2022-08

Diseño e Implementación de un Servicio de Back End para un directorio digital interactivo de instituciones públicas y privadas en Chile

Miranda Mejías, Martín Eduardo

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO CHILE

"Diseño e Implementación de un Servicio de Back End para un Directorio Digital Interactivo de Instituciones Públicas y Privadas en Chile"

MARTÍN EDUARDO MIRANDA MEJÍAS

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUíA
MARCOS ZÚÑIGA BARRAZA

PROFESOR CORREFERENTE
NICOLÁS JARA CARVALLO

Agosto 2022

Agradecimientos

Quiero agradecer a mi familia y amigos por apoyarme en todo momento de mi vida, en especial en esta etapa tan importante. También quiero agradecer a la Tuna y la Orquesta de la universidad por acompañarme durante toda mi vida universitaria.

Resumen

La falta de actualización y vigencia de la información de contacto de Instituciones públicas y privadas en Chile ocasiona generalmente problemas a la ciudadanía cuando necesita obtener información importante, ya sea para algún trámite o por que requiere contratar algún servicio. Este problema puede obtener una importancia crítica en algunos servicios, como por ejemplo los servicios de salud, las municipalidades,trámites notariales, etc.

Esto puede ser molesto para las personas ya que aumentan el tiempo que se demoran para poder realizar dichas actividades, tiempo que en situaciones de emergencias es crítico.

Para poder solucionar este problema se propone un sistema que se divide en dos partes:

La primera, es el diseño e implementación de un directorio digital que pueda ser utilizado por los distintos servicios, ya sea para uso institucional privado o para acceder a datos de contacto de otros servicios.

La segunda parte es el desarrollo de un *plugin* web que mantenga la información de contacto de cada servicio actualizada en tiempo real.

En base a la propuesta anterior se comprobará que el hecho de mantener estos datos centralizados permite una mejor administración y visualización de estos.

Finalmente, se revisarán aspectos a implementar en un futuro como puede ser la seguridad en la transmisión de la data o el levantamiento a producción del servicio desarrollado.

Este escrito se enfocará principalmente en el trabajo realizado por el equipo multidisciplinario desde el punto de vista del *BackEnd*, donde se encuentran las contribuciones principales del autor.

Esta Memoria se desarrolla en el marco del Programa de Memorias Multidisciplinarias, donde se busca resolver el problema anterior planteado por la empresa InforedChile.

Abstract

The lack of updating and validity of the contact information of public and private institutions in Chile generally causes problems to citizens when they need to obtain important information, either for any procedure or because they need to hire any service. This problem can become critical in some services, such as health services, municipalities, notary procedures, etc.

This can be annoying for people because it increases the time it takes to perform such activities, time that in emergency situations is critical.

In order to solve this problem we propose a system that is divided into two parts:

The first, is the design and implementation of a digital directory that can be used by different services, either for private institutional use or to access contact data from other services.

The second proposal is the development of a web plugin that keeps the contact information of each service updated in real time.

Based on the previous proposal, it will be proven that keeping this data centralized allows a better administration and visualization of these data.

Finally, aspects to be implemented in the future will be reviewed, such as security in the transmission of the data or the production of the developed service.

This paper will focus mainly on the work done by the multidisciplinary team from the point of view of the BackEnd, where the main contributions of the author are found.

This Thesis is developed within the context of the Multidisciplinary Memories Program, which seeks to solve the problem posed by the company InforedChile.

Glosario

ApiRest: Arquitectura de software para sistemas hipermedia distribuidos.

BackEnd: Parte que procesa la entrada desde el FrontEnd.

Bases de datos: Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

CLI: Interfaz de línea de comandos.

EndPoint: URLs de un API o un backend que responden a una petición.

Framework: Marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software.

FrontEnd: Parte del software que interactúa con los usuarios.

Heroku: Plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación.

HTML: Lenguaje de marcado que nos permite indicar la estructura de nuestro documento mediante etiquetas.

HTTP: Protocolo de transferencia de hipertextos.

JavaScript: Lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

MySQL: Tipo de base de datos relacional.

ORM: Modelo de programación que permite mapear las estructuras de una base de datos relacional.

Plugin: Pequeños programas complementarios que amplían las funciones de aplicaciones web.

Servidor: Conjunto de computadoras capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.

Widget: Pequeñas aplicaciones cuyo objetivo es dotar de información visual y facilitar el acceso a las funciones que se utilizan.

Índice general

1. Introducción

1.1. Contexto

Las bases de datos al día de hoy son uno de los recursos más valiosos en el mundo, y ello responde en gran medida al imparable crecimiento del mundo digital, reflejado en una generación constante de inmensa cantidad de información en internet, la que en su mayoría está almacenada en bases de datos.

Este fenómeno, que ocurre desde los inicios de internet no está exento de problemas, que van desde una estructura desordenada, problemas de actualización, mala visualización, entre otros. Además, no existe un sólo tipo de base de datos en el mundo ni una sola norma que las articule, en consecuencia, los esquemas de bases de datos actuales son de lo más diversos y presentan diferentes niveles de dificultad para trabajar en ellos.

En el caso de Chile, la bien denominada Transformación Digital ha dejado obsoletos importantes instrumentos de gestión y acceso a la información, como las agendas telefónicas o las “Páginas Amarillas”, que albergaban datos de contacto de un gran volumen de personas, empresas y servicios.

En los siguientes capítulos se abordará con mayor profundidad este problema y cómo se podría resolver.

En el capítulo 2 se revisarán las soluciones existentes y como nuestro servicio busca contribuir a solucionar este problema.

En los capítulos 4, 5 y 6 se analizarán los requisitos del sistema, su diseño, su implementación y su despliegue, enfocándose principalmente desde el punto de vista del BackEnd.

Finalmente en los capítulos 6 y 7 se realizará validación del trabajo realizado con pruebas de usuario y al sistema, para finalmente entregar las conclusiones finales.

1.2. Problemática a resolver

Actualmente, cada vez que en un servicio público se efectúan cambios en su directorio institucional, es necesario actualizar sus datos de contacto en cada una de las plataformas. Ello tiene como consecuencia que las actualizaciones se apliquen de manera tardía o incluso, en muchos casos, no se lleven a cabo, causando problemas y molestias a quienes requieran contactarse con algún servicio público.

InforedChile es un directorio de servicios públicos, además de pequeñas y medianas empresas. Esta plataforma se ve directamente afectada por el problema mencionado más arriba, ya que no existe una forma sencilla de saber si la información publicada está actualizada. A ello se debe sumar el retraso que implica solicitar los datos actualizados, ya sea de manera directa al servicio o a través de la Ley de Transparencia del Estado de Chile.

1.3. Hipótesis

Almacenar la información en un servidor centralizado de datos permite a los servicios públicos y a las pymes mejorar la administración y visualización de sus datos de contacto.

1.4. Objetivos

1.4.1. Objetivo General

Diseñar un directorio digital interactivo que permita visualizar, recolectar y administrar datos de contacto de instituciones en Chile.

1.4.2. Objetivos Específicos

- **OE1**: Elaborar una evaluación comercial y financiera del desarrollo e implementación del proyecto.
- **OE2**: Diseñar e implementar servicio de consulta de datos hacia la base de datos de contacto de instituciones y empresas de InforedChile.
- **OE3**: Diseñar e implementar aplicación web para administración de datos de las instituciones y personas almacenados en InforedChile.
- **OE4**: Implementar herramientas de visualización de datos de contacto en los sitios de las instituciones.

Los objetivos indicados anteriormente fueron planteados en el marco del “Programa de Memorias Multidisciplinarias”.

En este trabajo se aborda con mayor profundidad el OE2 el cual está enfocado principalmente en el servidor a implementar. También se abordará desde el lado del diseño el OE3 y el OE4 en los cuales el autor realizó aportes.

1.5. Acercamiento a la solución

Hoy por hoy, sólo se generan datos de contacto en las instituciones de forma manual y local en cada servicio. Este procedimiento no está centralizado, más allá del registro legal que entrega la Ley de Transparencia, que por medio de una solicitud vía correo electrónico, entrega planillas de contactos de la institución deseada que tenga hasta la fecha, con un tiempo de espera de hasta un mes.

Al no estar estos procesos centralizados en una sola plataforma, se vuelve necesario facilitar las labores de actualización manual de los datos, unificar los canales de recepción y automatizar su propagación.

Tomando en cuenta este escenario, se proponen las siguientes soluciones:

La primera, es el diseño e implementación de un directorio digital que pueda ser utilizado por los distintos servicios como directorio, ya sea para uso institucional privado o para acceder a datos de contacto de otros servicios. Este directorio busca ser una herramienta sencilla, donde los servicios puedan registrar, actualizar y acceder a sus datos de contacto, indicando si éstos son de uso público o privado.

La segunda propuesta es el desarrollo de un *plugin* web que mantenga la información de contacto de cada servicio actualizada en tiempo real.

Estas propuestas, si bien no solucionan el problema de la actualización manual de

los datos, sí facilitan considerablemente el proceso. Basta con realizar una sola vez la actualización de los datos de contacto para que éstos se vean reflejados en todas las plataformas utilizadas por un servicio público.

2. Estado del arte

2.1. Marco Teórico

Hoy existe una amplia gama de soluciones para el desarrollo de aplicaciones web y encontrar la mejor herramienta para cada aspecto del proyecto requiere de múltiples criterios, los que se detallan a continuación.

2.1.1. Servidor *Back End*

Arquitectura

Para el desarrollo de *back end* del desafío, se optó por utilizar la arquitectura *API REST*, debido a su simpleza y escalabilidad. Será profundizada en punto 4.2.

Entorno de ejecución

Se analizan las alternativas a *Node.js*, que se mencionan de antemano por conocimiento general, para el desarrollo de las APIs REST del *back end* de la solución, llegando a la conclusión que esta sigue siendo la mejor solución a la fecha, dada la versatilidad y dinamismo que entrega *Node.js*, sobre todo en el caso de este proyecto que se trata de una solución puramente web, donde el uso de *JavaScript* es imprescindible y por ende, desarrollar un *BackEnd* basado en ese lenguaje le brinda consistencia

y coherencia al proyecto. [5] [6] [7]

Entorno	Lenguaje	GitHub Stars (Usuarios)
Node.js	JavaScript	80.9k
SpringBoot	Java	56.9k
ASP.NET	.NET	25.3k
ELIXIR	Elixir	19k
PERL	Perl	1.1k

Tabla 2.1: Comparación de entornos de desarrollo de *BackEnd*.

Framework de BackEnd

Posteriormente, se estudiaron los *frameworks* de desarrollo de *BackEnd* sobre *Node.js* más populares, obteniendo la siguiente lista [9]:

Framework	GitHub Stars (Usuarios)	Descargas semanales NPM	Principal objetivo
Express.js	53.9k	16M	FW minimalista y flexible
Nest.js	39.4k	770k	Orientado a p. funcional
Meteor.js	42.6k	599	Prototipos y fácil de aprender
Koa.js	31.5k	883k	Alto rendimiento
Socket.io	54.1k	4M	Orientado a eventos en tiempo real

Tabla 2.2: Comparación de frameworks de desarrollo de *BackEnd*.

Al evaluar las ventajas y desventajas de cada *framework*, en primera instancia se descarta *Koa.io* y *Socket.io* dado que están dedicados al alto rendimiento y orientación a comunicación en tiempo real cliente-servidor. *Express* se deja preliminarmente fuera por ser la base de otros *frameworks*. Luego se llega a un empate técnico entre *Meteor.js* y *Nest.js* dado que ambos cumplen de forma similar con las necesidades del proyecto. [10] [11]

Dado que más adelante se elige *Next.js* como solución de *Front end*, se elige finalmente *Nest.js* como *framework* de *BackEnd* dado que al estar inspirado en *Express*, desarrollado en *TypeScript* y orientado a la programación funcional, este *Framework* propone la mejor forma de implementar la arquitectura *REST*. [11] [12] [13]

ORM

La integración del *BackEnd* con la base de datos requiere generalmente de un *ORM*. Para *Nest.js*, el más indicado, y por ende el que se usará en la solución es *TypeORM*, dado que es la alternativa más madura programada en *TypeScript*. [14]

2.1.2. Cliente *Front end*

Framework

Se analizaron distintos frameworks para desarrollo de aplicaciones web, entre el conocimiento previo del equipo e información en internet [15], llegando a acotar a las cuatro siguientes opciones, de las cuales se decidió optar por *React.js* para el *front end* de la aplicación web:

Framework	GitHub Stars (Usuarios)	Descargas semanales NPM	Característica principal
Vue.js	183k	2.4M	Simple de implementar.
React.js	173k	10.7M	Desarrollo eficiente, popular, en un solo lenguaje.
Angular	24.7k	1.3M/4.7M	Orientado a single page app.
Flutter	127k	No aplica	Multiplataforma, lenguaje dedicado.

Tabla 2.3: Comparación de *frameworks* de desarrollo de *back end*.

Analizando las diferentes ventajas y desventajas de cada *framework* en los distintos escenarios posibles [16], se llegó a la conclusión que *React* se ajusta de mejor

manera a las necesidades del proyecto, dado que está enfocado en sitios con rutas de páginas, lo que se ajusta a la solución planteada que consiste por un lado en una aplicación web tipo *dashboard*, con múltiples vistas, dos tipos de usuario y *plugins* incrustables, los cuales deben tener una visualización sencilla y dinámica en una página independiente.

2.1.3. Herramientas Adicionales

Para el desarrollo tanto del *dashboard* web para InforedChile y para sus clientes, como del plugin incrustable, se sugiere la utilización kit de componentes para interfaces de usuario de *React "Material-UI"*, diseñado con Material Design de Google.

Para facilitar el desarrollo de la aplicación web, se utilizará la plantilla para *Dashboards* de administración *"Minimals"*, que tiene licencia de uso comercial pagada y está desarrollada sobre el *Framework Next.js*.

Para el ambiente de desarrollo, se utilizará el *Framework* para React.js "Next.js" que provee de una experiencia de desarrollo más simplificada, reduciendo la "Fatiga de *JavaScript*", que consiste en evitar desperdiciar tiempo en la preparación del ambiente de desarrollo, la recopilación de bibliotecas y dependencias que se usan frecuentemente. Esto le permite al equipo de desarrollo enfocarse en el desarrollo de las vistas y la lógica de negocio en el *Front End*. Además provee características para el levantamiento de un ambiente de producción optimizado.

Plugin y Widget Incrustables

Para la integración de la aplicación web en los sitios de los stakeholders participantes, se utilizarán las herramientas de *HTML "iframe"* y *"widget"*, de forma directa,

por ejemplo:

```
<iframe
    src="https://plugins.inforedchile.cl/asocnotarios"
    height="200"
    width="300"
    title="description">
</iframe>
```

2.2. Soluciones existentes

Se realiza un análisis de las empresas que ofrecen un sistema de directorio dentro del país, considerando sus principales características y tarifas asociadas. En un primer momento se presentan a las empresas que prestan sus servicios de directorio a través de libros digitales o planillas Excel:

- **Directorio Nacional de Empresas Chilenas**: Portal que contiene datos sobre más de 6.000 empresas y 20.000 ejecutivos de todo Chile, tanto en el sector privado como el público. Su misión es crear contactos comerciales tipo B2B más efectivos, válidos, óptimos y oportunos a nivel nacional e internacional. Conocido como "Libro Rojo", sus versiones físicas ya no están disponibles y actualmente ofrecen sus servicios bajo las modalidades de software para computadoras, la cual incluye filtros de búsqueda y la impresión de fichas de clientes, con un costo de $89.250 con IVA incluido por la suscripción anual, además de la versión en libro digital, que incluye buscador, salto de página y zoom, pero al igual que la versión en *software* no permite copiar y pegar la información, además de ser intransferible a otro aparato, teniendo un precio de $53.330 con IVA incluido.

- **Base de datos Chile**: Empresa dedicada a la venta de bases de datos de toda índole. Un producto ofrecido es el "Gran libro rojo de directivos y empresas de Chile" con un valor de $89.990, el cual se encuentra digitalizado en Excel, donde se encuentran 25.500 contactos de las principales empresas de Chile y sus respectivos ejecutivos. Otra opción, es el "Directorio de empresas de Chile" con un valor de $576.000, también digitalizado en Excel, que cuenta con 288.000 contactos válidos. En total se ofrecen alrededor de 200 bases de datos de diversas características con precios desde los $25.000 a $1.000.000.

También encontramos dentro del mercado de directorios, empresas que ofrecen directorios en línea, de una forma similar a como lo hace *InforedChile*. Las búsquedas para los usuarios son generalmente de carácter gratuito. Las organizaciones más relevantes se presentan a continuación:

- **Amarillas de Chile**: Directorio comercial, enfocado principalmente en *Pymes*. Es administrado por la empresa "Gurú", la cual también ofrece el servicio de información para celulares "007" y de asesoramiento de marketing digital. En el portal de "Amarillas de Chile" se puede filtrar por palabras claves y lugares, sin ningún costo asociado, encontrado número de teléfono, e-mail, sitio web y/o dirección según cada organización. Además, permite contactar directamente por el portal e incluso está la opción de chatear en línea para algunas empresas. También posee una guía telefónica para algunos sectores de Chile, de forma gratuita en su versión PDF. Para las organizaciones que deseen ser parte de esta plataforma y compartir su información de contacto o publicitarse en el portal y/o en las guías telefónicas, existe un costo asociado, el cual debe ser cotizado directamente con la empresa "Gurú".

- **Directorio Chile**: Portal web destinado a la búsqueda de empresas por cuidad o categoría. En esta plataforma se puede encontrar el nombre, sitio web, teléfono

y dirección para diversas empresas. Inscribir y buscar organizaciones es gratis, pero para insertar el logo, eliminar la publicidad adyacente a la búsqueda y aparecer en los primeros lugares de búsqueda, se debe pagar un monto de $10 dólares norteamericanos mensuales.

- **Directorio de empresas Chile**: Portal web que contiene información de contacto de más de 12.000 empresas registradas gratuitamente por los mismos usuarios. Permite filtrar por localidad, categoría y rubro, además de la comunicación mediante mensajes a través de la misma plataforma. Existe publicidad en el portal, la que presenta un costo de $105.000 con IVA incluido por 3 meses. Se debe pagar una membresía si se desea acceder al listado de las empresas, la cual tiene un precio de $4.500 semanales, $13.500 mensuales, $32.500 trimestrales o $135.000 anuales.

- **Portal web Chile**: Plataforma que hace de vitrina virtual, donde se pueden buscar empresas sin ningún costo asociado, filtrando por categoría y/o ubicación. Para la inscripción de empresas en el portal existen 3 tipos de planes; el plan normal posee un costo de $12.000 e incluye textos descriptivos, contactos, 10 fotografías, fondo musical, soporte técnico; el plan avanzado de $24.000 incluye además servicios de *Google Maps* y *Street*, e inserción de audio radial; por último, el plan dedicado de $36.000 incluye multimedia y eventos, además de soporte 24/7.

Junto a lo anterior, existen instituciones que complementan los servicios de directorio con el de agenda digital, teniendo un mayor número de funcionalidades y, por ende, costos asociados más altos. Dentro de las organizaciones que ofrecen estos servicios a nivel nacional encontramos:

- **Agendapro**: empresa con sede de operaciones tanto en Chile como en Argentina, Brasil, Canadá, Estados Unidos, Reino Unido, entre muchos otros, que se

centra en brindar servicios de agenda online, control de pagos, herramientas de marketing, reporte de gestión comercial, fidelización de clientes, entre otros. Se centra en organizaciones del ámbito de la estética, belleza, salud y bienestar principalmente, aunque también tienen clientes de otros rubros. En lo que respecta al diseño, presenta una plataforma bastante atractiva a ojos de los usuarios.

Presenta 3 planes generales, siendo el primero de ellos el *Básico*, el cual cuenta con un valor de **$39.990** mensuales, que incluye solo servicios de agenda y cuenta con disponibilidad para 5 usuarios. Otro plan, denominado *Premium*, el cual cuenta con servicios de marketing digital y fidelización de cliente, que presenta un valor de **$59.990** mensuales y cuenta con capacidad para 10 usuarios. Finalmente encontramos el plan *Pro*, con un costo mensual de **$79.900**, con disponibilidad para 15 agendas, que además de los servicios de los otros planes, también cuenta con capacidad para personalizar funciones de usuarios y acceso a API, que permite integrar los servicios actuales con los que cuenta la organización cliente. [18]

- **Reservo**: empresa que ofrece un software de agenda digital, contacto y gestión de registros financieros con operaciones en Chile y Latinoamérica. Al igual que *Agendapro*, se especializan en negocios del rubro de la salud, estética, belleza y deporte. Dentro de sus servicios ofrecidos, podemos encontrar agenda online, ficha electrónica personalizada, control de sesiones, confirmación de correos electrónicos, registros financieros, herramientas de fidelización de clientes, cálculo de comisiones, además de reportes y estadísticas del rendimiento del negocio.

  La cotización de los servicios se hace de forma particular, ya que ofrecen una plataforma más personalizada para las necesidades de cada organización y el número de usuarios que requieren. [19]

Siguiendo con las empresas que ofrecen sistemas de agenda digital, encontramos

a las grandes empresas mundiales en este ámbito, presentadas a continuación:

- **G Suite**: Desarrollada por la empresa *Google*, ofrece correo electrónico empresarial, aplicaciones tipo *Office* en línea, servicio de vídeo conferencias, integración de contactos de otras compañías, almacenamiento en nube de datos, calendario, asistencia en línea y servicio de protección de amenazas.

  Cuenta con 3 planes principales para empresas, donde se encuentra el *Business Starter*, que consta de un valor de **$5,4 USD** mensual por usuario con las funciones generales que ofrece el sistema, un plan *Business Standard*, que cuenta con un valor de **$10.8 USD** mensual por usuario, que tiene la ventaja de poseer un mayor espacio de almacenamiento y más capacidad de participantes en vídeo conferencias y por último el plan *Business Plus*, que cuenta con un valor de **$18 USD** mensual por usuario, que posee un espacio de almacenamiento y capacidad de participantes en vídeo conferencias aún mayor que la opción anterior, además de funciones de gestión avanzada.[20]

- **Microsoft Outlook**: De la compañía tecnológica *Microsoft Corporation*, que al igual que la empresa anterior presenta presencia en una gran cantidad de países e integra diversas funciones, tales como correo electrónico, contactos, calendario, tareas, notas, diario y búsqueda en red. Sus principales funciones son crear citas y eventos, organizar reuniones, ver programación de grupos, calendarios en paralelo, superposición de vistas y administración de calendarios de otros usuarios. Además de estas funcionalidades, *Microsoft Corporation* en su paquete *Microsoft 365* ofrece muchas otras funciones, tales como *Microsoft Teams* para videoconferencias con un máximo de 300 personas, el uso en línea de aplicaciones ampliamente conocidas y útiles como *Word*, *Excel* y *Power Point*, además de una nube de almacenamiento de datos llamada *OneDrive*.

  Cuenta con 3 planes principales para empresas, donde se encuentra el *Microsoft 365 Empresa Básico*, que consta de un valor de **$5 USD** mensual con las funcio-

nes generales que ofrece el sistema, el plan *Microsoft 365 Empresa Estándar*, el cual posee un valor de **$12,5 USD** mensual por usuario, que incluye lo mismo que el plan *Básico*, además de herramientas para desarrollar y administrar la empresa, y versiones de escritorio de aplicaciones de *Office* para equipos *PC* y *Mac* y la otra alternativa es *Microsoft 365 Empresa Premium*, que cuenta con un valor de **$20 USD** mensual por usuario, e incluye lo mismo que el plan *Empresa Estándar*, además de protección contra amenazas avanzada, y Administración de PC y dispositivos móviles. [21]

Por otro lado, y en vista de la información encontrada, según las últimas cifras oficiales entregadas por *Microsoft* y *Google*, *Outlook* y *G Suite* cuentan con 400 y 900 millones de usuarios alrededor del mundo respectivamente, haciendo que su cartera de contactos las más nutridas del mundo. Esto es una ventaja sustancial para empresas que buscan ampliar su fronteras.

En vista de lo anterior, se nos presenta una barrera importante para implementar nuestro modelo de negocios en grandes empresas, por lo que debemos buscar rubros más específicos, que utilicen en sus procesos un directorio de contacto más acotado a ciertos rubros en los cuales enfocarnos. De esta forma podemos acaparar y mantener un número adecuado de clientes para hacer conveniente el desarrollo del proyecto.

2.3. Contribuciones

Los servicios mostrados anteriormente están pensados para directorios grandes donde además se consideran servicios de agendas y visualización de datos. En nuestro caso se busca un servicio que sea económico y fácil de utilizar para servicios pequeños que requieren una administración simple.

3. Requisitos del sistema

3.1. Requisitos funcionales

- **RF1** - El sistema debe permitir el registro de nuevas organizaciones.
- **RF2** - El sistema debe permitirle a InforedChile admitir o rechazar los registros de nuevas organizaciones.
- **RF3** - El sistema debe permitir a las organizaciones registradas publicar nuevas instituciones (representadas como avisos) en InforedChile.
- **RF4** - El sistema debe permitir a aquellas organizaciones que actualmente están en la base de datos de InforedChile que gestionen de manera autónoma su información pública existente.
- **RF5** - El sistema debe proveer de un *Plugin/widget* web con el Directorio Digital Interactivo para ser inscrustado en los sitios de las organizaciones, acotado a las instituciones que le pertenecen.
- **RF6** - El Directorio debe contener un buscador de datos limitado a la organización en la que están incrustados.
- **RF7** - Los datos en el directorio deben tener un indicador de estado.
- **RF8** - El sistema debe proveer un método para que los usuarios informen sobre datos con problemas.

- **RF9** - El sistema debe permitirle a InforedChile administrar las organizaciones inscritas.

3.2. Requisitos extra funcionales

- Alojamiento en servidores de InforedChile.
- Uso de base de datos *MySQL* de InforedChile.
- Los buscadores implementados deben ser sencillos y opciones de filtros.
- Opción de subir directorio mediante plantilla.
- Se debe garantizar la separar datos entre instituciones.

3.3. Requisitos de hardware

- Servidor Linux con los recursos suficientes para levantar múltiples instancias de Apache y Nodejs.
- Computadores personales y accesorios de trabajo (mouse, teclado, audífonos).

3.4. Requisitos de Software

- Navegador web compatible con el motor JS V8 de Chrome, actualmente compatible con cualquier navegador basado en Chromium.
- *IDE* para desarrollo en *JavaScript, TypeScript*, idealmente con bibliotecas para el desarrollo en *React.js* y *Node.js*.
- Cliente *SFTP/SSH* para la sincronización de los archivos del proyecto en el ambiente de producción de InforedChile.

3.5. Tecnología a Utilizar

La arquitectura seleccionada es la arquitectura Api Rest donde la base de datos se implementa en MySQL, el servidor BackEnd en Nest.js y el FronEnd en Next.js.

4. Diseño

4.1. Modelo de dominio

Al analizar el contexto donde funcionará el sistema se pueden reconocer 4 actores importantes:

- **InforedChile:** Empresa que plantea el desafío quien se encargará de gestionar y mantener operativo el directorio.
- **Clientes InforedChile** (Organizaciones): Empresa encargada de crear y administrar los datos almacenados en el directorio digital.
- **Instituciones:** Son almacenadas en el directorio y cumplen el rol de avisos en la plataforma de InforedChile.
- **Solicitante:** Persona que busca información en el widget a implementar o en la plataforma de InforedChile.

Con estos actores se genera el modelo de dominio (figura 4.4) presentado a continuación.

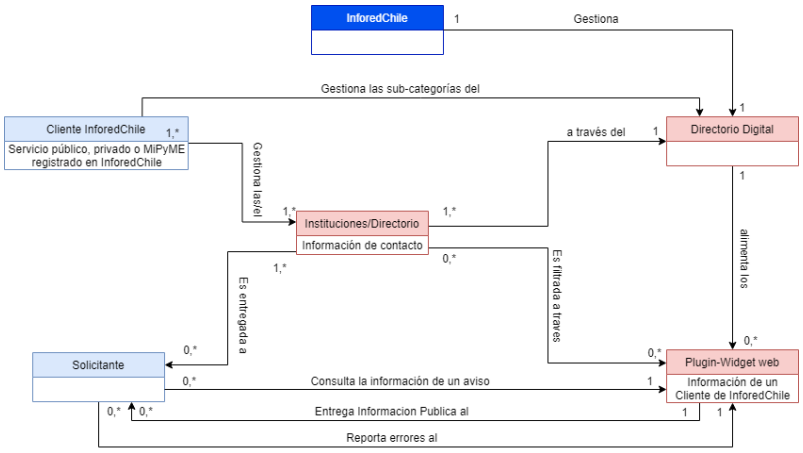

Fig. 4.1: Dominio del sistema

4.2. Modelo de datos

Para poder satisfacer las nuevas necesidades que se van a generar con el sistema en desarrollo, es necesario realizar actualizaciones al actual modelo de datos de InforedChile. Para realizar esta tarea se van a considerar 3 tipos de tablas de datos:

- **Tablas que no se modifican:** Estas tablas no requieren una actualización, por lo que quedan igual al modelo original de datos de InforedChile. Estas tablas son:
  - tbl_ciudad
  - tbl_publico_cate
  - tbl_publico_cate_sub
- **Tablas modificadas:** Estas tablas requieren añadir nuevos parámetros los cuales son importantes para el desarrollo del proyecto. Esta tabla es:
  - tbl_publico_aviso - Se añaden las columnas : organización, plantilla asociada, errores.

- Tablas propuestas: Estas tablas son propuestas por el equipo multidisciplinario, para la implementación de las nuevas funcionalidades a realizar. Estas tablas son:
    - tbl_organizacion:
        - Nombre.
        - Ubicación.
        - Descripción.
        - Correo 1.
        - Correo 2.
        - Telefono 1.
        - Teléfono 2.
        - Contraseña.
        - Categoría Principal.
        - Subcategoría Principal.
    - tbl_personas:
        - Organización
        - Institución.
        - Plantilla Asociada.
        - Nombre.
        - Rut.
        - Depto.
        - Área.
        - SubÁrea.
        - Dirección(Profesional).
        - Correo.
        - Teléfono.

    - Cargo.
    - Visibilidad.
- tbl_plantilla:
    - Organización.
    - Nombre Plantilla.
    - V_ Nombre.
    - V_Rut.
    - V_Depto.
    - V_Área.
    - V_SubÁrea.
    - V_Dirección(Profesional).
    - V_Correo.
    - V_Teléfono.
    - V_Cargo.

Este modelo de datos está basado en las necesidades de la asociación de notarios y conservadores de Chile, por lo que puede ser modificado a posterioridad para poder cumplir de manera general con las necesidades de los servicios públicos.

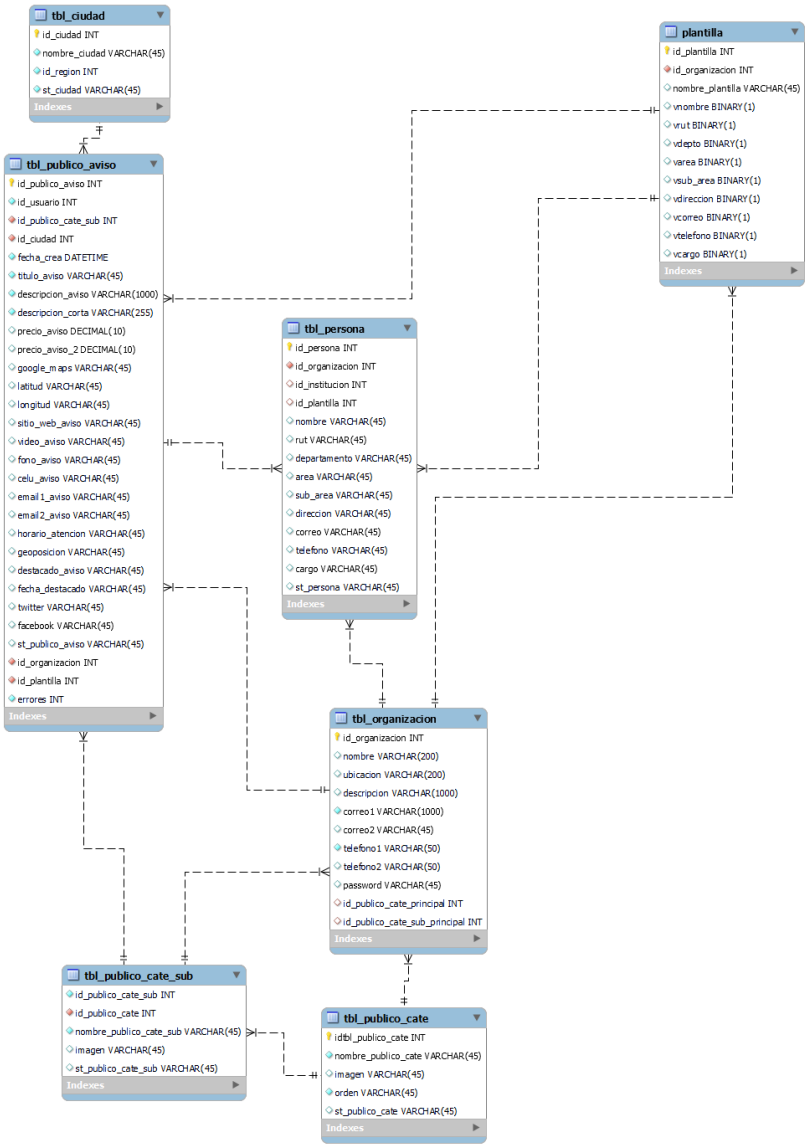

Fig. 4.2: Modelo de datos

4.3. Arquitectura del sistema

La arquitectura a utilizar en este proyecto es *API REST*, como se representa en la figura 4.3, la cual establece la comunicación entre cliente y servidor mediante el uso de métodos *HTTP*. Esto es útil puesto que permite independizar el desarrollo del *Back End* del *Front End*, permitiendo que este último pueda ser levantado en distintas

plataformas manteniendo un solo servidor centralizado.[4]

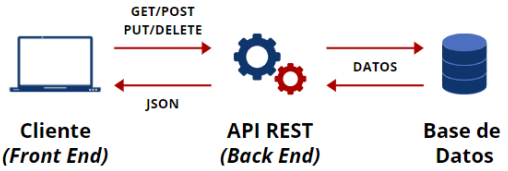

Fig. 4.3: Arquitectura REST

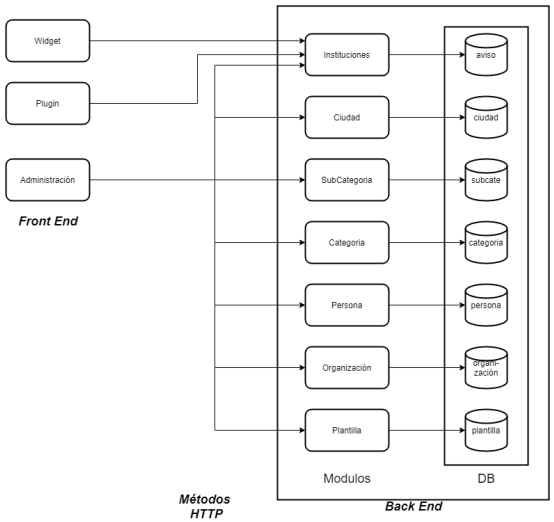

Fig. 4.4: Arquitectura del sistema

4.4. Módulos del Sistema

El *Back End* del sistema debe ser capaz de procesar las peticiones *HTTP* que van a ser realizadas por el *Front End* y también realizar las modificaciones en los datos almacenados.

Para lograr este objetivo el servidor se separa en 7 módulos que se encargan de manera exclusiva de administrar las consultas a su tabla de datos correspondiente. Esto es posible puesto que actualmente cada tabla de la base de datos de InforedChile es

independiente y se relacionan exclusivamente por la identificaciones que, aunque no están marcadas como llaves foráneas, sí cumplen dicha función.

Para establecer la conexión entre cada módulo y su tabla se va a utilizar un *ORM* llamado *TypeORM*, el cual se encarga de realizar la conexión interna.

Para poder describir cada módulo se entrega una breve descripción cuando es necesario, la tabla asociada que tienen y sus *endpoints* principales. Cabe destacar que no se necesita implementar todos los métodos *HTTP* en cada módulo, si no que se implementan de acuerdo a las necesidades de cada módulo.

Dado lo anterior, el modelo utilizado para la definición de cada módulo es *API REST*.

4.4.1. MB1 - Categorías

Este módulo se encarga de administrar las solicitudes hacia la tabla "tbl_ publico_ cate".

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/cat/		Lista de categorías	Entregar la lista de categorías registradas en el sistema.
2	GET	/cat/:id_cat	Id categoría	Categoría	Entrega los datos de la categoría solicitada mediante su id.

Tabla 4.1: Módulo Categorías

4.4.2. MB2 - Subcategorías

Este módulo se encarga de administrar las solicitudes hacia la tabla "tbl_publico_cate_sub".

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/scat/		Lista de categorías	Entregar la lista de subcategorías registradas en el sistema
2	GET	/scat/:id_scat	Id subcategoría	Subcategoría	Entrega los datos de la subcategoría solicitada mediante su id.
3	GET	/scat/cat/:id_cat	Id categoria	Lista de subcategorías	Entrega lista de subcategorías asociadas a la categoría de interés.

Tabla 4.2: Módulo Subcategorías

4.4.3. MB3 - Ciudad

Este módulo se encarga de administrar las solicitudes hacia la tabla "tbl_ciudad".

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/ciudad/		Lista de ciudades	Entrega la lista de ciudades registradas en el sistema
2	GET	/cate/:id_cludad	Id ciudad	Ciudad	Entrega los datos de la ciudad solicitada mediante su id.
3	GET	/ciudad/region /:id_region	Id region	Lista de ciudades	Entrega la lista de ciudades asociadas a una región en concreto.

Tabla 4.3: Módulo Ciudad

4.4.4. MB4 - Institución

Antes de explicar qué función realiza este módulo es necesario tener presente que actualmente no se almacenan instituciones, sino que se almacenan avisos que son los datos principales de contacto hacia ellas. Por esta razón, se considerará que un aviso es una institución almacenando la misma información y añadiendo una columna extra para identificar alguna organización encargada de los datos de esta institución.

Este módulo se encarga de administrar las solicitudes hacia la tabla "tbl_publico_aviso".

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/insti/		Lista de instituciones	Entrega la lista de instituciones registradas en el sistema.
2	GET	/insti/:id_insti	Id institución	Institución	Entrega los datos de la institución solicitada mediante su id.
3	GET	/insti/org/:id_org	Id organización	Lista de instituciones	Entrega una lista de todas las instituciones asociadas a una organización en particular.
4	POST	/insti/	Datos institución	Institución	Añade una nueva institución a la base de datos.
5	PUT	/insti/:id_insti	Id institución, datos editados	Institución	Edita la información de una institución ya registrada.
6	DELETE	/insti/:id_insti	Id institución		Elimina una institución de la base de datos.

Tabla 4.4: Módulo Institución

4.4.5. MB5 - Organización

Una organización es una entidad cuyo rol es administrar la información de un conjunto de instituciones y su información. Por ejemplo, una notaría es una institución y la asociación de notarios es una organización que administra la información de cada notaría. Esto se plantea así puesto que pueden existir instituciones con varias áreas, por ejemplo una municipalidad, con información de contacto propia pero administrada por la municipalidad, entonces se consideraría cada área de la municipalidad como una institución y la municipalidad en sí como una organización puesto que ella administra la información.

Este módulo se encarga de administrar las solicitudes hacia la tabla "tbl_organizacion".

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/org/		Lista de organizaciones	Entrega la lista de organizaciones registradas en el sistema
2	GET	/orgi/:id_org	Id organización	Organización	Entrega los datos de la organización solicitada mediante su id.
3	POST	/org/	Datos organización	Organización	Añade una nueva organización a la base de datos.
4	PUT	/org/:id_org	Id organización, datos editados	Organización	Edita la información de una organización ya registrada.
5	DELETE	/org/:id_org	Id organización		Elimina una organización de la base de datos.

Tabla 4.5: Módulo Organización

4.4.6. MB6 - Personas

El objetivo del módulo de personas es poder almacenar y administrar los datos de contacto de trabajadores de la institución. Estos datos pueden ser públicos o de uso interno por lo que la institución tendrá la labor de tomar dicha decisión.

Como el conjunto de datos de este módulo es un directorio se va a usar la ruta "/dir/" como ruta base.

Este módulo se encarga de administrar las solicitudes hacia la tabla "tbl_persona".

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/dir/		Lista de personas	Entrega la lista de personas registradas en la tabla de directorio.(Función exclusiva de InforedChile)
2	GET	/dir/:id_per	Id persona	Persona	Entrega los datos de la persona solicitada mediante su id.
3	GET	/dir/org/:id_org	Id organización	Lista de personas	Entrega una lista de personas pertenecientes a una organización.
4	GET	/dir/insti/:id_inst	Id institución	Lista de personas	Entrega una lista de personas pertenecientes a una institución.
5	POST	/dir/	Datos persona	Persona	Añade una nueva persona a la base de datos.
6	PUT	/dir/:id_per	Id persona, datos editados	Persona	Edita la información de una persona ya registrada.
7	DELETE	/dir/:id_per	id persona		Elimina a una persona de la base de datos.

Tabla 4.6: Módulo Persona

4.4.7. MB7 - Plantilla

Dado que se busca generar un grado de personalización en los datos a ser mostrados, se le facilitará a las organizaciones el poder generar una plantilla de visualización de datos donde van a poder decir qué datos generales, como por ejemplo nombre, correo, teléfono, deben ser visibles y cuáles datos quieren que se mantengan como datos privados.

Este módulo se encarga de administrar las solicitudes hacia la tabla “tbl_plantilla”.

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	GET	/plantilla/		Lista de plantillas	Entrega la lista de plantillas registradas en la tabla de plantilla.
2	GET	/plantilla/:id_pla	id plantilla	Plantilla	Entrega los datos de la plantilla solicitada mediante su id.
3	POST	/plantilla/	satos persona	Plantilla	Añade una nueva plantilla a la base de datos.
4	PUT	/plantilla/:id_pla	id plantilla, datos editados	Plantilla	Edita la información de una plantilla ya registrada.
5	DELETE	/plantilla/:id_pla	id plantilla		Elimina a una plantilla de la base de datos.

Tabla 4.7: Módulo Plantilla

4.5. Interfaz de Usuario

Aunque no es el foco principal de este trabajo, sí es parte importante del desarrollo del proyecto, por lo que se muestra a continuación el trabajo realizado por el equipo para el diseño de la interfaz del sistema.

4.5.1. Modelo de Navegación

Este modelo permite comprender el viaje del usuario en cada vista de la interfaz de usuario. La navegación en la aplicación se basa principalmente en un modelo de navegación lateral embebida. Los widgets y plugins son de vista única por lo que no cuentan con navegación, más allá de la interacción con ciertos botones.

Navegación Lateral del Dashboard

Este modelo ilustra cómo fluye la interacción entre cada vista del dashboard de administración de Organizaciones. Las líneas azules corresponden a redirecciones por eventos de autenticación.

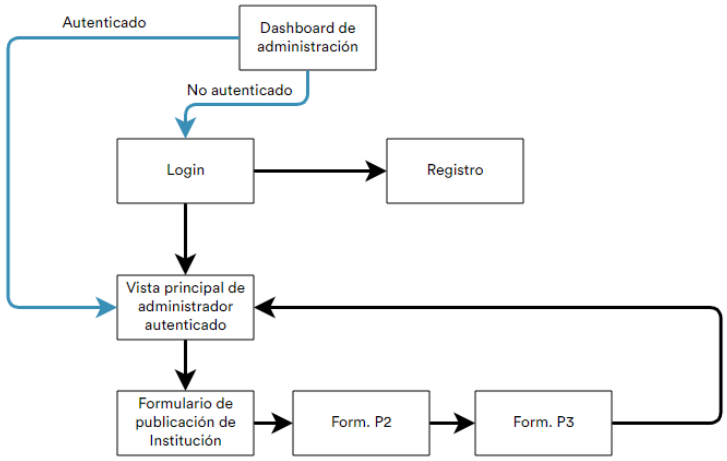

Fig. 4.5: Modelo de Navegación lateral del Dashboard.

4.5.2. Diseño de Interfaces Usuarias

Las interfaces de usuario del sistema se muestran a través de representaciones gráficas *"Mock Ups"*, mostradas a continuación:

Widget Incrustable

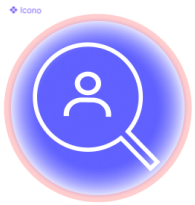

Fig. 4.6: Vista del botón flotante.

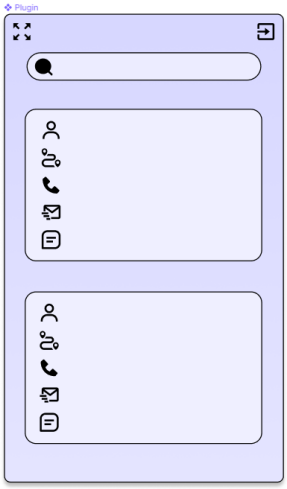

Fig. 4.7: Vista del widget incrustable abierto, no expandido.

Widget abierto y expandido/Vista pública del Directorio

Este mock up representa el widget sobre el sitio web de una institución, particularmente abierto y expandido. También representa la vista general y pública del Directorio Digital Interactivo.

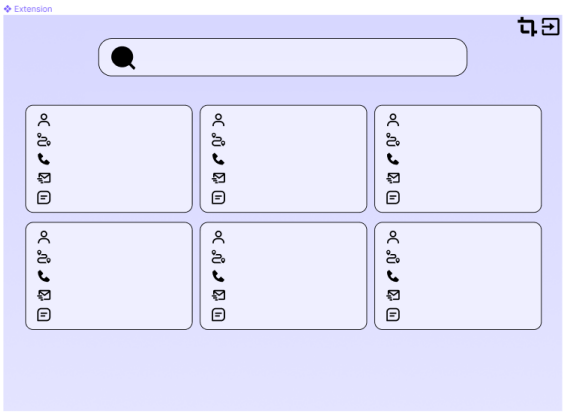

Fig. 4.8: Vista del widget abierto y expandido/Vista del Directorio.

Dashboard de Administración

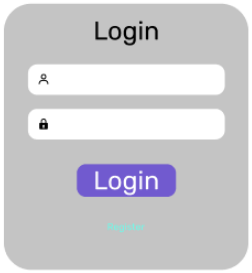

Fig. 4.9: Vista de Login del dashboard de la aplicación.

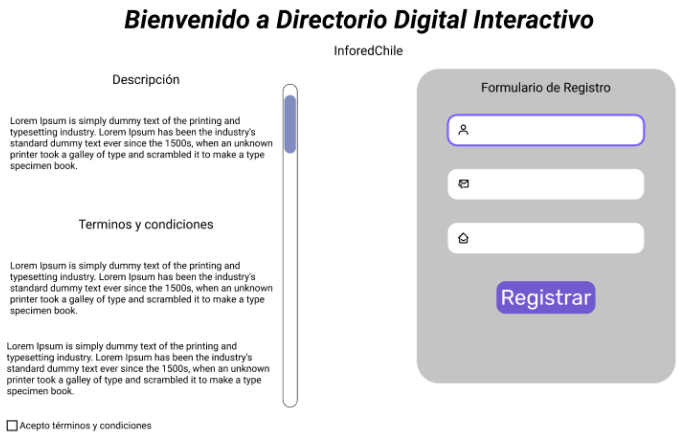

Fig. 4.10: Vista de registro y bienvenida a la aplicación.

A continuación se muestra la vista del perfil de Organización. La interfaz del perfil de InforedChile es semejante.

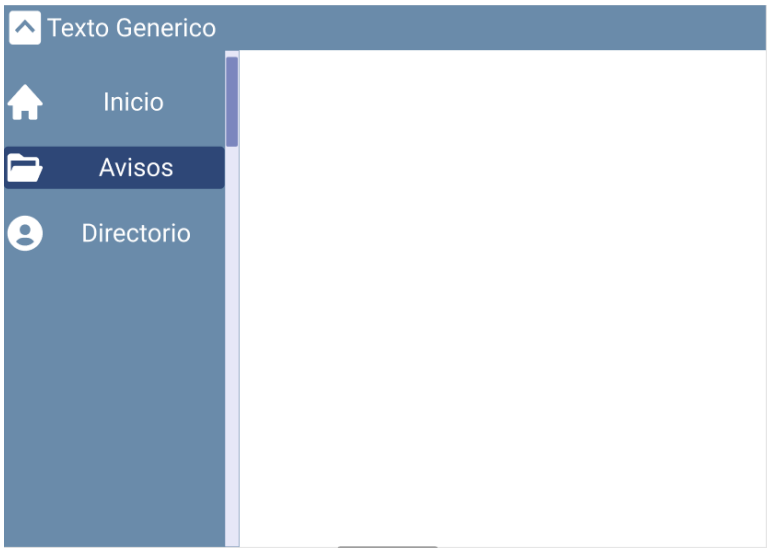

Fig. 4.11: Inicio del Dashboard.

Acá se visualiza en tres etapas el formulario de publicación de una institución al sistema.

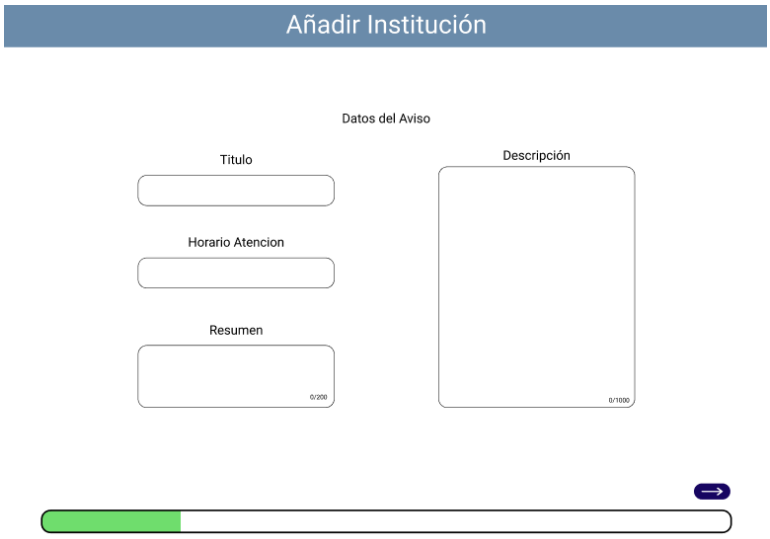

Fig. 4.12: Formulario de publicación de Institución nueva (parte 1).

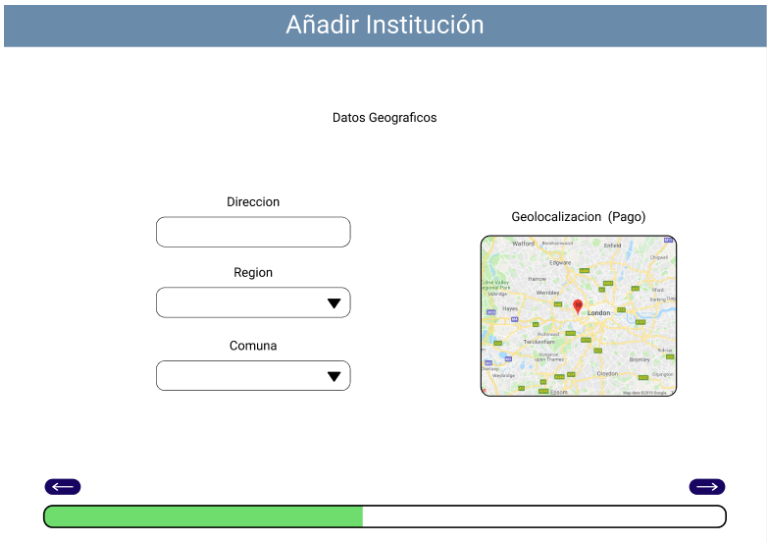

Fig. 4.13: Formulario de publicación de Institución nueva (parte 2).

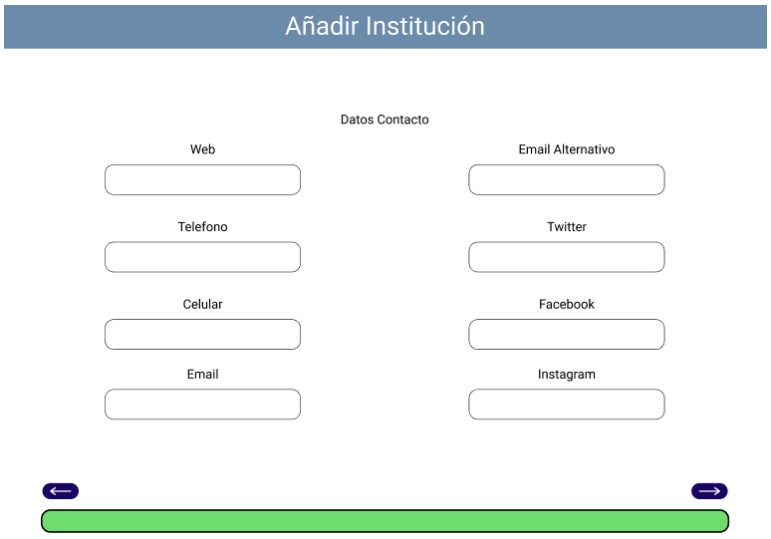

Fig. 4.14: Formulario de publicación de Institución nueva (parte 3).

5. Implementación

El desarrollo técnico del proyecto requiere de múltiples validaciones para comprobar que este cumple con los requisitos, desde el punto de vista funcional. Para esto, se desarrolla un prototipo a ser evaluado. Este prototipo consiste en el levantamiento tanto en ambientes de desarrollo local, como en entornos de prueba que simulan el entorno de producción final. A continuación se definen las condiciones de equipamiento de sistemas y se detallan los procedimientos necesarios para el despliegue de los sistemas de *BackEnd* y *FrontEnd*.

5.1. Entornos

5.1.1. Entornos de Desarrollo

Un entorno de desarrollo es un conjunto de procedimientos y herramientas utilizadas por los desarrolladores para codificar, generar, depurar, actualizar, integrar, testear, validar y ejecutar aplicaciones.

Es el proceso integral de gestión del desarrollo de *software* y funciona como un espacio de trabajo en el que los cambios se implementan en diferentes entornos hasta que se ponen en marcha en la versión real.

Hay diversos paradigmas que definen cada entorno de desarrollo que debiese tener

un equipo u organización. Uno de los más sencillos y el que se ocupa para el desarrollo del proyecto es el que define los siguientes entornos.

5.1.2. Entorno de desarrollo local

Este consiste en acondicionar el equipo personal o laboral de cada desarrollador para que cuente con todas las herramientas necesarias para el funcionamiento local del sistema, pudiendo realizar integraciones locales del sistema y también integraciones entre partes locales y remotas.

En este ambiente no se consideran las posibles diferencias de sistema operativo de cada equipo ni las versiones de cada herramienta de software utilizada.

5.1.3. Entorno de Prueba

Para este proyecto, se define un entorno de prueba para hacer la validación final con las personas que lo van a utilizar y detectar posibles errores de programación, de configuración del entorno o de integración de cada módulo. En esta instancia también pueden surgir nuevas necesidades o sugerencias que se pueden implementar previo al paso a producción.

Se elige la plataforma de aplicaciones en la nube **"Heroku"** para el despliegue tanto de la base de datos de prueba, como para la API. Ambos módulos del sistema van muy ligados y esta plataforma cumple perfectamente con las necesidades de este entorno de desarrollo, además de ser una de las soluciones más utilizadas para proyectos de baja escala por su bajo costo y su facilidad de implementación.

Para su configuración, es necesario tener instalado Heroku CLI en un equipo don-

de se encuentre el entorno de desarrollo local, el cual permite replicar el proyecto a la plataforma en la nube, generando así el entorno de pruebas. Las instrucciones de instalación para los sistemas operativos Linux, Windows y macOS se encuentran en la documentación de Heroku en su sitio web. [27]

El ambiente de pruebas de la interfaz gráfica se decide desplegar en la plataforma de aplicaciones en la nube **"Vercel"**, dado que la empresa del mismo nombre es la propietaria y desarrolladora de el *Framework Next.js* donde está desarrollado el *Front End* del sistema.

La característica principal de estas plataformas es que no se requiere configurar el sistema operativo ni el entorno de ejecución. Sin embargo, están basadas en las versiones LTS de Linux y Node.js, lo que atiende tanto el requisito no funcional del cliente como la restricción de funcionamiento definida por el equipo para el funcionamiento del sistema, por lo que la correcta ejecución de este entorno de desarrollo es un buen precedente para el levantamiento a producción.

5.1.4. Entorno de Producción

Uno de los aspectos más importantes cuando se finaliza el desarrollo inicial de una aplicación, es dejar el sistema accesible a los usuarios objetivo, lo que implica el despliegue en un entorno de producción. En esta etapa, el sistema ya es funcional a nivel general. Desde este punto, el sistema pasa a su etapa de mejoramiento y mantención, donde el equipo de desarrollo actual u otro que continúe con el proyecto, se deben encargar de velar por mantener el sistema funcionando, corregir los posibles errores no detectados previamente que puedan presentar y realizar las mejoras futuras planificadas y por planificar.

Es necesario considerar previamente las capacidades técnicas que tiene el servidor donde se realice el despliegue, así como sus costos de funcionamiento y de mantención, dado que este debe mantenerse operativo sin importar la demanda de recursos que pueda generar el flujo de usuarios.

Durante la etapa de planificación de este proyecto, se definió que el ambiente de producción sería alojado en el servidor de InforedChile, integrado inclusive a su base de datos. Esto no se ha descartado, sin embargo el alcance del proyecto cubre hasta un prototipo funcional levantado en el entorno de pruebas.

5.2. Base de datos

Como recordatorio, uno de los requisitos para el sistema es el uso de la base de datos de InforedChile, cuyo modelo de datos se dispuso a ser modificado para atender las necesidades del proyecto, que resultan en la creación de nuevas tablas y añadir nuevas columnas a las existentes. El resultado de esto es la propuesta del nuevo modelo de datos extendido de InforedChile, definido en la sección 4.2.

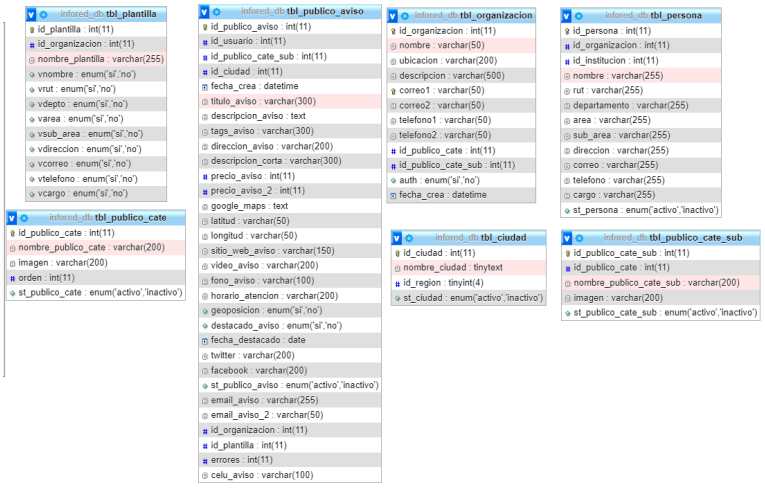

Fig. 5.1: Implementación Modelo de datos en MySQL.

5.2.1. Despliegue de la base de datos en el entorno de pruebas

Para realizar la integración completa del sistema en ambiente de pruebas, es necesario contar con una instancia completamente operativa de una base de datos *MySQL* con el nuevo modelo de datos y algunos datos de prueba que provengan de la base de datos original de InforedChile, con tal de corroborar tempranamente la compatibilidad de estos datos, a pesar de las modificaciones pertinentes. Estos datos son provistos por el cliente a través de una semilla ".sql" y gracias a *TypeORM*, se pueden reconstruir fácilmente en la base de datos del ambiente de pruebas.

Configuración de la plataforma Heroku

Heroku ofrece *PostgreSQL* como base de datos por defecto para la configuración de proyectos, lo que dificulta el despliegue de bases de datos como *MySQL*, que es la utilizada en este proyecto. Por ello, se utiliza la extensión *ClearDB* de Heroku, que se ofrece como alternativa a *PostgreSQL* y permite el usar *MySQL*. La extensión se instala a través de *Heroku CLI* en el servicio de *Back End* del proyecto.

Para poder levantar la base de datos se ejecuta el siguiente comando de Heroku CLI en el directorio del proyecto a través del terminal:

```
heroku addons:create cleardb:ignite
```

Esto genera una nueva extensión que aloja una base de datos de *MySQL*.

Una vez generada la extensión, con el comando

```
heroku config $|$ grep CLEARDB\_DATABASE\_URL
```

se obtiene la ruta de la base de datos, la cual entrega información relevante como: nombre de usuario, contraseña, ruta y nombre de la base de datos.

Finalmente, a través de herramientas de administración de bases de datos, como PHPMyAdmin, se pueden añadir los datos de prueba de InforedChile en el archivo semilla ".sql". Gracias al modo de sincronización, activado por defecto al configurar de forma adecuada la instancia de *TypeORM*, la propagación de la semilla ocurre de forma automática.

5.3. Módulos

5.3.1. Ciudad

Este módulo se encarga de administrar las consultas que se quieran realizar a la tabla ciudad. En la figura 5.2 se presentan las rutas implementadas para este módulo.

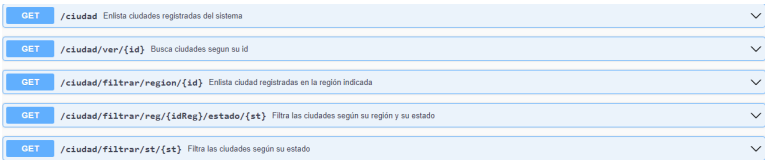

Fig. 5.2: Rutas Módulo Ciudad.

No se implementan métodos POST, PUT y DELETE puesto que para las consideraciones de este proyecto no son relevantes.

5.3.2. Organización

Este módulo se encarga de administrar las consultas que se quieran realizar a la tabla organización, la cual es propuesta por el equipo. En la figura 5.3 se presentan las rutas implementadas para este módulo.

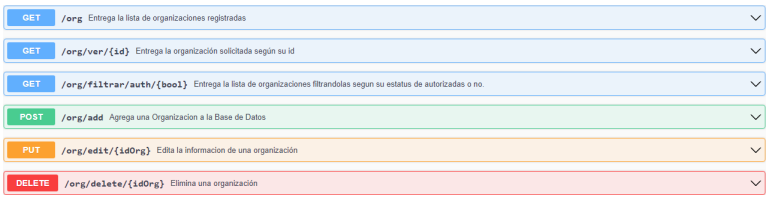

Fig. 5.3: Rutas Módulo Organización.

5.3.3. Institución

Como se indica en el capitulo 4.4 este módulo se encarga de administrar las consultas que se quieran realizar a la tabla de avisos a la que se le proponen cambios que son necesarios para el correcto funcionamiento del sistema. En la figura 5.4 se presentan las rutas implementadas para este módulo.

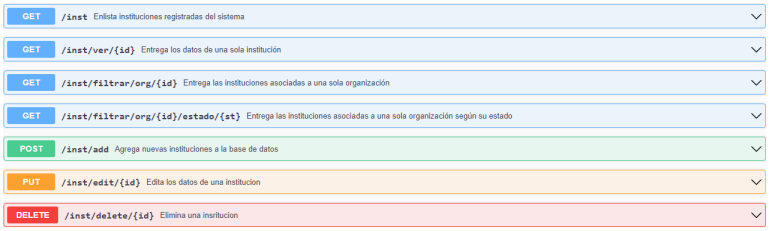

Fig. 5.4: Rutas Módulo Institución.

5.3.4. Categoría

Este módulo se encarga de administrar las consultas que se quieran realizar a la tabla categorías. En la figura 5.5 se presentan las rutas implementadas para este módulo.

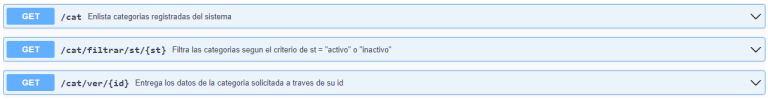

Fig. 5.5: Rutas Módulo Categoría.

No se implementan métodos POST, PUT y DELETE, según lo dispuso el cliente, dado que estas son fijas.

5.3.5. Subcategoría

Este módulo se encarga de administrar las consultas que se quieran realizar a la tabla subcategorías. En la figura 5.6 se presentan las rutas implementadas para este módulo.

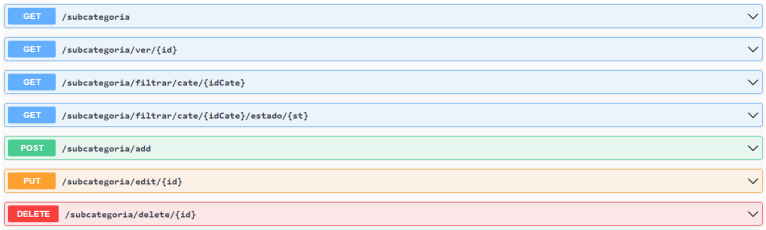

Fig. 5.6: Rutas Módulo Subcategoría.

En este módulo se considera que las organizaciones pueden proponer subcategorías por lo que se implementan los metodos POST, PUT y DELETE.

5.3.6. Plantilla

Este módulo se encarga de administrar las consultas que se quieran realizar a la tabla plantilla, la cual es propuesta por el equipo. En la figura 5.7 se presentan las rutas implementadas para este módulo.

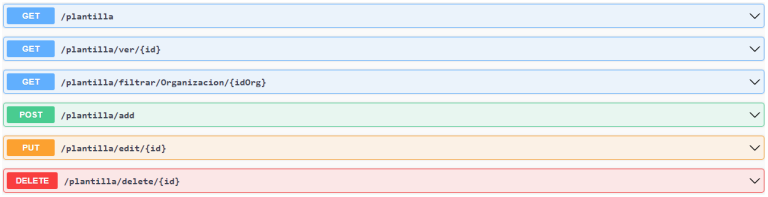

Fig. 5.7: Rutas Módulo Plantilla.

5.3.7. Persona

Este módulo se encarga de administrar las consultas que se quieran realizar a la tabla persona, la cual es propuesta por el equipo. En la figura 5.8 se presentan las rutas implementadas para este módulo.

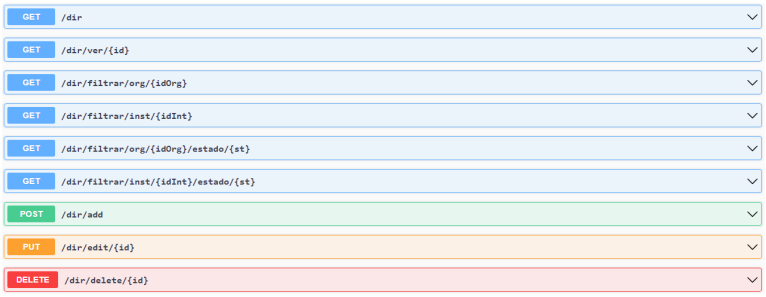

Fig. 5.8: Rutas Módulo Persona.

5.4. Despliegue de la API en el entorno de pruebas.

Se decide también utilizar la plataforma *Heroku* para levantar el ambiente de prueba, lo que permite probar funcionamiento y corregir errores de manera segura sin afectar los servicios de InforedChile que se encuentran en funcionamiento.

5.4.1. Configuración de la plataforma Heroku

Para desplegar el proyecto en el ambiente de prueba es necesario crear un nuevo proyecto a través de la plataforma web de Heroku. Luego se sube el proyecto a Heroku git en la rama *main*. Además, es necesario tener un tener un archivo de configuración del proyecto llamado **Procfile** en el que debe indicarse el comando a ejecutar para que el sistema inicie en producción. Esto se hace agregando la línea “web: npm run start:prod”.

5.5. Interfaz Gráfica

La aplicación web implementada consiste en una interfaz gráfica integrada al sistema de *back end* y es la puerta de entrada principal que se le entrega al usuario final.

La implementación del *Front End* se realizó de la siguiente forma:

5.5.1. Formulario de Registro de Organizaciones

Este formulario, le permite a una organización que no participa activamente en InforedChile, enviar una solicitud para acceder al sistema, la cual puede ser visualizada en el **Perfil de InforedChile** del Dashboard de Administración. Al recibirlo, InforedChile establece el contacto con la organización para la firma de contratos y una vez el proceso legal está terminado, la institución es aprobada en el sistema y las credenciales de acceso ingresadas en el formularios son activadas para ser utilizadas como **Perfil de Organización**.

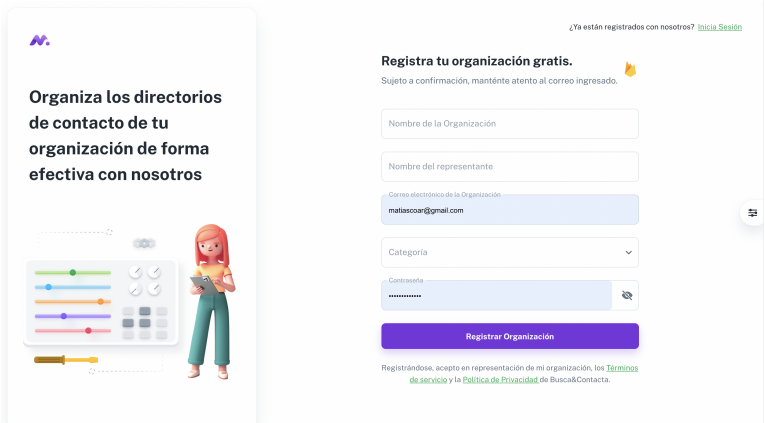

Fig. 5.9: Vista Prototipo Formulario de Registro.

5.5.2. Login

El acceso al sistema, ya sea para los usuarios de InforedChile, como para las Organizaciones, requiere de credenciales de acceso. Estas deben ser ingresadas en el login del sistema. A excepción del formulario de registro, todas las vistas están protegidas y sólo son visibles con una sesión iniciada. Cualquier otro intento de acceso será redirigido al Login.

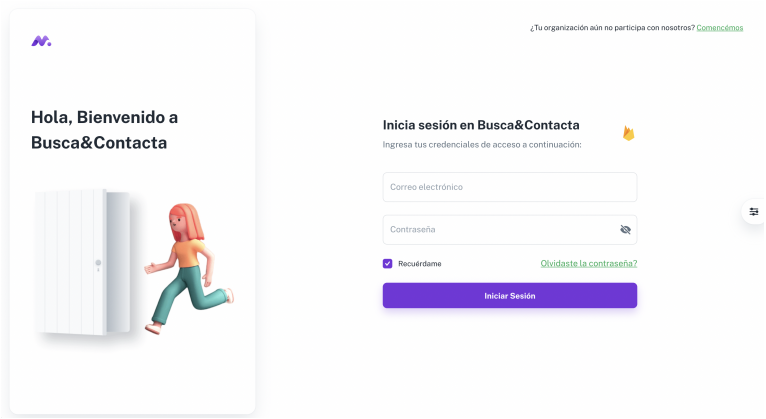

Fig. 5.10: Vista Prototipo Login.

5.5.3. Dashboard de Administración

El dashboard de administración es el motor base del sistema. Tanto los usuarios pertenecientes a InforedChile como a las Organizaciones participantes, acceden a esta parte del sistema. Este además genera las vistas públicas que alimentarán los *widgets* y *plugins* para incrustar en los sitios de las organizaciones.

El perfil de Organizaciones es la sección del dashboard dedicada a la administración de las instituciones que le pertenecen a una organización. Cabe recordar que en la base de datos de InforedChile, estas instituciones se traducen en Avisos.

La implementación del **Dashboard de Administración, perfil InforedChile** es bastante similar a ésta por lo que no es prioridad de este hito.

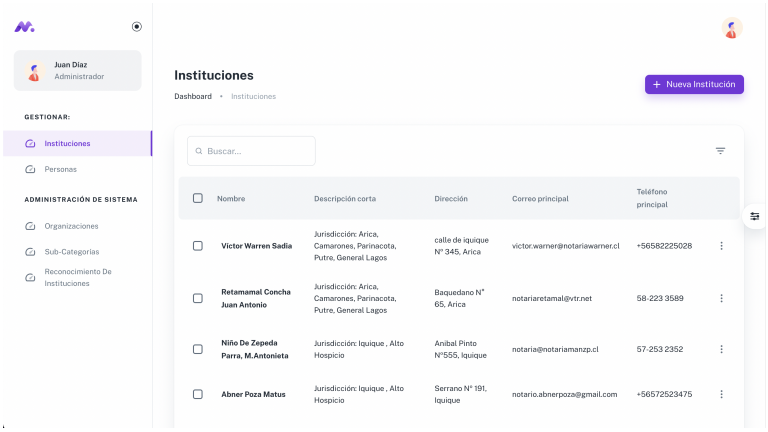

Fig. 5.11: Listado Instituciones, Dashboard Organizaciones.

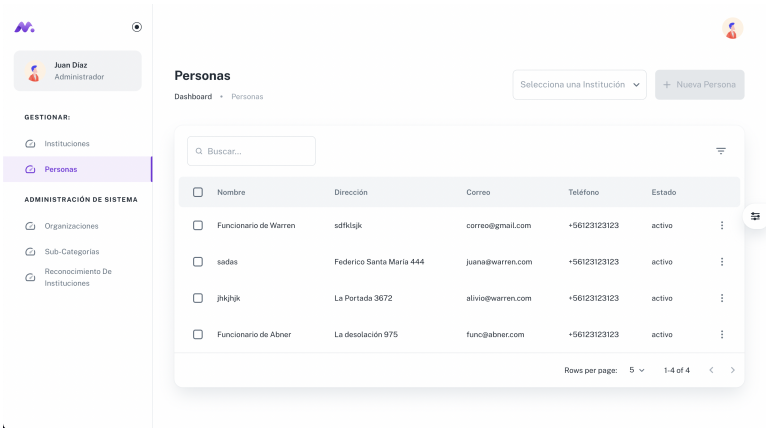

Fig. 5.12: Listado Personas, Dashboard Organizaciones.

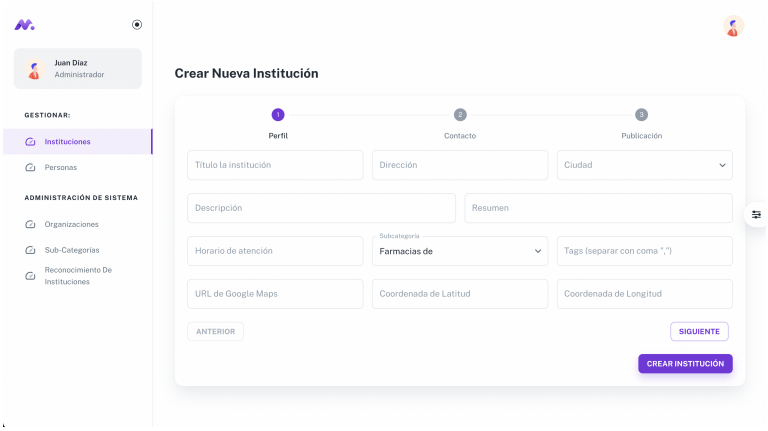

Fig. 5.13: Formulario de creación de nueva institución (1/3).

El formulario de edición de instituciones es el mismo, rellenando todos los campos con los datos existentes en la base de datos. Hay un ejemplo más abajo en el formulario de personas que funciona de la misma forma.

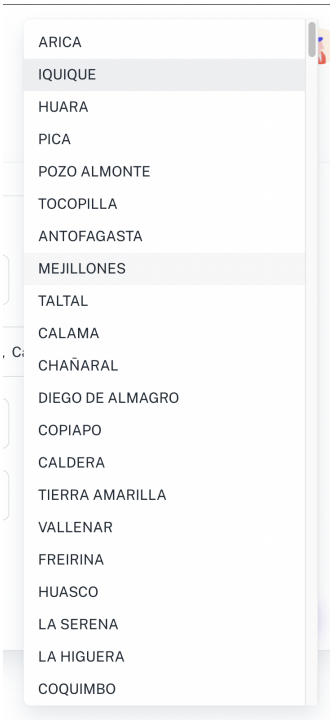

Fig. 5.14: Lista de comunas, formulario de creación de nueva institución.

Listado de comunas habilitadas para crear instituciones, generado a partir de la base de datos.

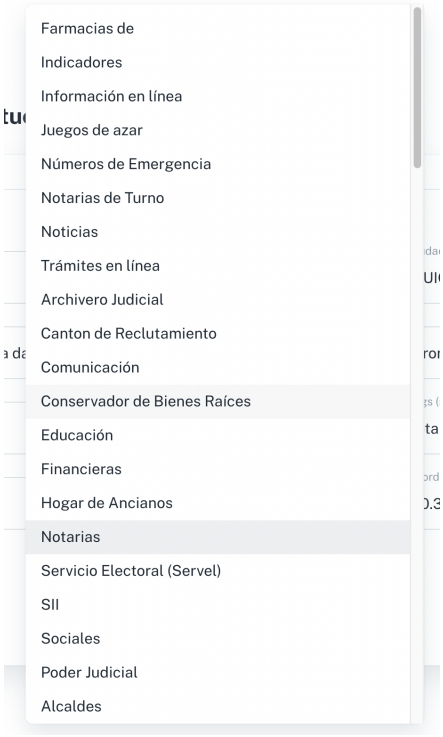

Fig. 5.15: Lista de subcategorías, formulario de creación de nueva institución.

Listado de las subcategorías habilitadas para generar instituciones, obtenido desde la base de datos.

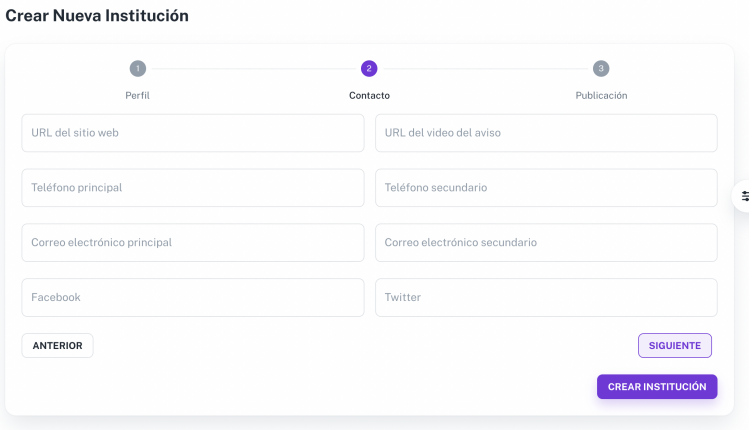

Fig. 5.16: Formulario de creación de nueva institución (2/3).

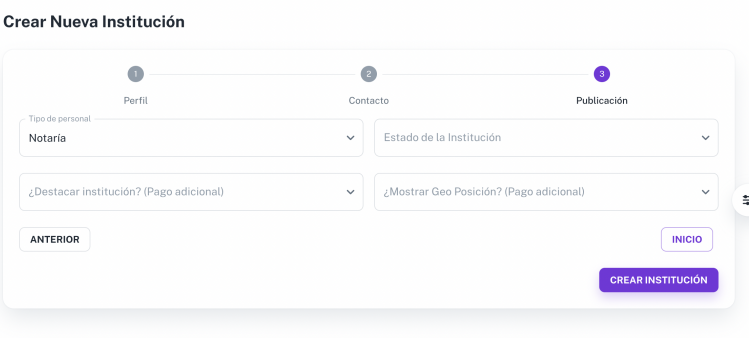

Fig. 5.17: Formulario de creación de nueva institución (3/3).

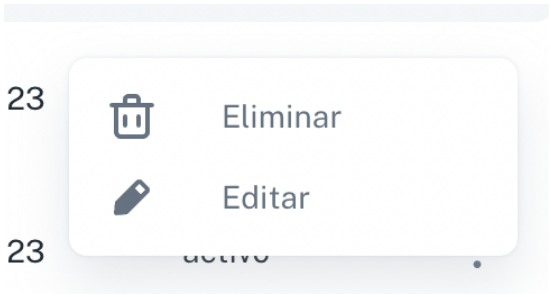

Fig. 5.18: Menú editar entrada (Instituciones y Personas).

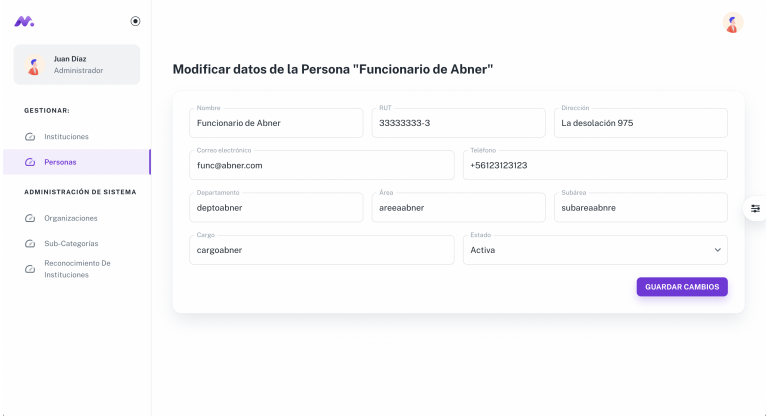

Fig. 5.19: Formulario de modificación de persona.

La creación de personas consiste en el mismo formulario en blanco.

6. Validaciones

6.1. Pruebas de funcionamiento

Para validar el correcto funcionamiento del servidor *BackEnd*, es necesario evaluar el comportamiento del sistema ante solicitudes *HTTP* realizadas a través de sus distintos *EndPoint*. Esto se debe realizar en todos los *EndPoint* implementados para cada módulo, comparando el comportamiento ante distintos tipos de entradas de datos.

Cabe destacar que dependiendo del módulo, se seleccionaron los *EndPoint* más relevantes para realizar las validaciones.

A nivel de *FrontEnd* estas pruebas, aunque son relevantes, se evalúan al analizar los casos de uso del sistema donde se integran las pruebas de funcionamiento con el comportamiento de un usuario.

6.1.1. Ciudad

En este módulo los *EndPoint* relevantes son:

- Filtrar por región.
- Filtrar por estado.
- Filtrar por región y estado.

Observación: Región se considera el identificador de la región en la que se encuentra la ciudad, el cual está definido en Chile mediante números, y el estado indica si la ciudad se considera activa o inactiva.

Filtrado por Región

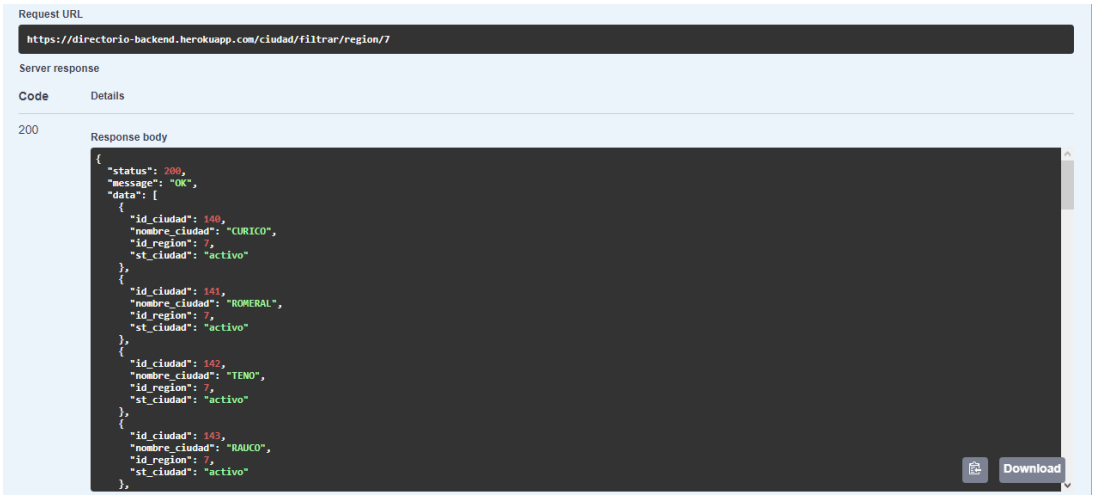

Fig. 6.1: Filtrado de ciudades por región.

En la figura 6.1 se observa que el sistema entrega la lista de ciudades de la región solicitada.

Filtrado por Estado

En la figura 6.2 se observa que el sistema entrega la lista de ciudades con el estado indicado.

El estado de una ciudad puede ser activo o inactivo.

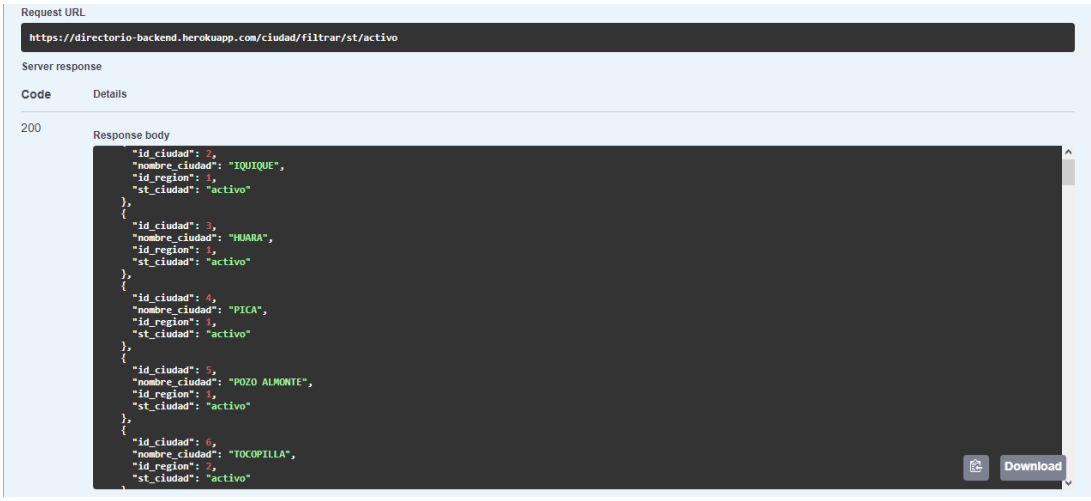

Fig. 6.2: Filtrado de ciudades por estado.

Filtrado por Región y Estado

Fig. 6.3: Filtrado de ciudades por región.

En la figura 6.3 se observa que el sistema entrega la lista de ciudades de la región con el estado solicitado. En este caso, es sólo una ciudad.

6.1.2. Organización

En este módulo los *EndPoint* relevantes son:

- Obtener lista de Organizaciones.
- Añadir una Organización.

- Editar una Organización.

Obtener lista de Organizaciones

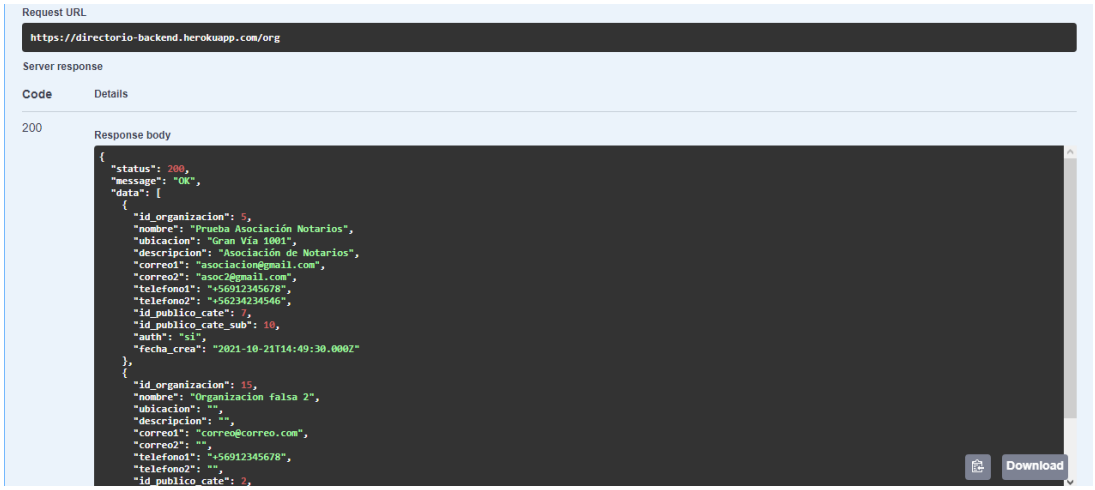

Fig. 6.4: Lista de organizaciones registradas.

Como se observa en la figura 6.4 el sistema entrega la lista de organizaciones registradas en el sistema.

Añadir una Organización

Fig. 6.5: Organización añadida.

```
{
  "id_organizacion": 25,
  "nombre": "Organizacion falsa 4",
  "ubicacion": "",
  "descripcion": "",
  "correo1": "correo3@correo.com",
  "correo2": "",
  "telefono1": "+56912345672",
  "telefono2": "",
  "id_publico_cate": 4,
  "id_publico_cate_sub": 12,
  "auth": "no",
  "fecha_crea": "2021-12-21T19:48:32.000Z"
}
```


Fig. 6.6: Comprobación de Organización añadida.

Como se observa en la figura 6.5, efectivamente es posible añadir organizaciones en el sistema, lo que se comprobó al revisar la figura 6.6.

Editar una Organización

Fig. 6.7: Organización añadida.

Como se observa en la figura 6.5, efectivamente es posible editar organizaciones en el sistema, lo que se comprobó al revisar la figura 6.6. Esto es importante dado que

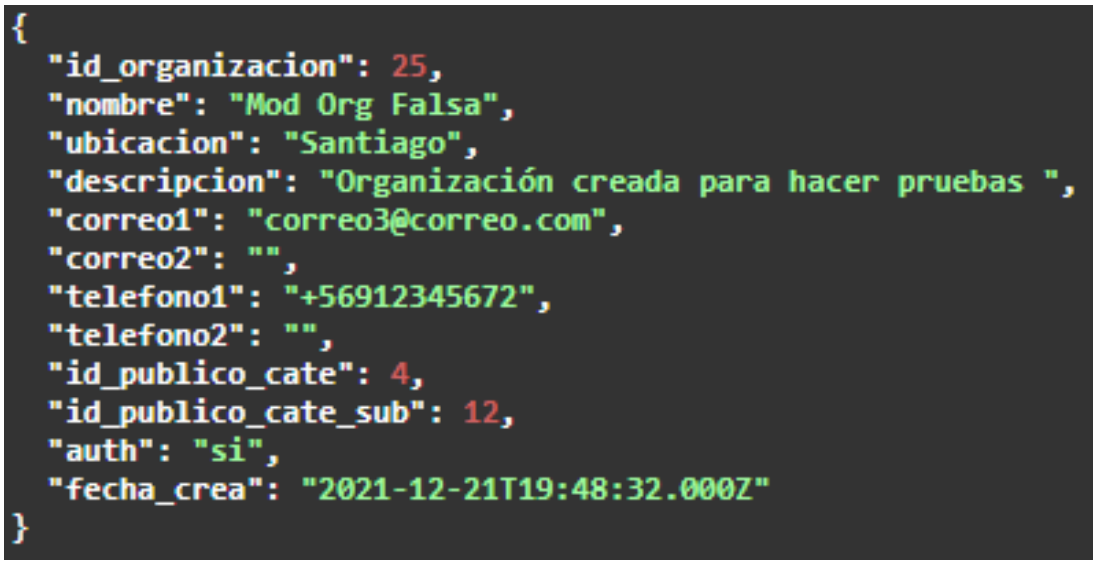

Fig. 6.8: Comprobación de Organización editada.

las organizaciones requieren ser autorizadas por InforedChile antes de poder utilizar el sistema.

6.1.3. Categoría

En este módulo los *EndPoint* relevantes son:

- Obtener lista de categorías.
- Filtrar categorías.

Obtener lista de Categorías

Filtrar Categorías según su estado

Como se observa en la figura 6.9 las categorías pueden estar activas o inactivas, por lo que esta función filtra según su estado para mostrarle al usuario sólo las que puede utilizar. La Figura 6.10 muestra categorías inactivas.

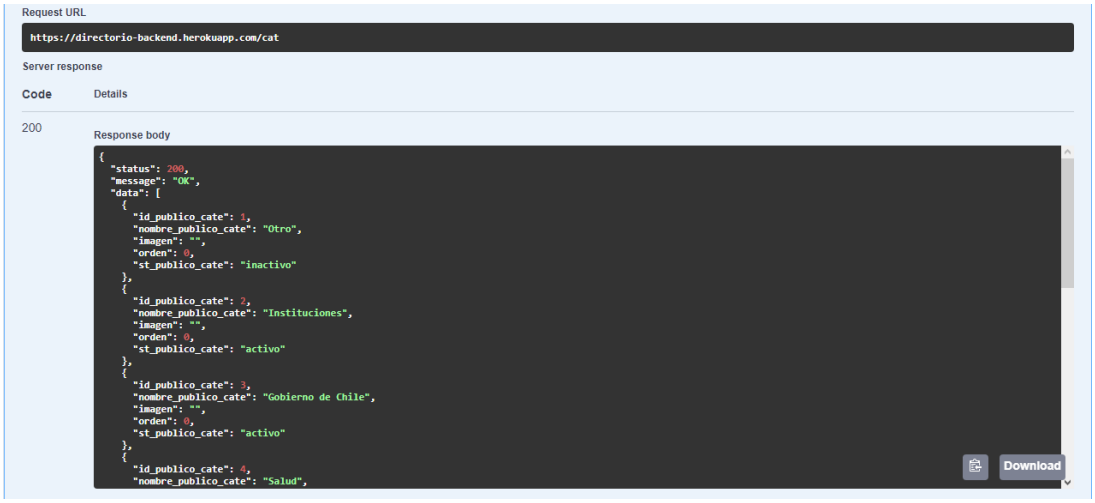

Fig. 6.9: Lista de Categorías.

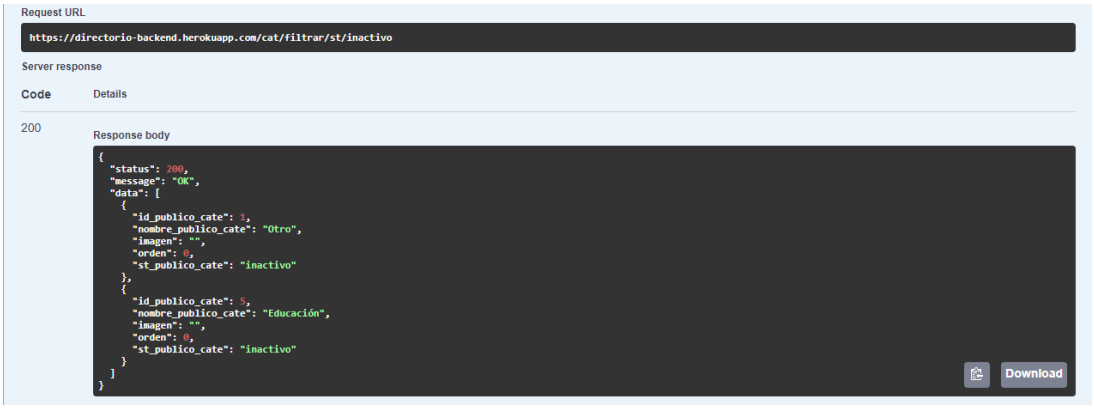

Fig. 6.10: Filtro de Categorías.

6.1.4. Subcategoría

En este módulo los *EndPoint* relevantes son:

- Obtener lista de Subcategorías.
- Filtrar Subcategorías según su categoría.

Observación: Aunque están implementados los *EndPoint* de añadir, editar y eliminar Subcategorías, no se consideran relevantes en este caso dado que no fueron implementados en conjunto con la aplicación web.

Obtener lista de Subcategorías

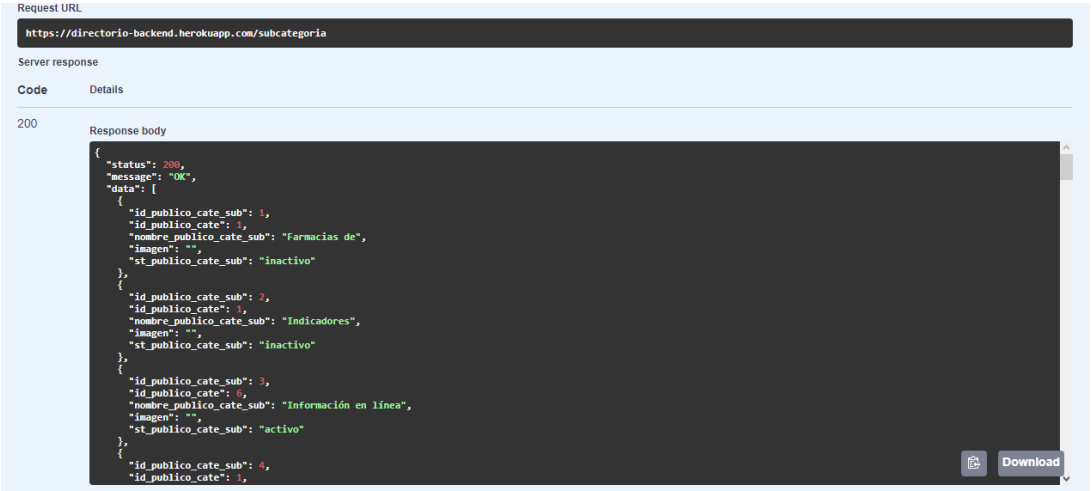

Fig. 6.11: Lista de Subcategorías.

Filtrar Subcategorías según Categorías

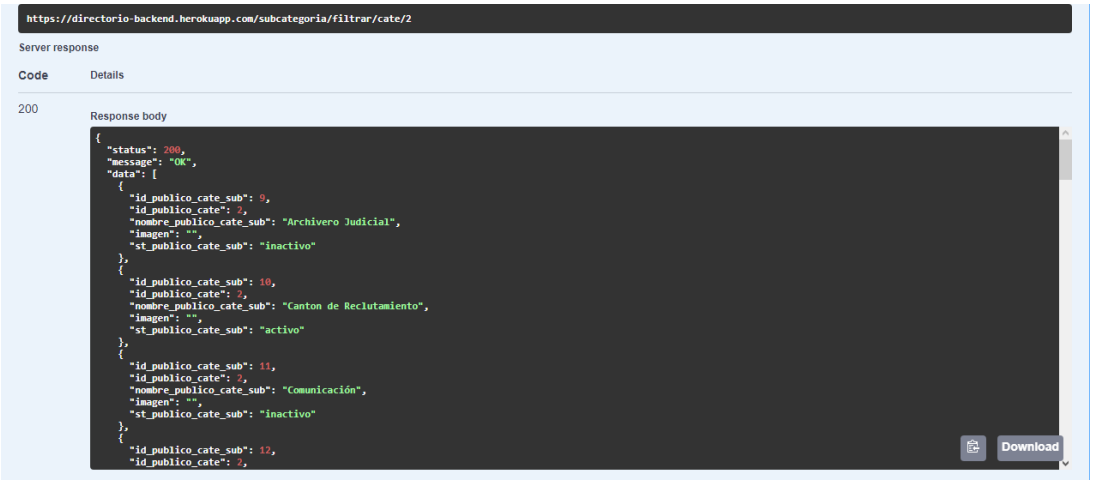

Fig. 6.12: Filtro de Categorías.

Como se observa en la figura 6.11 existen muchas subcategorías en el sistema las cuales están clasificadas según su categoría, por lo que esta función filtra según la categoría asociada para facilitar la selección. La figura 6.12 muestra las subcategorías

asociadas a la categoría 2.

6.1.5. Institución

En este módulo los *EndPoint* relevantes son:

- Filtrar instituciones según su rganización.
- Añadir una institución.
- Editar una institución.
- Eliminar una institución.

Filtrar instituciones según su Organización

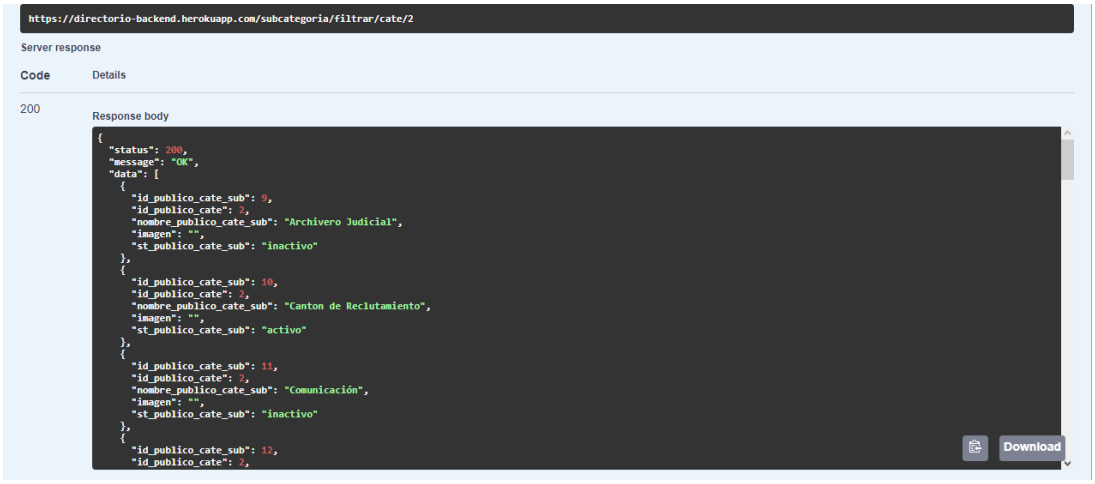

Fig. 6.13: Filtro de Instituciones según su Organización.

En la figura 6.13 se observan las instituciones asociadas a la Organización 5 quien se encargará de mantener sus datos de contacto actualizados.

Fig. 6.14: Error al añadir una Institución.

Añadir una Institución

En la figura 6.14 se observa cómo el sistema, al detectar un dato necesario faltante, responde con un mensaje de error.

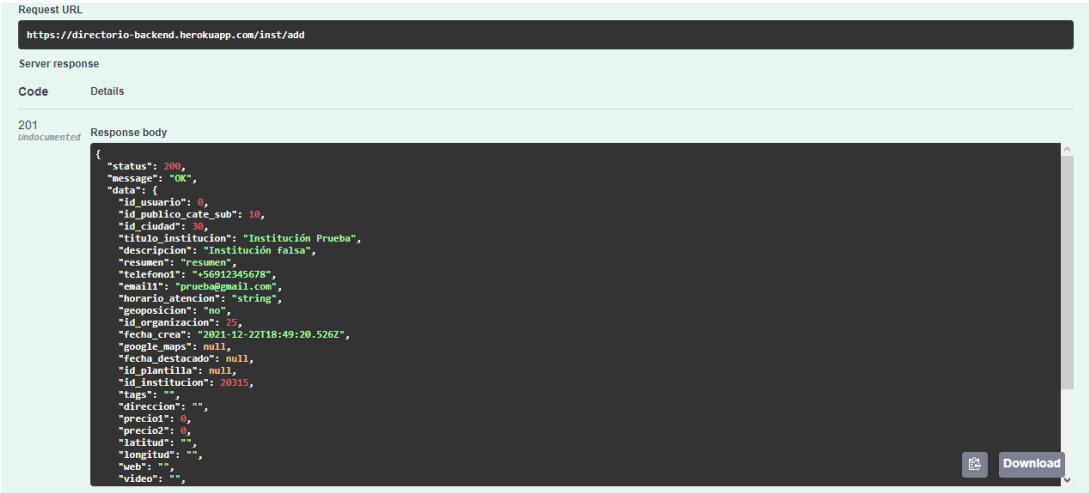

Fig. 6.15: Institución añadida correctamente.

Como se puede observar en la figura 6.15, la institución se crea correctamente, resultado que se contrasta con la figura 6.16.

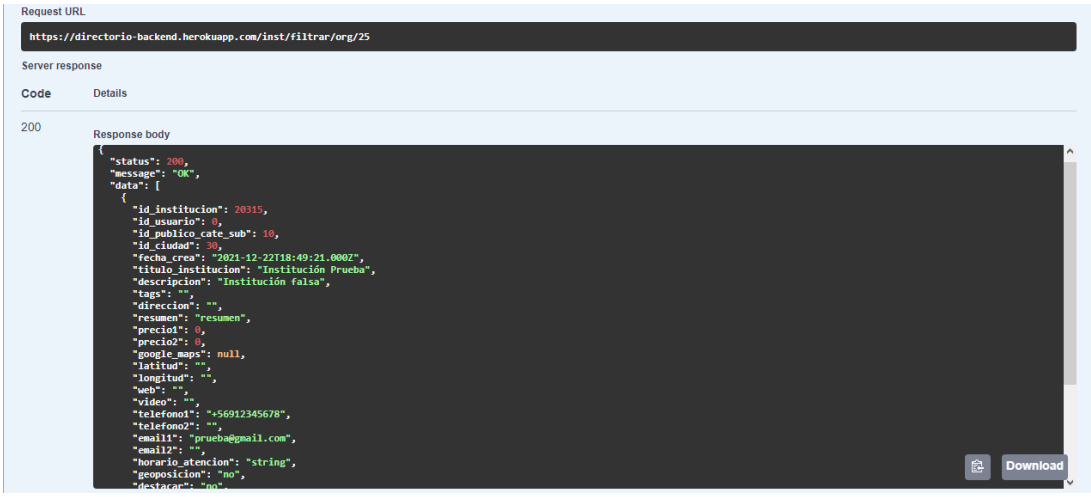

Fig. 6.16: Prueba Institución añadida.

Fig. 6.17: Editar una Institución.

Editar una Institución

Como se puede observar en la figura 6.17 se pueden editar los datos correctamente. Además, si se notifica un número grande de errores como en este caso, el estado de la institución pasa a ser inactivo.

Fig. 6.18: Eliminar una Institución.

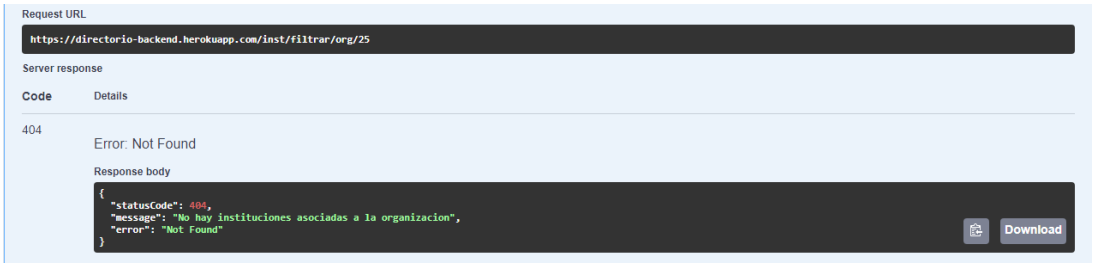

Fig. 6.19: Prueba Institución eliminada.

Eliminar una Institución

En la figura 6.18 se genera la solicitud para eliminar una institución, solicitud que se prueba que funciona en la figura 6.19.

6.1.6. Plantilla

Observación: Aunque se implementó en el BackEnd el módulo, no se alcanzó a conectar con la aplicación web por lo que no se realizarán pruebas de funcionamiento a este.

6.1.7. Persona

En este módulo los *EndPoint* relevantes son:

- Filtrar personas según su institución.
- Añadir una persona.

- Editar una persona.
- Eliminar una persona.

Filtrar Personas según su Institución

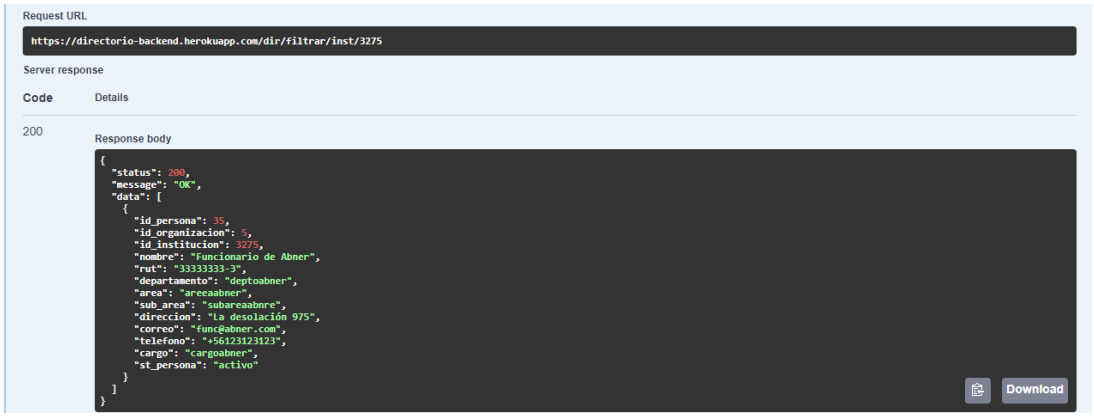

Fig. 6.20: Filtro de Personas según su Institución.

En la figura 6.20 se observa las personas asociadas a la institución 3275.

Añadir una Persona

Fig. 6.21: Añadir una Persona.

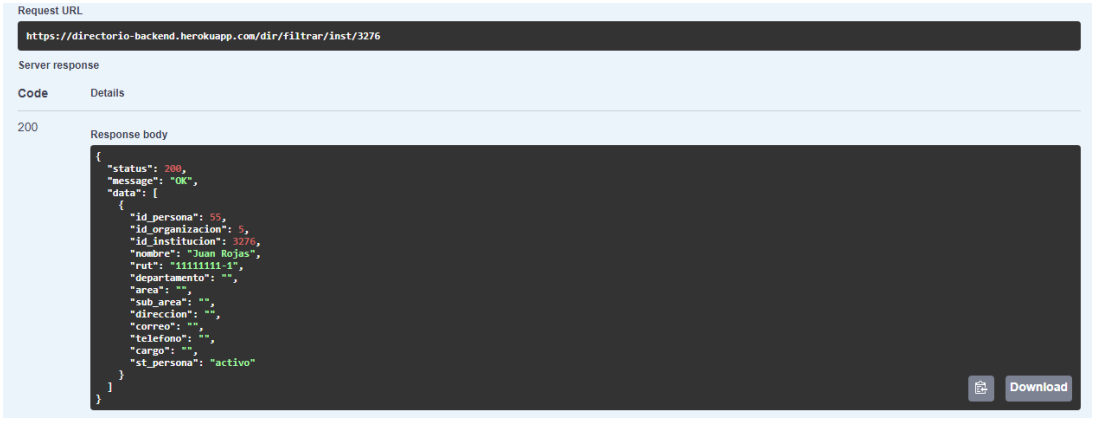

Fig. 6.22: Prueba Persona Añadida.

En la figura 6.21 se agrega una persona a la institución 3276, lo que se comprueba en la figura 6.22.

Editar una Persona

Fig. 6.23: Editar Persona.

Como se observa en la figura 6.23, al editar los datos de una persona, estos son actualizados en la base de datos.

Fig. 6.24: Eliminar una Persona.

Fig. 6.25: Prueba Persona Eliminada.

Eliminar una Persona

En la figura 6.24 se elimina una persona a la institución 3276, lo que se comprueba en la figura 6.25.

6.2. Evaluación casos de usos

Para evaluar los casos de usos, se les solicitó a cuatro conocidos que simularán ser un usuario de la plataforma simulando el uso que tendría. Para esto se les entregó una lista de tareas a realizar sin indicarles el cómo deben hacerlo.

Aunque los casos de uso se evalúan respecto al FrontEnd, para este caso se va a extrapolar a partes del BackEnd que son útiles para el usuario.

6.2.1. Crear una Institución

Objetivo

El administrador de una organización debe ser capaz de crear instituciones. Para esto, el sistema debe entregar un formulario y realizar una validación de los datos ingresados para evitar errores.

Resultado

El usuario logra crear de manera correcta una institución, el servidor *BackEnd* revisa la información ingresada y devuelve una respuesta indicando si se puedo agregar la institución y en caso contrario indica cuál fue el error.

El sistema cumple con lo esperado ya que advierte al usuario en caso de algún error y crea la institución si todo está correcto.

El usuario indica lo sencillo que es crear instituciones y que en caso de equivocarse las ayudas del sistema les facilitan el realizar la corrección lo que cumple con las heurísticas de Nielsen.

6.2.2. Editar una Institución

Objetivo

El administrador de una organización debe ser capaz de editar los datos de una institución como por ejemplo los datos de contacto.

Resultado

El formulario de edición es el mismo que el de añadir institución sólo que con pequeñas modificaciones y una función necesaria que es entregar información precargada. Esto es importante dado que el usuario puede elegir qué modificar sabiendo lo que había previamente.

El sistema cumple de manera correcta, pero si ocurre un problema de compatibilidad de formatos, esto el usuario lo puede solucionar fácilmente reingresando los datos. Sin embargo, esto ocasionó pequeñas molestias a los usuarios aunque no son relevantes para ellos.

6.2.3. Buscar Institución

Objetivo

Un usuario debe ser capaz de buscar una institución a través del *widget* implementado.

Resultado

A través del *BackEnd* se pueden obtener las instituciones pertenecientes a una organización. Esto el usuario lo visualiza mediante el *widget*, el cual utilizando el patrón de diseño buscar y navegar, hace que las personas puedan acceder de manera sencilla a la información que necesitan.

6.2.4. Añadir una Persona

Objetivo

Un administrador debe ser capaz de añadir una persona a una institución para generar un directorio de personas.

Resultado

Añadir una persona es similar a añadir una institución, con la diferencia de que para que se despliegue el formulario hay que primero seleccionar esta última. Esto para el usuario en un principio resultó confuso, pero después de un tiempo, indicó que era bastante cómodo y útil, ya que podía revisar el directorio antes de añadir una nueva persona.

6.2.5. Editar una Persona

Objetivo

Un administrador debe ser capaz de poder editar datos de una persona como por ejemplo datos de contacto.

Resultado

Editar una persona se realiza de manera similar que editar una institución. El *BackEnd* cumple el mismo objetivo y los resultados son similares.

6.2.6. Buscar Personas

Objetivo

Un administrador debe ser capaz de buscar una persona dentro del directorio de personas.

Resultado

Este caso se implementó mediante el patrón de buscar y navegar lo que a través de la barra de búsqueda facilita mucho esta labor.

6.3. Cumplimiento de requisitos

El *BackEnd* al ser el encargado de establecer la comunicación entre la aplicación web y la base de datos no puede cumplir por definición con los requisitos de interfaz.

Los requisitos funcionales se lograron cumplir en su totalidad implementando *EndPoints* necesarios para la administración completa de los datos que se almacenan en el sistema.

No se pudo cumplir con todos los requisitos extrafuncionales, quedando pendientes tareas como: alojar el sistema en los servidores de InforedChile y el poder subir información mediante plantillas. Estos requisitos son entonces parte del trabajo futuro del proyecto.

Se logra comprobar que mantener centralizado los datos ayuda a que el usuario pueda mantener actualizado de forma eficiente sus datos de contacto.

7. Conclusiones

Almacenar de manera centralizada los datos efectivamente permite administrar de forma eficiente la información de contacto de las distintas instituciones. Esto, aunque soluciona el problema principal, no soluciona un conflicto de fondo más profundo que es hacer que el usuario mantenga actualizado los datos.

Este problema igual se facilita si el sistema es cómodo de utilizar, lo que no se ve desde el punto de vista del *BackEnd*.

Respecto a los *EndPoint* implementados, se puede concluir que las funcionalidades realizadas son útiles para los usuarios ya que en conjunto con la interfaz implementada motivan a estos a querer seguir usando la aplicación.

7.1. Trabajo a futuro

Uno de los principales problemas del sistema, es la seguridad al momento de transmitir la información por lo que queda como trabajo futuro implementar encriptación y manejo de sesiones para la comunicación. Esto, aunque no parece importante dado que sólo se almacenan datos públicos, sí es relevante para aquellas empresas que deseen que no sea visible toda la información de sus trabajadores.

Otro aspecto a implementar en un futuro es el levantamiento a producción, lo que implicaría realizar ajustes al modelo de datos implementado y modificar la base de datos actual de la empresa.

Bibliografía

[1] **Alonso Martínez, M.**, 1992. *"Conocimiento y Bases de Datos: una propuesta de integración inteligente"*. Universidad de Cantabria.

[2] **Contreras rebolledo, O.**, 2020. *"Comparación de rendimiento entre una base de datos relacional y una columnar, para el diseño de un data warehouse para Cumplo Chile s.a."*. Memoria para optar al título de Ingeniero Civil en Informática. Universidad Técnica Federico Santa María.

[3] **E-commerce en el país salta 234 % en 5 años: ¿Cómo será el próximo quinquenio y cuánto gastan online los chilenos?**. (2021, marzo). *Emol*. https://www.emol.com/noticias/Economia/2021/03/01/1013410/Ecommerce-Chile.html

[4] **Definición de la API REST: ¿Qué es una API REST (API RESTful)?** (2020, enero). *Astera*. https://www.astera.com/es/type/blog/rest-api-definition/

[5] **Node.js Alternatives**(s. f.).*EDUCBA* https://www.educba.com/node-dot-js-alternatives/

[6] **A World Without Node.js?**(2019, julio).*Medium* https://medium.com/techinpieces/a-world-without-node-js-12fec4b18733

[7] **Node.js vs Springboot Java -Which one to choose and when?**(2020, marzo).*Chapter247*https://www.chapter247.com/blog/node-js-vs-springboot-java-which-one-to-choose-and-when/

[8] **API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos**(2016, marzo).*BBVA API Market*https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/

[9] **Top Node.js Frameworks to use in 2021**(2021, marzo).*JavaScript in Plain English*https://javascript.plainenglish.io/top-node-js-frameworks-to-use-in-2021-4951ee5940b8

[10] **When to use these Nodejs frameworks? Express, Koa, Nest, Socket.io, Meteor.js**(2020, agosto).*Tejas Kaneriya*https://dev.to/tejaskaneriya/when-to-use-these-nodejs-frameworks-express-koa-nest-socket-io-meteor-js-3p63

[11] **Meteor vs NestJS**(s. f.).*Stackshare*https://stackshare.io/stackups/meteor-vs-nestjs

[12] **Angular & Nest, a match made in heaven**(2020, diciembre).*Kevin Kreuzer*https://kevinkreuzer.medium.com/angular-nest-a-match-made-in-heaven-e52cb8e4105a

[13] **Angular 2 & Meteor 1.3 — Friends or Foes?**(2016, mayo).*Tally Barak*https://tally-b.medium.com/angular-2-meteor-1-3-friend-or-foes-5de2498927a0

[14] **SQL (TypeORM)**(s. f.).*NestJs*https://docs.nestjs.com/recipes/sql-typeorm

[15] **Angular vs React vs Vue 2021**(2021, enero).*Aris Pattakos* https://athemes.com/guides/angular-vs-react-vs-vue/

[16] **React vs Angular: Which to choose for your project?**(2019, julio).*Michał Skóra* https://www.futuremind.com/blog/react-vs-angular-which-choose-your-project

[17] **Delivering Web & Mobile Solutions**.*Ackveo*https://www.akveo.com/?utm_source=akveo.github.io

[18] **Agendapro**. Agendapro. Recuperado 7 de junio de 2021, de https://agendapro.com/cl

[19] **Reservo**. Reservo. Recuperado 7 de junio de 2021, de https://reservo.cl/

[20] **G Suite**. *Google Workspace*. Recuperado 7 de junio de 2021, de https://workspace.google.com/intl/es-419/

[21] **Introducción al Calendario de Outlook**. *Microsoft*. Recuperado 7 de junio de 2021, de https://support.microsoft.com/es-es/office/introducci %C3 %B3n-al-calendario-de-outlook-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8

[22] **¿Qué es la Ley de Transparencia?**. *Consejo de la transparencia*. Recuperado 7 de junio de 2021, de https://www.consejotransparencia.cl/inicio_old/que-es-la-ley-de-transparencia/

[23] **InforedChile SpA**. (2021). *Términos y condiciones generales de la adquisición de productos o utilización de los servicios ofrecidos por InforedChile en el sitio web*. https://inforedchile.cl/img/terminos.pdf

[24] **NestJs**. *Swagger Documentation* https://docs.nestjs.com/openapi/introduction

[25] **MDN WEB Docs** *Códigos de estado de respuesta HTTP* https://developer.mozilla.org/es/docs/Web/HTTP/Status

[26] **EKON** *Entornos de desarrollo* https://www.ekon.es/entornos-desarrollo-software/

[27] **DevCenter Heroku** *ClearDB MySQL* https://devcenter.heroku.com/articles/cleardb