta ruta tiene la siguiente estructura: “Campania/idCampania/Pruebas/idUsuario”. Posteriormente se crea un nombre único para el archivo mediante un algoritmo de encriptación que retorna una cadena de caracteres aleatorios, este nombre se anexa al final de la ruta creada anteriormente. Para subir el archivo, se crea un objeto con los parámetros que requiere Amazon S3 para subir un archivo a un bucket, este objeto tiene la siguiente estructura (Ver Figura 10):

```
const params = {  
  Bucket: bucketName,  
  Key: fullPath,  
  Body: file.buffer,  
  ContentType: file.mimetype,  
  ACL: "public-read",  
};
```

Figura 10: Parámetros para subir un archivo a un bucket.

Los parámetros significan lo siguiente:

- Bucket: Nombre del bucket donde se guardará el archivo.
- Key: La ruta donde se guardará el archivo.
- Body: El contenido del archivo que se subirá.
- ContentType: El formato del archivo.
- ACL: La lista de control de acceso para indicar quien podrá ver el archivo y quien no.

Estos parámetros se envían al método correspondiente al objeto que se creó anteriormente en el archivo de configuración, y de esta manera el archivo quedará subido.

El siguiente paso es insertar toda la información en la tabla de la base de datos. Una gran parte de los campos requeridos ya venían en el FormData, por lo que sólo es cuestión de extraerlos y colocarlos en un nuevo objeto que se envía al método encargado de insertar los datos, el único campo nuevo que se debe crear es la URL completa de la imagen, es decir, la URL que contiene tanto el servidor del bucket y la ruta que se creó anteriormente. Este nuevo campo se anexa al objeto y finalmente se envía para insertar esos datos.

Este método retorna el ID del nuevo registro que se creó, este se almacena en una variable

para utilizarlo posteriormente. Cabe mencionar que, hasta el momento de la redacción de este reporte, las solicitudes son aceptadas automáticamente, esto por petición de la empresa, lo ideal es que en un futuro las solicitudes pasen por su debido proceso de revisión.

También se deben de insertar los detalles de la solicitud, esto es los productos que se compraron y la cantidad. Al igual que los datos de la inserción anterior, estos detalles también se obtienen del FormData, sólo es cuestión de extraerlos y anexarlos al nuevo objeto que se va a enviar; a este nuevo objeto también se le va a anexar el ID que se almacenó anteriormente.

Para terminar este proceso, sólo falta calcular los nuevos puntos del usuario. Lo primero que se debe hacer es obtener los puntos del usuario, esto se puede hacer utilizando el query que se creó para otro endpoint. El resultado de esta consulta determinará el camino que debe seguir el proceso.

Si el usuario ya tenía puntos se debe realizar una actualización; se toman los puntos que ya tenía y se suman los nuevos que se están solicitando. Realizada esta suma, los nuevos puntos son enviados al método que se encarga de actualizar los puntos.

Si el usuario no tenía puntos se debe realizar una inserción. El primer paso es obtener el día de canje de puntos de la campaña, esto para poder asignarle una vigencia a los nuevos puntos, y después se guarda la fecha actual. A la fecha actual se le adelanta 1 mes, y después se le asigna el día del canje. Por ejemplo: el día del canje es 5, y la fecha de hoy es 20/12/2024, a esta fecha se le adelanta 1 mes por lo que quedaría en 20/01/2025, por último se le establece que el día sea 5 y ahora la fecha quedaría 5/01/2025, esta sería la fecha de vigencia de los puntos.

Esta nueva fecha calculada se anexa al objeto que contiene los datos que se enviarán al método encargado de insertar los puntos en la tabla de la base de datos, y finalmente se insertan los datos.

Como último detalle, todas estas operaciones que se realizaron están dentro de una transacción, por lo que cualquier error que pueda ocurrir en cualquiera de estas detendrá todo el proceso, borrará cualquier operación que se haya realizado y quedará como si no

hubiera ocurrido nada.

### 6.11.2 Interfaz de la sección “Detalles de las campañas”

Anteriormente se había mencionado que las Cards donde se mostraban las campañas, cuando el usuario hace clic en ellas, dirigían a una nueva página. Esta nueva página es la sección donde el usuario puede leer toda la información de la campaña que haya seleccionado.

Debido a que esta sección es un poco más grande que otras que se han descrito, se va a separar en dos: el encabezado de la sección y el contenido.

Al igual que en secciones anteriores, mientras se está cargando la información se renderiza el componente “Skeleton”. Una vez cargada la información, se muestra el contenido normal de la página.

#### 6.11.2.1 Encabezado de la sección “Detalles de las campañas”

Cuando el usuario entra a ver la información de una campaña, lo primero que puede ver es el encabezado de la página. En este encabezado se puede ver información básica de la campaña como su nombre, el logo y el banner, además, el usuario también podrá ver la cantidad de puntos que tiene en esa campaña.

Otro componente que podrá ver el usuario es uno llamado “Breadcrumbs” que básicamente es un pequeño menú de navegación que muestra toda la ruta de páginas que debió seguir el usuario para llegar a la sección donde se encuentra actualmente.

El encabezado se ve de la siguiente manera (Ver Figura 11):



Figura 11: Encabezado de la sección.

### 6.11.2.2 Contenido de la sección “Detalles de la campaña”

El contenido de la sección está dividido en 8 Cards diferentes.

La primera Card es la de “Mis puntos”, el usuario puede ver un historial de las solicitudes de puntos que ha realizado. Si el usuario nunca ha realizado una solicitud de puntos, en la Card sólo le mostrará un mensaje indicándole esto (Ver Figura 12).

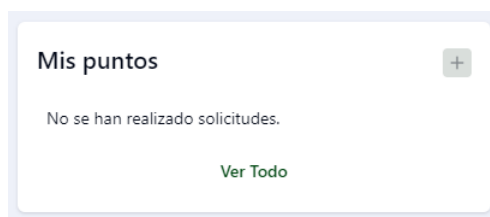


Figura 12: Card de puntos sin contenido.

Pero si el usuario ya tiene solicitudes hechas, se le mostrarán máximo las últimas 6 solicitudes que ha realizado con información general de la solicitud como la imagen de la evidencia que se subió, la cantidad de puntos que se solicitaron, la fecha en la que se realizó la solicitud y el estatus de la misma (Ver Figura 13).



Figura 13: Card de puntos con contenido.

Si el usuario quisiera ver todas las solicitudes que ha realizado, puede presionar el botón

que se encuentra en la parte inferior de la Card con la leyenda “Ver Todo”. Al presionarlo, se mostrará un componente llamado “Dialog” y su contenido dependerá de la cantidad de solicitudes hechas; si no hay ninguna, el Dialog sólo contendrá un mensaje indicando esto. (Ver Figura 14).

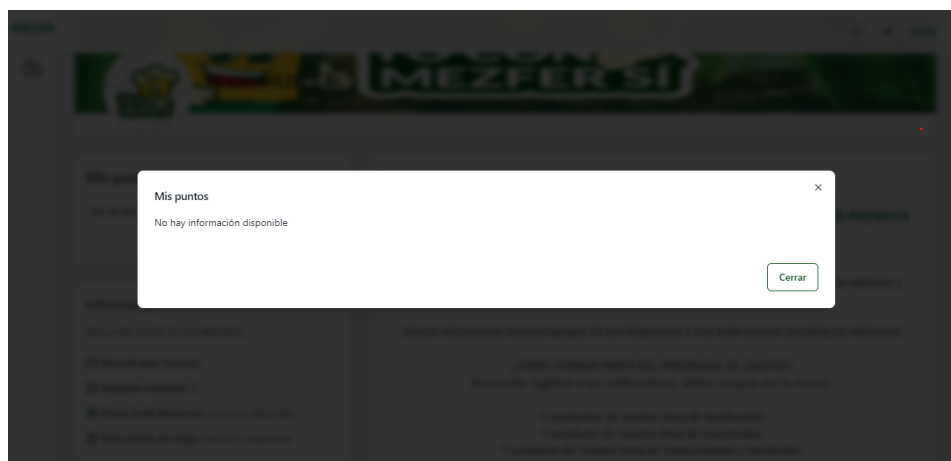


Figura 14: Dialog de puntos vacío.

Si ya hay solicitudes, el Dialog contendrá una tabla con la información de todas las solicitudes (Ver Figura 15).

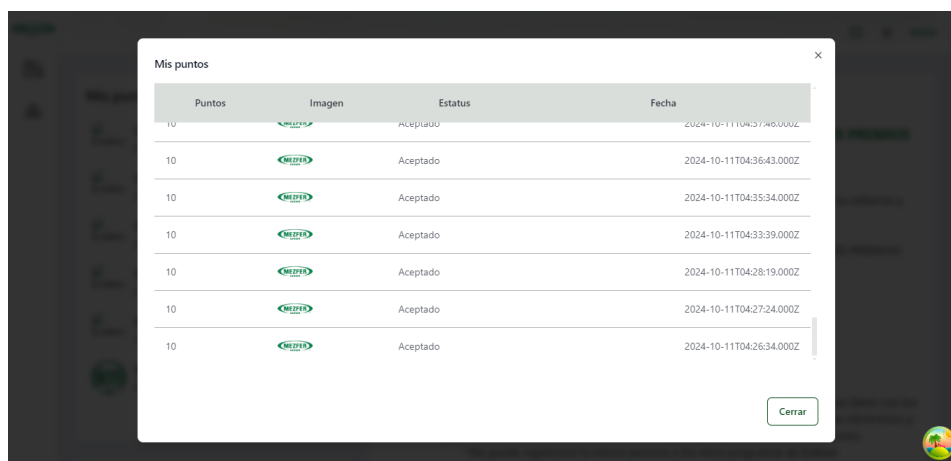


Figura 15: Dialog de puntos con contenido.

Otra característica importante de esta Card es que permite realizar una solicitud de puntos, esto se hace mediante el botón con un ícono “+” que se encuentra en la esquina superior derecha. Al presionarlo, al usuario se le mostrará un Dialog con un formulario para que



ingrese toda la información necesaria (Ver Figura 16).

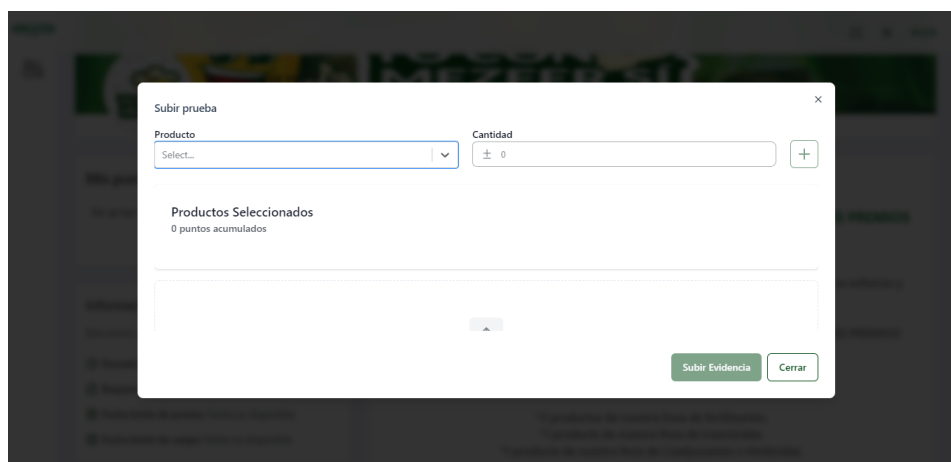


Figura 16: Dialog con formulario para realizar la solicitud.

En la segunda Card se muestra información relevante de la campaña respecto a la periodicidad de esta y las fechas de solicitudes y canjes de los puntos (Ver Figura 17).

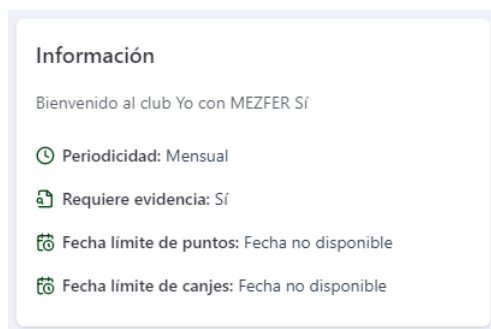


Figura 17: Card con información de la campaña.

La tercera Card es donde el usuario puede canjear sus puntos por un premio. Hasta el momento de la redacción de este reporte, esta funcionalidad no está implementada, en la Card sólo se ve un botón pero al presionarlo no ocurre nada (Ver Figura 18).

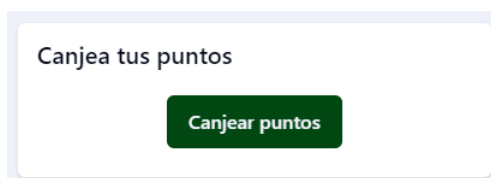


Figura 18: Card de canje de puntos.

La cuarta Card muestra los premios que se pueden obtener en la campaña. A simple vista, al usuario se le muestran 6 premios como máximo y sus respectivos detalles como la imagen del premio, el nombre y la cantidad de puntos que se requieren para obtenerlo (Ver Figura 19).

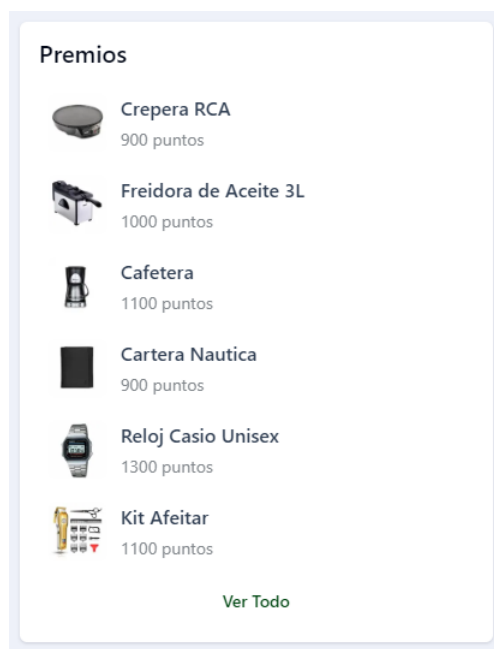


Figura 19: Card con los premios de las campañas.

Si el usuario quisiera ver todos los premios disponibles, deberá presionar el botón con la leyenda “Ver Todo” que se encuentra en la parte inferior de la Card. Al presionarlo, se mostrará un Dialog con una tabla que contiene la información de todos los premios (Ver Figura 20).

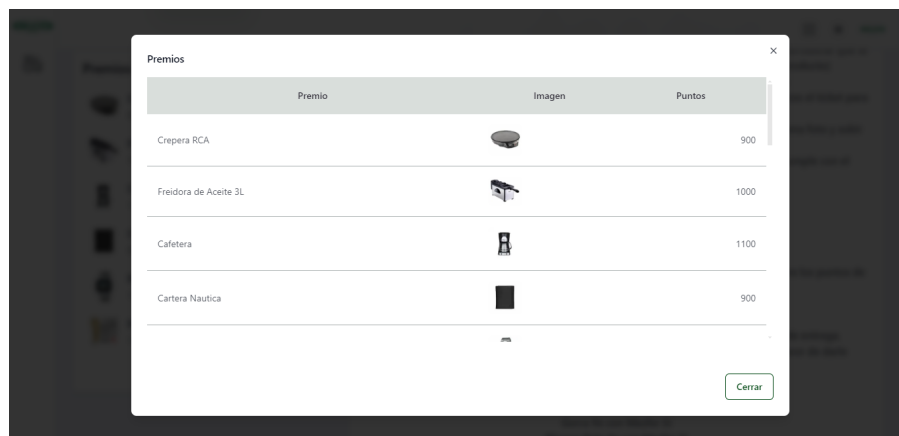


Figura 20: Dialog con los premios de la campaña.

La quinta Card es donde se muestra la descripción de la campaña. En la base de datos, las descripciones están guardadas con información que permite darles el orden y el formato adecuado:

- Número: Este número indica el orden de cada párrafo de la descripción.
- Tipo: A cada párrafo se le asigna un tipo como título, subtítulo, información, etc., de esta manera se le puede asignar un color.

Para poder dar un formato adecuado de todo el texto en la Card, se utilizan arreglos y objetos: en el arreglo se ordenan los párrafos para que el texto tenga sentido, y en el objeto se crean pares llave/valor para que, de acuerdo al tipo de la descripción, se le asigne un color. Otro punto importante es que en la base de datos, el texto de cada párrafo tiene caracteres especiales para indicar los saltos de línea; si este texto se muestra sin darle un formato, estos caracteres también se mostrarán, por lo que es importante que el código pueda identificar estos caracteres y saber que se trata de saltos de línea, y para lograr esto se hace uso de expresiones regulares.

Cubiertos todos estos aspectos, la descripción de la campaña se puede colocar correctamente en la Card (Ver Figura 21).

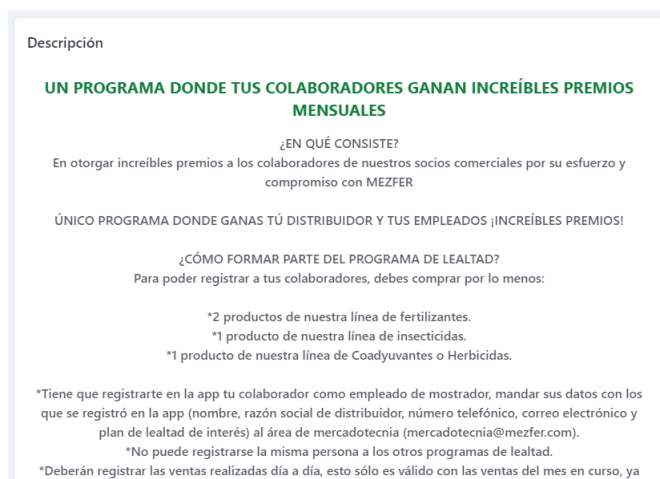


Figura 21: Card con la descripción de la campaña.

En la sexta Card, se muestra el cuadro de equivalencias de la campaña. Este cuadro de equivalencias muestra todos los productos participantes y la cantidad de puntos que se

pueden ganar al comprarlos (Ver Figura 22).



Figura 22: Card con el cuadro de equivalencias.

La séptima Card contiene una tabla con los productos que participan en la campaña. Esta tabla está paginada, cada página contiene como máximo 7 productos; también se puede realizar la búsqueda de un producto mediante su nombre con el campo que se encuentra en la parte superior de la tabla, este campo va filtrando el arreglo con todos los productos y retorna las coincidencias que haya con lo que el usuario haya escrito. (Ver Figura 23).

Productos Participantes			
<input type="text" value="Buscar producto..."/>		<button>Ver Todo</button>	
#	PRODUCTO	PUNTOS	UNIDADES
0853	USER 10 KG	5	1
0185	BIOS NUTRITION 8-24 NEW 19LT	3	1
5008	PRIMICIA 1 LT	5	1
0183	ARRANCADOR 8 24 0 19LT	3	1
0186	ARRANCADOR 15-10-5 1L	3	1
0252	POWER K1L	3	1
0516	KUMARI 1% GF 20 KG	5	1

0 de 50 fila(s) seleccionadas.

< 1 2 3 4 5 6 7 8 >

Figura 23: Card con los productos participantes.

Si el usuario no quisiera navegar por las páginas de la tabla, puede presionar el botón con la leyenda “Ver Todo” que se encuentra en la esquina superior derecha. Este botón mostrará un Dialog con una tabla con todos los productos (Ver Figura 24).

Numero	Producto	Unidades	Puntos
0853	USER 10 KG	1	5
0185	BIOS NUTRITION 8-24 NEW 19LT	1	3
5008	PRIMICIA 1 LT	1	5
0183	ARRANCADOR 8 24 0 19LT	1	3
0186	ARRANCADOR 15-10-5 1L	1	3
0252	POWER K1L	1	3

Figura 24: Dialog con los productos de la campaña.

La octava y última Card de esta sección contiene una tabla en la que el usuario podrá ver todos los canjes que ha realizado, esta tabla también contiene un campo que permite al usuario buscar cierto canje que haya realizado. Al igual que en Cards anteriores, si el usuario no ha realizado ningún canje, la Card contendrá un mensaje indicándole esto (Ver Figura 25).

FECHA	PREMIO	PUNTOS	ESTATUS
No results.			

0 de 0 fila(s) seleccionadas.

Figura 25: Card de canjes sin contenido.

Al presionar el botón con la leyenda “Ver Todo” que se encuentra en la esquina superior derecha, al usuario se le mostrará un Dialog con una tabla sin paginación con todos los canjes que ha realizado. En este caso, como no hay información que mostrar, sólo se mostrará un mensaje indicando esto (Ver Figura 26).

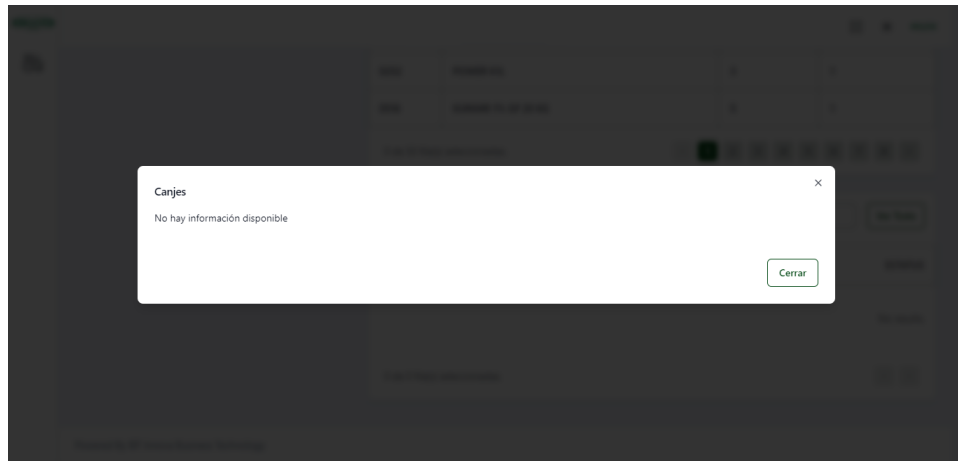


Figura 26: Dialog de canjes sin contenido.

Si ya hay canjes realizados, el usuario podrá ver todos los detalles de estos (Ver Figura 27).

Mis canjes			
		Q Buscar canje...	Ver Todo
FECHA	PREMIO	PUNTOS	ESTATUS
28 de septiembre de 2024	Tarjeta Spotify	1	Aceptado
28 de septiembre de 2024	Tarjeta Netflix	1	Aceptado
0 de 2 fila(s) seleccionadas.			
<div> <span>&lt;</span> <span>1</span> <span>&gt;</span> </div>			

Figura 27: Card de canjes con contenido.

Y esta misma información la podrá ver en el Dialog con una tabla sin paginación que se muestra al presionar el botón “Ver Todo” (Ver Figura 28).

Premio	Cantidad	Estatus	Fecha
Tarjeta Spotify	1	Aceptado	2024-09-28T21:36:44.000Z
Tarjeta Netflix	1	Aceptado	2024-09-28T21:40:18.000Z

Figura 28: Dialog de canjes con contenido.

## **6.12 Cambios generales en el Frontend de los clientes**

Los cambios que se realizaron en el Frontend de los clientes son relacionados al código y a algunos componentes.

En cuanto al código, se cambió la manera en la que se estaban realizando las peticiones HTTP. Las peticiones se estaban realizando de manera repetida en diferentes partes del código, por lo que se decidió realizarlas desde un mismo lugar y la información que se obtuviera se pasaría a los diferentes componentes que la necesitaran. Además, como la misma estructura de código se repetía, se decidió crear un hook para realizar las peticiones; ahora en vez de escribir el mismo código múltiples veces, simplemente se manda a llamar un método que solicita una URL y el tipo de petición.

Otro cambio importante que se realizó fue en los componentes. Algunos componentes como tablas, Dialogs y Cards se estaban creando por cada sección que los ocupara y esto sólo hacía que el tamaño del proyecto estuviera aumentando, es por eso que se decidió crear componentes reutilizables, esto permitió que en un solo archivo estuviera el código general de cada componente y sólo se les pasaba la información que se quisiera mostrar en ellos.

## **6.13 Dashboard de los administradores**

Como se menciona en el título de la sección, este es el dashboard exclusivo de los administradores, ningún usuario que no tenga el rol de administrador podrá verlo.

En este Dashboard se le presenta a los administradores estadísticas importantes de ventas, las campañas y los productos de la empresa.

### **6.13.1 Endpoints del Backend del Dashboard de administradores**

#### **6.13.1.1 Endpoint para obtener las ventas que se han realizado con el tiempo**

Este endpoint permite obtener información de todas las ventas que se han realizado a lo largo del tiempo.

El query que se construyó para este endpoint obtiene la información de una sola tabla, la tabla donde se almacena la información de las ventas. De aquí sólo se extrae la cantidad total de productos que se vendieron, el importe de la venta y la fecha en la que se realizó.

También es importante ordenar las ventas por su fecha, desde la primera que se hizo hasta la más reciente.

Una vez construido el query, se podrán realizar las peticiones HTTP para obtener la información.

#### **6.13.1.2 Endpoint para obtener los usuarios que se han registrado con el tiempo**

Este endpoint permite obtener todos los usuarios que se han registrado desde que se inició el proyecto de “Mezfer Rewards”.

Al igual que el endpoint anterior, el query para este endpoint obtiene la información de una sola tabla, la de los usuarios. En esta tabla se guarda la fecha en la que se registró cada uno de los usuarios, pero para dar un mejor manejo a esta información, estas fechas fueron divididas en periodos mensuales y de cada periodo se realiza una cuenta de la cantidad total de usuarios que se registraron. También se deben ordenar los resultados por la fecha, desde el primer registro hasta el último.

De esta manera, el endpoint está finalizado y listo para recibir peticiones.

#### **6.13.1.3 Endpoint para obtener el Top 10 de productos**

Este endpoint permite obtener un Top de los productos que generan mayores ingresos para Mezfer.

Mediante la tabla de ventas se obtiene la cantidad e importe de cada producto y, mediante la función Rank() de PostgreSQL, se realiza un ordenamiento para obtener el Top con base en la suma de los importes de cada venta de cada producto. Esta tabla se une con la tabla de productos para poder obtener el nombre de cada uno de los productos que entran en el top.

Con el query finalizado, es posible comenzar a realizar las peticiones HTTP.

#### **6.13.1.4 Endpoint para obtener el Top 10 de distribuidores**

Con este endpoint se obtiene un Top de los mejores distribuidores de productos que tiene Mezfer, es decir, los que venden mayor cantidad.



De la misma manera que con el query del top anterior, se utiliza la tabla de ventas para poder obtener información como el importe y la cantidad de productos que vende cada distribuidor. Con la ayuda de la función Rank() se realiza un ordenamiento para obtener el top con base en la suma de los importes de cada venta de cada producto. Por último, esta tabla se une con la tabla de distribuidores para poder obtener el nombre de cada distribuidor.

Con esto queda finalizado el query y, a su vez, el endpoint.

#### **6.13.1.5 Endpoint para obtener una comparativa de ventas entre el año actual y el año anterior**

Mediante este endpoint se obtiene la información de cuanto vendió la empresa en el año actual y cuanto vendió en el año anterior.

Nuevamente la tabla de ventas es el punto importante, pues es de aquí donde se extraen los datos de la venta como el importe y cantidad. Cuando se realiza la suma para obtener el ingreso total tanto del año anterior como del actual, se aplica una condición para limitar los datos que van a participar en la comparativa; esta condición indica que si el año de la venta es igual al actual o al anterior, se toma el valor del importe, si no lo es, se coloca como valor un 0.

Por último, para limitar los resultados del query, se añade una última condición para indicar que solo participan las ventas del año anterior y el actual.

De esta manera el query queda concluido y listo para recibir peticiones.

### **6.13.2 Interfaz del Dashboard de administradores**

En el dashboard de los administradores se encuentran 3 Cards diferentes, cada una con información diferente.

En esta sección también se utiliza un componente nuevo, que son las gráficas.

#### **6.13.2.1 Card con los ingresos de la empresa**

Esta Card tiene distintos elementos que permiten ver la información de los ingresos en diferentes formatos.

Principalmente, en el encabezado de la Card se encuentran dos pestañas que, al darles clic, muestran la información de diferente manera. La pestaña por defecto es la que lleva por título “Acumulado”, y en esta se muestra una gráfica de línea acumulada, es decir, una gráfica que muestra la suma acumulada de valores a lo largo del tiempo. Otro elemento importante en la manipulación de la gráfica es un botón que permite mostrar los valores en moneda o en cantidad, lo que también alterará como se ve la gráfica; el formato que está seleccionado por defecto es el de moneda.

Esta gráfica acumulada con formato de monto se ve de la siguiente manera (Ver Figura 29):



Figura 29: Gráfica de línea acumulada con formato de moneda.

Esta gráfica se verá alterada al cambiar el formato a cantidad, pues el conjunto de datos que se utiliza para generarla es diferente (Ver Figura 30).

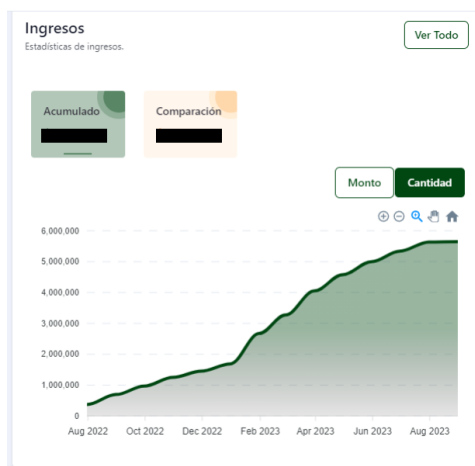


Figura 30: Gráfica de línea acumulada con formato de cantidad.

Para ver una comparación de ventas entre el año anterior y el actual, sólo es necesario dar clic en la pestaña que tiene como título “Comparación”. La gráfica que se presenta en esta pestaña también es de línea pero ya no es acumulativa, simplemente crece o se reduce de acuerdo a las ventas que se hayan realizado mes con mes, además, se muestran dos líneas para cada año.

Al igual que en la pestaña anterior, es posible cambiar el formato en el que se presentan los datos, ya sea en moneda o en cantidad. La gráfica con formato de moneda se ve de la siguiente manera (Ver Figura 31):



Figura 31: Gráfica de línea comparativa con formato de moneda.

Y la gráfica con formato de cantidad se ve así (Ver Figura 32):



Figura 32: Gráfica de línea comparativa con formato de cantidad.

Una última característica de todas las Cards es que tienen en la esquina superior derecha un botón con la leyenda “Ver Todo”. Este botón, al presionarlo, muestra un Dialog con una tabla con información más detalla de lo que se muestra en las gráficas. No es posible anexar una imagen de este Dialog pues contiene información que no se puede mostrar.

### 6.13.2.2 Card con los tops de la empresa

Esta Card, en el encabezado, tiene dos pestañas que mostrarán tops diferentes. La pestaña por defecto es la que lleva por título “Productos”, y en esta se muestra un gráfico llamado Treemap que muestra los datos jerárquicamente en forma de rectángulos anidados.

En esta Card también se encuentran los botones que permiten manipular la gráfica mediante el formato de moneda o cantidad. El formato por defecto es el de moneda (Ver Figura 33):

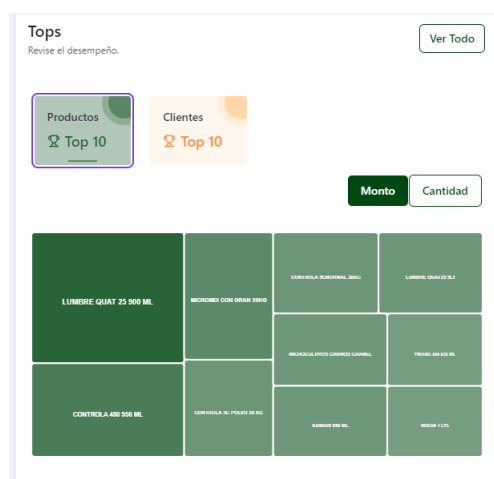


Figura 33: Gráfica treemap de productos con formato de moneda.

El gráfico mostrará 10 rectángulos con los 10 productos que le generan mayores ingresos a la empresa. Los rectángulos tendrán el nombre del producto y al pasar el cursor sobre ellos, se podrán ver los ingresos.

Al cambiar el formato a cantidad, el aspecto de la gráfica cambiará pues se está utilizando otro conjunto de datos para renderizarla. La gráfica volverá a mostrar 10 rectángulos con los 10 productos de los cuales se ha vendido mayor cantidad, y esto también provocará que la jerarquía de los rectángulos cambie (Ver Figura 34).

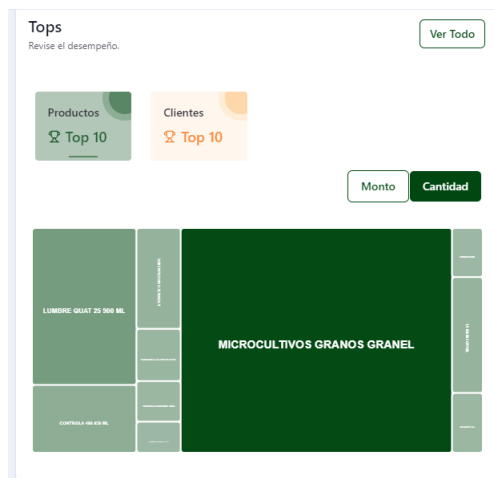


Figura 34: Gráfica treemap de productos con formato de cantidad.

Lo mismo ocurrirá al cambiar a la pestaña de clientes, sólo que ahora se mostrará quienes son los 10 clientes que generan más ingresos para Mezfer (Ver Figura 35):

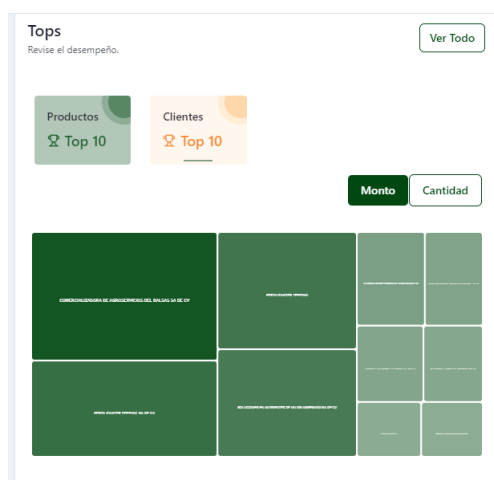


Figura 35: Gráfica treemap de clientes con formato de monto

### 6.13.2.3 Card con los usuarios que se han registrado a Mezfer Rewards

Esta Card también contiene en el encabezado dos pestañas que permiten ver la información de diferente manera. La pestaña predeterminada es la que lleva por título “General”, y aquí se muestra una gráfica de barras con los usuarios que se han registrado mes con mes desde que se creó el proyecto de Mezfer Rewards hasta la actualidad (Ver Figura 36).

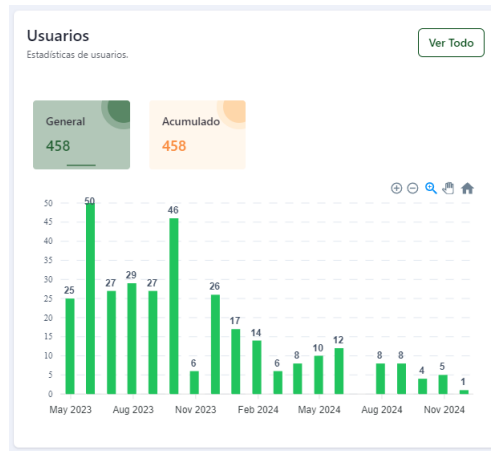


Figura 36: Gráfica de barras de los usuarios registrados.

En la pestaña con el título “Acumulado” se muestra una gráfica de línea acumulada (Ver Figura 37).

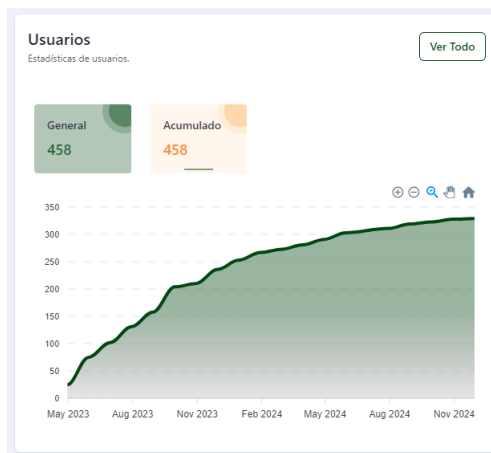


Figura 37: Gráfica de línea acumulada de los usuarios registrados.

## 6.14 Cambios generales en el Frontend de los administradores

En esta actividad sólo se realizaron cambios en el código para hacerlo más legible y algunas funciones fueron modificadas para mejorar su rendimiento, pero de manera general no hubo cambios muy grandes o drásticos.

Otro cambio que se realizó fue en las gráficas que se utilizaron para el dashboard de los administradores, ahora son componentes reutilizables, de esta manera se podrán utilizar en cualquier parte de la aplicación sin necesidad de escribir el mismo código múltiples veces.

## 6.15 Prueba