EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

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

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 respectivas 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 administradoes 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. 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.
- Cascading Style Sheet (CSS): Este lenguaje de estilos es utilizado para describir la presentación de documentos HTML o XML.
- Hypertext Preprocessor (PHP): Es un lenguaje de programación del lado del servidor que puede integrarse en HTML para crear aplicaciones y sitios web dinámicos.
- 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.

El desarrollo web se puede dividir en tres categorías: Frontend, Backend y Full Stack.

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 sitemas.

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 parete 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 framewrok 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.

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.

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 componenetes de React para construir la interfaz de usuario mientras que Next.js se encarga de algunas características adicionales y optimizaciones.

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, 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. 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

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.

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.

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.

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.

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.

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.

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 horizon-

talmente 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.

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, tambien 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.

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 serviciosy 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.

La infraestructura de AWS está basada en centros de datos distribuidos en todo el mundo, lo que permite a los usuarios implementar sus apicaciones y servicios en ubicaciones

geográficas específicas según sus necesidades. Además, posee una amplia gama de herramientas de gestión y automatizaciíon para ayudar a los usuarios a administrar y controlar sus recursos en la nube de mamera 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 desarollar 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 fina de satisfacer las necesidades en ocho motores.
- Amazon Simple Sotrage 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.

5.5 Metodología para la gestión de proyectos

5.5.1 Metodología Kanban

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 estás 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 utilizas 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 tecnologia 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 escribimos 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 dependen-

cias 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 ejectuar 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 dependecias.

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:

```
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 ejectuar 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

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 vinuclados, 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.

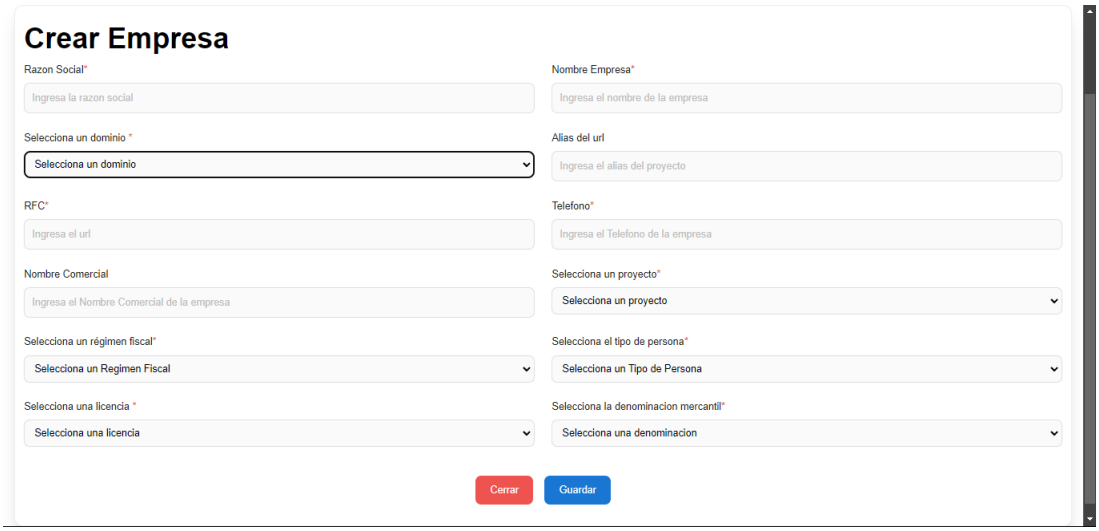

Figura 2: Formulario para la creación de empresa.

Posteriormente, el usuario debe ser vinculado con la empresa.

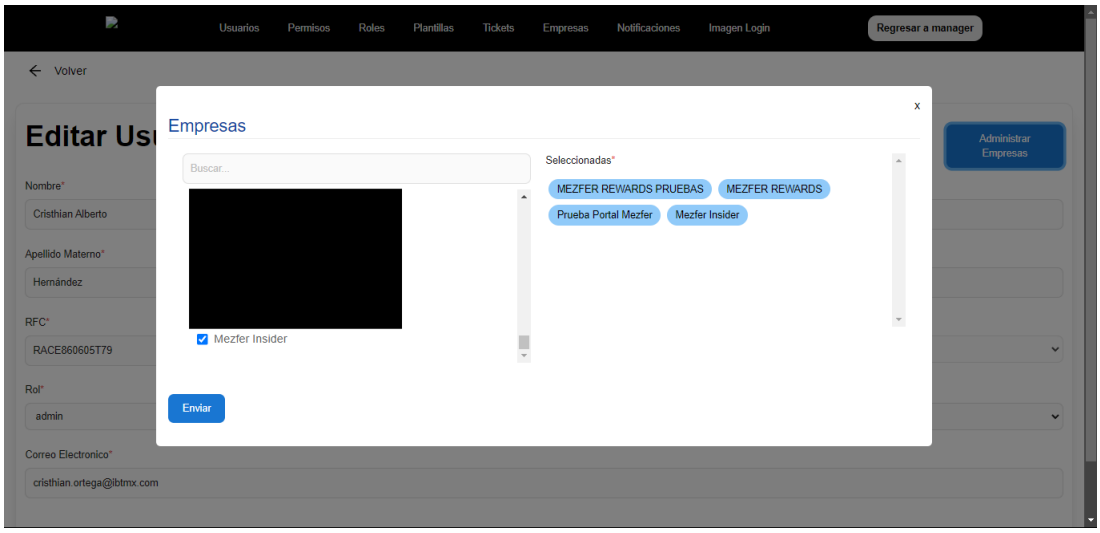

Figura 3: Vinculación de usuario con empresa.

Y con esto realizado, el usuario ya podrá tener acceso a la empresa.

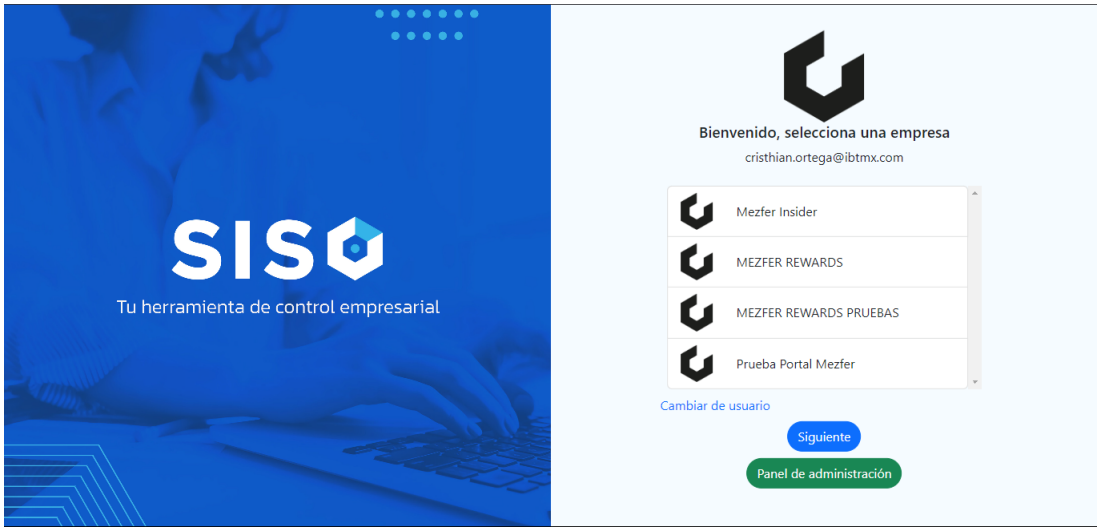

Figura 4: Selección de empresa.

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, incluído 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:

```
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 Formulario de inicio de sesión

Para que un cliente pueda iniciar sesión necesita de un formualrio 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.

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.

6.8.2.2 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 busqueda 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 desencriptar 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.

6.9 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.

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 esta 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.