Documentación 42LABS

Documentación Técnica

Toma de Control de la plataforma Control de Acceso (CDA) de Megacentro

Empresa: 42LABS
Version: 1.0

13 de Noviembre de 2023

Equipo de Desarrollo:

Nombre	Rol	Contacto
Carlos Arias	Desarrollador Full Stack	carlosarias@42labs.cl

Contraparte

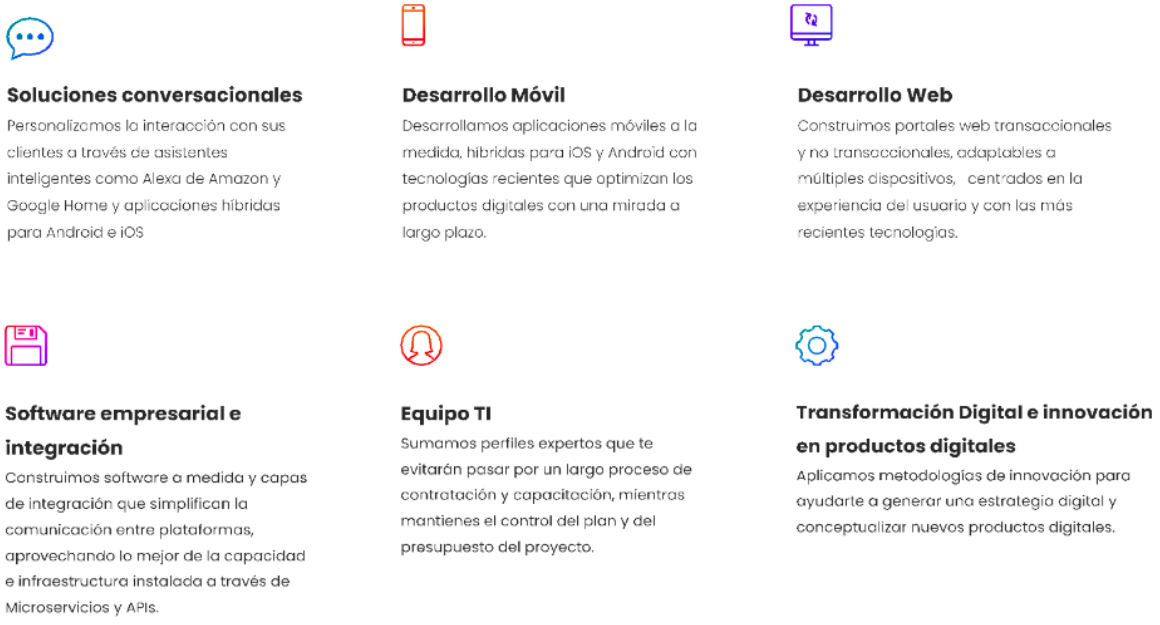
Nombre	Rol	Contacto
Oscar Cantillana	Arquitecto de Sistemas	oscar.cantillana@redmc.com

Índice

1. Sobre 42Labs

42Labs es una empresa de tecnología que nace el 2019 como respuesta a la necesidad crecientes de desarrollo de soluciones y servicios tecnológicos que se ajustaran a los tiempos y requerimientos de negocios cada vez más exigentes.

Actualmente ofrecemos:



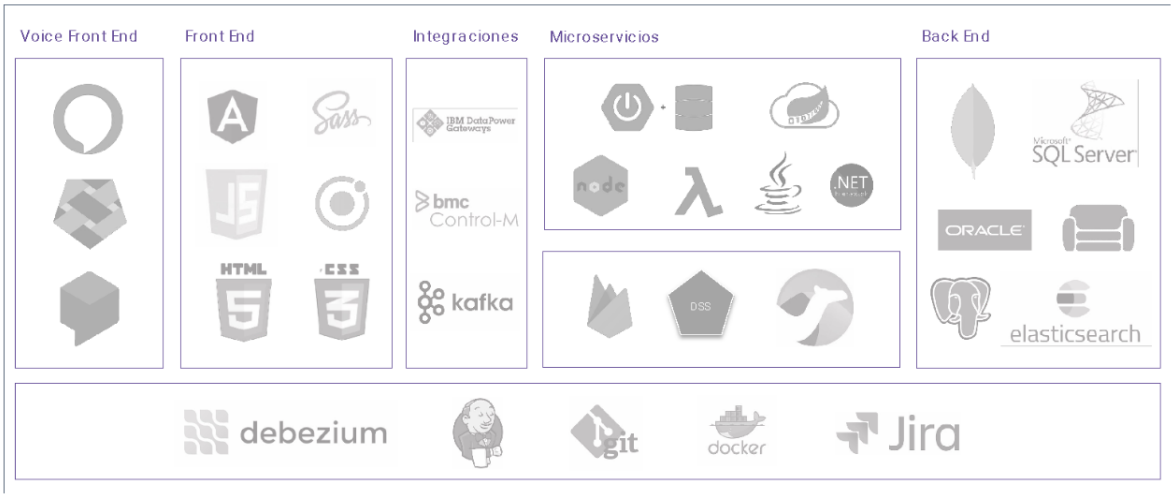
Nuestro equipo ejecutivo tiene más de 15 años de experiencia en la industria financieras y de servicios en Chile y Latinoamérica, tiempo en el que hemos desarrollado con éxito proyectos de fusión bancaria, implementación de plataformas web, sistemas de pago, proyectos de transformación digital y aplicaciones móviles.

Nos movemos rápido y nos movemos bien, contamos con profesionales certificados en metodologías ágiles y entregamos siempre en tiempo y forma. Usamos nuestra creatividad para dar "la milla extra" que permita maximizar el valor que agregamos al negocio. Las dificultades nos fortalecen y las resolvemos colaborativamente, no nos detenemos.

Han confiado en nosotros:

Equipo directivo, C-Level

Tecnologías utilizadas en nuestros los proyectos



2. Sobre el documento

El presente documento tiene como objetivo proporcionar un análisis detallado y una documentación exhaustiva de la API Operaciones API, la base de datos, y el frontend asociado. Este análisis está enfocado en comprender en profundidad la estructura, los componentes y las integraciones de estos sistemas, para facilitar su gestión, mantenimiento evolutivo y correctivo.

El análisis abarcará la API Operaciones API desarrollada con .NET Framework 4.7.2, detallando su interacción con dos bases de datos fundamentales: una en SQL Server llamada MEGACDA y otra en PostgreSQL. Además, se explorarán los aspectos clave del frontend y cómo este interactúa con la API y las bases de datos para realizar operaciones diversas, incluyendo transacciones y manejo de datos.

El alcance de este documento también incluye la presentación de la configuración y los métodos utilizados para la conexión a las bases de datos, así como la descripción de los procesos y las funcionalidades clave que ofrece la API. Este análisis permitirá obtener un entendimiento claro y detallado de la plataforma, necesario para tomar control de la misma de manera eficaz y realizar ajustes o mejoras según se requieran.

Este documento servirá como una guía técnica para los desarrolladores, administradores de sistemas y cualquier otro personal técnico involucrado en la gestión y el mantenimiento de la plataforma.

3. Tareas Ejecutadas

A continuación se detallan las tareas realizadas para la realización de este documento.

3.1. Recolección de Información Inicial

- Se llevó a cabo una reunión inicial con los responsables técnicos y el equipo interno para obtener una comprensión general de la plataforma y sus componentes.
- Se recopilaron las credenciales y detalles de acceso necesarios para los servidores, bases de datos y sistemas externos que forman parte de la plataforma.

3.2. Inmersión en Componentes, Sistemas e Integraciones

- Se realizó un acceso detallado a los distintos servidores y sistemas para comprender a fondo los componentes de la plataforma.
- Se examinó el frontend, incluyendo su interacción con la API Operaciones API, y cómo esta última conecta y gestiona las bases de datos SQL Server y PostgreSQL.

3.3. Análisis de Bases de Datos

- Se analizó la estructura de las bases de datos, identificando tablas clave, relaciones y la manera en que se almacenan y gestionan los datos.
- Se destacó la importancia de la base de datos MEGACDA en SQL Server para la gestión centralizada de la información y su conexión con las bases de datos PostgreSQL de cada centro.

3.4. Revisión y Documentación del Código Fuente

- Se revisó el código fuente de la API, identificando patrones de diseño, dependencias y cómo se manejan las solicitudes y respuestas.
- Se documentaron las funciones clave, servicios y la estructura general del código para facilitar el mantenimiento y futuras actualizaciones.

3.5. Evaluación de Seguridad y Rendimiento

- Se evaluaron las prácticas de seguridad implementadas en la API y en la base de datos.
- Se realizaron pruebas de rendimiento para identificar posibles cuellos de botella y áreas de mejora.

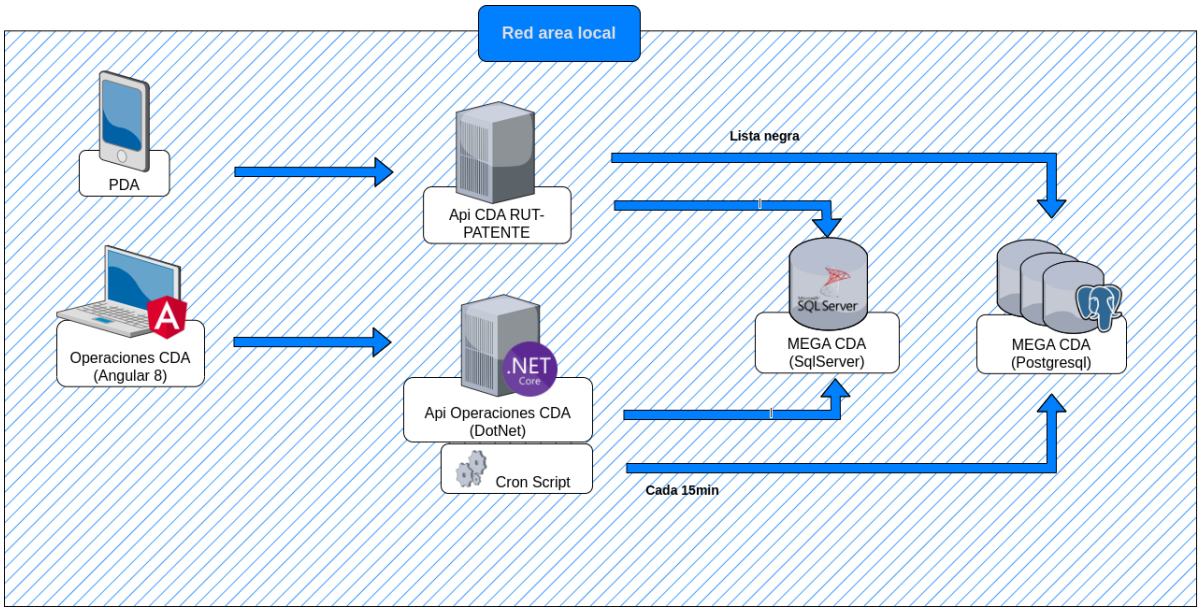
3.6. Preparación de Documentación Técnica

- Se elaboró documentación técnica detallada, incluyendo diagramas de arquitectura, descripciones de la base de datos y guías de usuario para la API.
- Se proporcionaron recomendaciones para mejores prácticas y posibles mejoras en la plataforma.

Estas tareas proporcionan una base sólida para el entendimiento completo de la plataforma, permitiendo así la realización eficiente de mantenimientos evolutivos y correctivos.

4. Arquitectura del Sistema

La arquitectura de la plataforma se caracteriza por su diseño distribuido y orientado a servicios, lo que permite una interacción fluida entre los diferentes componentes que la integran.



A continuación se presenta una descripción detallada de los elementos que componen la arquitectura y su interconexión.

4.1. Dispositivos de Usuario

- **PDA**: Dispositivo portátil utilizado por el personal de campo para realizar operaciones en sitio y enviar datos a las APIs centrales.
- **Operaciones CDA (Angular 8)**: Interfaz de usuario basada en la web desarrollada con Angular 8, que proporciona a los usuarios corporativos acceso a la plataforma y sus funciones.

4.2. APIs y Servicios Centrales

- **Api CDA RUT-PATENTE**: Servicio API que gestiona y valida la información relacionada con las identificaciones de usuarios y vehículos (RUT y patentes).

- **Api Operaciones CDA (.Net)**: API principal de la plataforma, desarrollada en .NET Core, que actúa como el nexo central para el procesamiento y la distribución de datos.
- **Cron Script**: Script de automatización que se ejecuta periódicamente, cada 15 minutos, para realizar tareas de mantenimiento y actualización de datos.

4.3. Bases de Datos

- **MEGA CDA (SQL Server)**: Base de datos central en SQL Server que almacena la información crítica de la organización y coordina la sincronización de datos con otros sistemas.
- **MEGA CDA (PostgreSQL)**: Conjunto de bases de datos en PostgreSQL que almacenan información específica de cada centro, permitiendo una gestión descentralizada y especializada.

4.4. Flujos de Datos e Integración

- Los datos fluyen entre los dispositivos de usuario y las APIs centrales de manera segura y eficiente, asegurando que la información esté actualizada y accesible en tiempo real.
- La API Operaciones CDA se comunica con la base de datos MEGA CDA para recuperar o actualizar la información necesaria para las operaciones diarias.
- Un cron script asegura que las listas negras y otros datos críticos se sincronicen regularmente entre las bases de datos SQL Server y PostgreSQL.

La arquitectura descrita está diseñada para garantizar la escalabilidad, el rendimiento y la seguridad en la gestión de datos de la plataforma, apoyando así los procesos empresariales de manera integral.

5. CDA Operaciones Web

Se detalla la situación actual del proyecto y los detalles importantes que construyen el frontend en angular.

5.1. Listado completo de librerías Angular

Se detallan las librerías usadas en el proyecto Angular, separandolas en las librerías principales y extras.

5.1.1. Librerías principales del Proyecto

Nombre	Versión	Descripción
@angular/animations	8.1.2	Utilizada para animaciones complejas en aplicaciones Angular.
@angular/cdk	8.1.2	Conjunto de herramientas para Angular Material.
@angular/common	8.1.2	Contiene funcionalidades comunes necesarias en muchas aplicaciones Angular.
@angular/compiler	8.1.2	Permite compilar plantillas HTML y TypeScript en componentes Angular.
@angular/core	8.1.2	Núcleo de Angular, necesario para crear componentes y servicios.
@angular/forms	8.1.2	Proporciona herramientas para crear formularios reactivos y basados en plantillas.
@angular/platform-browser	8.1.2	Contiene funcionalidades específicas del navegador, como DOM sanitization.
@angular/platform-browser-dynamic	8.1.2	Permite compilar y ejecutar aplicaciones Angular en el navegador.
@angular/router	8.1.2	Maneja la navegación entre vistas y rutas en aplicaciones Angular.
@angular/cli	8.1.2	Interfaz de línea de comandos para Angular.

Cuadro 1: Listado de librerías principales en el proyecto Angular

5.1.2. Librerías extras del Proyecto

Nombre	Versión	Descripción
apexcharts	3.35.2	Librería de gráficos moderna para visualización de datos.
bootstrap	4.3.1	Framework de front-end para diseño web responsivo y moderno.
css-animator	2.3.0	Herramienta para animaciones CSS complejas.
jquery	3.4.1	Biblioteca de JavaScript rápida y pequeña para simplificar el HTML.
moment	2.24.0	Librería para analizar, validar, manipular y mostrar fechas y horas.
ng-click-outside	4.0.0	Directiva Angular para detectar clics fuera de un elemento.
ng2-validation	4.2.0	Conjunto de validadores para formularios en Angular.
ngx-perfect-scrollbar	8.0.0	Adaptación de la librería perfect-scrollbar para Angular.
rxjs	6.5.2	Biblioteca para programación reactiva utilizando Observables.
rxjs-compat	6.5.2	Paquete de compatibilidad para la actualización a RxJS 6.
screenfull	4.2.0	Interfaz simple para el uso de pantalla completa en la web.
sweetalert2	11.4.4	Librería para crear alertas personalizadas.
tslib	2.1.0	Biblioteca de tiempo de ejecución para ayudar en la transpilación TypeScript.
xlsx	0.18.5	Herramienta de análisis y escritura para varios formatos de hojas de cálculo.
zone.js	0.9.1	Implementación de Zonas para JavaScript, utilizada en Angular para la detección de cambios.
@angular-devkit/build-angular	0.801.2	Herramientas necesarias para compilar aplicaciones Angular.

Cuadro 2: Listado de librerías extras en el proyecto Angular

Nombre	Versión	Descripción
@ng-bootstrap/ng-bootstrap	4.2.1	Integración de Bootstrap con Angular, proporcionando componentes nativos de Bootstrap.
@angular/compiler-cli	8.1.2	CLI que proporciona la funcionalidad del compilador de Angular.
@angular/language-service	8.1.2	Servicio que proporciona información sobre Angular templates.
@types/jasmine	3.3.8	Tipos TypeScript para Jasmine, un framework de pruebas.
@types/jasminewd2	2.0.3	Tipos TypeScript para Jasmine WebDriver.
@types/jquery	3.3.30	Tipos TypeScript para jQuery.
@types/node	8.9.4	Tipos TypeScript para Node.js.
codelyzer	5.0.0	Herramienta de análisis estático para Angular.
jasmine-core	3.4.0	Framework de pruebas para JavaScript.
jasmine-spec-reporter	4.2.1	Reporter para resultados de pruebas Jasmine.
karma	4.1.0	Entorno de pruebas para JavaScript.
karma-chrome-launcher	2.2.0	Lanzador para pruebas Karma en Chrome.
karma-coverage-istanbul-reporter	2.0.1	Reporter para cobertura de código en Karma.
karma-jasmine	2.0.1	Integración de Jasmine con Karma.
karma-jasmine-html-reporter	1.4.0	Reporter HTML para Jasmine y Karma.
node-sass	4.12.0	Biblioteca que proporciona enlace a libSass, compilador de Sass a CSS.
protractor	5.4.0	Framework de pruebas para aplicaciones Angular.
ts-node	7.0.0	Herramienta para ejecutar TypeScript directamente en Node js.
tslint	5.15.0	Herramienta de análisis estático para mejorar la calidad del código TypeScript.
typescript	3.4.3	Lenguaje de programación tipado superset de JavaScript.

Cuadro 3: Listado de librerías extras en el proyecto Angular

5.2. Estructura de directorio y archivos de proyecto

A continuación se detalla la estructura del directorio y archivos del proyecto Angular.

5.2.1. Directorio y archivos principales del proyecto

Componente/Ruta	**Descripción**
src/assets/fonts	Carpeta donde se almacenan las fuentes del portal.
src/assets/images	Carpeta donde se almacenan las imágenes del portal.
src/app/theme/	Carpeta donde se ubican directivas, estructura del dashboard y componentes comunes que serán usados en otras partes del proyecto.
src/app/theme/shared	Carpeta donde se ubican directivas y componentes comunes que serán usados en otras partes del proyecto.
src/app/theme/layout	Carpeta donde se ubica la estructura del dashboard.
src/app/pages/	Carpeta donde se ubican las vistas que cargan en el dashboard.
src/app/dashboard/	Carpeta donde se ubican los componentes que cargan los indicadores que se mostrarán en dashboard.
src/app/app.component	Componente raíz de la aplicación Angular. Gestiona la navegación principal y los eventos de ruta.
src/app/api.service.ts	Servicio central para la gestión de todas las llamadas a la API. Incluye métodos para realizar operaciones CRUD, manejar la autenticación y almacenar información del usuario. Centraliza la interacción con los endpoints de la API del backend.
src/app/app.module.ts	Módulo raíz de la aplicación Angular que declara y agrupa componentes, directivas, y servicios. Incluye la configuración básica de la aplicación, como módulos importados, componentes y proveedores de servicios.

Cuadro 4: Directorio y archivos principales del Proyecto Angular

5.2.2. Directivas usadas en el proyecto

Componente/Ruta	Descripción
src/app/theme/shared/only-number/only-number.directive.ts	Directiva personalizada que permite únicamente la entrada de números en campos de formulario. Esta directiva restringe la entrada del usuario para evitar la inserción de caracteres no numéricos.
src/app/theme/shared/loading.directive.ts	Directiva que muestra un indicador de carga y desactiva interacciones (como botones e inputs) en un elemento durante operaciones de carga. Utiliza una técnica de binding y manipulación del DOM para controlar la visualización y el comportamiento de los elementos.
src/app/directives/ toggle-full-screen.directive.ts	Directiva que permite alternar el modo de pantalla completa de un elemento. Utiliza la librería 'screenfull' para controlar el estado de pantalla completa y modificar la visualización del elemento y sus iconos.

Cuadro 5: Directivas del Proyecto Angular

5.2.3. Directorio de componentes generales en 'src/app/pages/'

Componente/Ruta	Descripción
authentication/	Carpeta donde se ubican componentes de login, cambio de contraseña y reseteo de contraseña.
blacklist/	Componente que maneja la visualización y gestión de una lista negra. Incluye funcionalidades para buscar, editar, eliminar y descargar registros. También gestiona la paginación y activación/desactivación de registros en la lista.
blacklistedit/	Componente utilizado para agregar o editar entradas en la lista negra. Proporciona una interfaz para ingresar o modificar información relevante a un registro de lista negra, como el centro, la patente y detalles adicionales. Incluye validaciones y funcionalidad para guardar los cambios.

Cuadro 6: Directorio de componentes generales en 'src/app/pages/'

centers/	Componente dedicado a la gestión de centros en la aplicación. Permite visualizar, editar, agregar y eliminar centros. Incluye funcionalidades para la búsqueda y filtrado de centros, así como la descarga de los datos de los centros en formato Excel. Maneja la activación y desactivación de registros y la paginación de los resultados.
centersedit/	Componente utilizado para agregar o editar información de centros. Ofrece una interfaz para la administración de detalles de centros, como nombre, servidor, bases de datos y configuraciones relacionadas. Incluye funcionalidades para guardar cambios, validar formularios y manejar la interacción con porterías asociadas al centro.
enterprises/	Componente para la gestión y visualización de empresas. Permite buscar, visualizar y editar información empresarial.
enterprisesedit/	Componente para editar detalles de empresas. Incluye funcionalidades para la gestión y edición de información empresarial, como bodegas y productos.
home/	Componente principal de la aplicación que muestra un panel de control con estadísticas e indicadores. Incluye gráficos y tablas para visualizar datos relevantes, y ofrece funcionalidades para descargar informes y ajustar filtros de búsqueda.
kpifunctional/	Componente que muestra indicadores clave de rendimiento funcional (KPIs) para la aplicación. Incluye gráficos y tablas para visualizar y descargar datos detallados de operaciones y rendimiento.
kpigeneral/	Componente para visualizar indicadores clave de rendimiento general (KPIs). Presenta gráficos y tablas para análisis detallado y descarga de datos, enfocados en la eficiencia y rendimiento operacional.
kpioperative/	Componente para mostrar indicadores de rendimiento operativo. Proporciona análisis detallados y visualizaciones de datos, incluyendo gráficos y tablas para la eficiencia operativa y métricas de rendimiento.
kpivisitas/	Componente que muestra indicadores de rendimiento relacionados con visitas. Proporciona visualizaciones de datos, gráficos y tablas para el análisis de visitas, incluyendo el seguimiento de vehículos y personas.

Cuadro 7: Directorio de componentes generales en 'src/app/pages/'

secureusers/	Componente para la gestión de usuarios seguros. Proporciona funcionalidades para visualizar, agregar, editar y activar o desactivar usuarios, así como para descargar informes de usuarios con sus detalles asociados.
secureusersedit/	Componente para editar o agregar usuarios seguros. Incluye campos para ingresar información del usuario, asignar centros y empresas, y establecer perfiles. Proporciona validaciones y funcionalidades para guardar o cancelar cambios.
userlist/	Componente para visualizar y gestionar una lista de inventario de un usuario. Ofrece opciones para buscar, editar y eliminar registros del inventario. Permite filtrar por diversos criterios como almacén, usuario y tipo de transferencia.
whitelist/	Componente para la gestión de la lista blanca. Permite visualizar, agregar, editar, eliminar y activar o desactivar registros de la lista blanca, y descargar informes relacionados. Incluye opciones de búsqueda y filtrado por centro, empresa, patente, nombre y estado.
whitelistedit/	Componente para editar o agregar registros a la lista blanca. Incluye campos para el ingreso de información detallada del registro, como patente, nombre, rut y empresa asociada. Permite seleccionar centros y empresas según el perfil del usuario y ofrece opciones para guardar o cancelar los cambios realizados.
tables/	carpeta contenedora para distintos tipos de tablas creadas con bootstrap (basic, border, sizing, styling).
layout/	carpeta contenedora para crear componentes con varias tarjetas informativas que muestran estadísticas y datos clave como visitantes, volumen, archivos y correos, utilizando íconos y gráficos para una presentación visual y fácil de entender.

Cuadro 8: Directorio de componentes generales en 'src/app/pages/'

5.3. Environment

En Angular, se utilizan archivos de entorno para definir variables de configuración específicas del entorno en el que se ejecuta la aplicación. Estos archivos permiten una fácil adaptación del código a diferentes entornos, como desarrollo, pruebas y producción.

5.3.1. Archivo de Entorno de Producción (`environment.prod.ts`)

El archivo de entorno de producción se utiliza cuando la aplicación se construye en modo producción. Contiene configuraciones específicas de este entorno. Por ejemplo:

```
export const environment = {
  production: true,
  api: 'https://operacionescda.redmegacentro.cl/api/api/',
};
```

Aquí, la propiedad `production` se establece en `true`, y la URL de la API se configura para el entorno de producción.

5.3.2. Archivo de Entorno de Pruebas (`environment.ts`)

Para el entorno de desarrollo o pruebas, se utiliza un archivo de entorno diferente. Este archivo se reemplaza por `environment.prod.ts` durante la construcción del proyecto en modo producción. Un ejemplo es:

```
export const environment = {
  production: false,
  api: 'http://localhost:56675/api'
};
```

En este caso, `production` se establece en `false`, indicando que no es un entorno de producción, y la URL de la API apunta a un servidor local para pruebas de desarrollo.

La gestión de estos archivos permite modularizar y adaptar la configuración de la aplicación de manera eficiente y controlada según el entorno de ejecución.

6. CDA Operaciones API

Se detalla la situación actual del proyecto y los detalles importantes que construyen la API Operaciones CDA desarrollada en .NET Framework.

6.1. Información general sobre la API

La API Operaciones API es una interfaz de programación de aplicaciones desarrollada con .NET Framework 4.7.2, diseñada para facilitar el acceso y la gestión de datos en el entorno empresarial de nuestra organización.

Esta API ofrece una conexión segura y eficiente a dos bases de datos fundamentales: una en SQL Server llamada MEGACDA y otra en PostgreSQL. Su capacidad para integrarse con sistemas de gestión internos y aplicaciones de terceros la convierte en una herramienta esencial para el manejo centralizado de datos empresariales.

Funcionalidades Principales:

- Acceso centralizado a datos empresariales, incluyendo conexiones con las bases de datos SQL Server (MEGACDA) y PostgreSQL.
- Integración con sistemas de gestión internos y aplicaciones de terceros.
- Soporte para operaciones de CRUD (Crear, Leer, Actualizar, Eliminar).
- Capacidad para manejar grandes volúmenes de datos con alta eficiencia.

6.2. Listado completo de librerías .NET.

Se detallan las librerías usadas en el proyecto .NET Framework.

6.2.1. Librerías 'Microsoft.AspNet' del Proyecto.

Nombre	Versión	Descripción
Cors	5.2.8	Permite la configuración de la política de origen cruzado en aplicaciones web.
Mvc	5.2.7	Marco para construir aplicaciones web escalables y basadas en estándares.
Mvc.es	5.2.7	Versión en español de Microsoft.AspNet.Mvc.
Razor	3.2.7	Motor de plantillas para integrar código del lado del servidor con HTML.
Razor.es	3.2.7	Versión en español de Microsoft.AspNet.Razor, para integración de código del servidor con HTML.
Web.Optimization	1.1.3	Proporciona funcionalidades para la optimización de recursos web como CSS y JavaScript.
Web.Optimization.es	1.1.3	Versión en español de Microsoft.AspNet.Web.Optimization.
WebApi	5.2.7	Marco para construir servicios HTTP, parte esencial para aplicaciones web API.
WebApi.Client	5.2.8	Provee clases para enviar peticiones HTTP y recibir respuestas HTTP.
WebApi.Client.es	5.2.8	Versión en español de Microsoft.AspNet.WebApi.Client.
WebApi.Core	5.2.8	Núcleo de ASP.NET Web API, contiene las funcionalidades principales.
WebApi.Core.es	5.2.8	Versión en español de Microsoft.AspNet.WebApi.Core.

Cuadro 9: Listado de librerías 'Microsoft.AspNet' del proyecto .Net

WebApi.Cors	5.2.8	Proporciona soporte para compartir recursos entre distintos orígenes en Web API.
WebApi.HelpPage	5.2.7	Ayuda a generar páginas de ayuda para servicios web API.
WebApi.WebHost	5.2.7	Facilita la integración de ASP.NET Web API con IIS usando ASP.NET Routing.
WebApi.WebHost.es	5.2.7	Versión en español de Microsoft.AspNet.WebApi.WebHost.
WebPages	3.2.7	Proporciona una forma sencilla de combinar código de servidor con HTML.
WebPages.es	3.2.7	Versión en español de Microsoft.AspNet.WebPages.

Cuadro 10: Listado de librerías 'Microsoft.AspNet' del proyecto .Net

6.2.2. Librerías 'Microsoft.AspNetCore' del Proyecto.

Nombre	**Versión**	**Descripción**
Cors	2.2.0	Permite la configuración de la política de origen cruzado en aplicaciones ASP.NET Core.
Http.Abstractions	2.2.0	Define las abstracciones para HTTP, como HttpContext, HttpRequest, HttpResponse, etc.
Http.Extensions	2.2.0	Provee métodos de extensión para trabajar con objetos HTTP en ASP.NET Core.
Http.Features	2.2.0	Define las interfaces para las características del servidor HTTP en ASP.NET Core.

Cuadro 11: Listado de librerías 'Microsoft.AspNetCore' del proyecto .Net

6.2.3. Librerías extras del Proyecto.

Nombre	Versión	Descripción
Antlr	3.5.0.2	Utilizada para el análisis de lenguajes, facilitando la interpretación de lenguajes de programación.
bootstrap	3.4.1	Framework de front-end para diseño web responsivo y moderno.
jQuery	3.4.1	Biblioteca de JavaScript rápida y pequeña para simplificar la manipulación de HTML y eventos.
Microsoft.Bcl.AsyncInterfaces	6.0.0	Proporciona interfaces para operaciones asíncronas en .NET.
Microsoft.Bcl.HashCode	1.1.1	Facilita la creación de códigos hash consistentes en diferentes entornos.
Microsoft.CodeDom.Providers .DotNetCompilerPlatform	2.0.1	Provee un proveedor de compilación para lenguaje C y VB usando las herramientas de Roslyn.
Microsoft.Extensions .Configuration.Abstractions	2.2.0	Abstracciones para acceder a la configuración en aplicaciones .NET.
Microsoft.Extensions .DependencyInjection.Abstractions	2.2.0	Abstracciones para el patrón de diseño de inyección de dependencias.
Microsoft.Extensions .FileProviders.Abstractions	2.2.0	Abstracciones para proveedores de archivos en aplicaciones .NET.
Modernizr	2.8.3	Biblioteca de JavaScript que ayuda a los desarrolladores web a detectar características del navegador.

Cuadro 12: Listado de librerías extras del proyecto .NET Framework

6.3. Estructura de directorio y archivos de proyecto

A continuación se detalla la estructura del directorio y archivos del proyecto .NET.

6.3.1. Directorio y archivos principales del proyecto

Componente/Ruta	**Descripción**
models/	Este directorio contiene los archivos con extensión ".cs", que estan diseñados para representar y manejar modelos datos.
fonts/	Este directorio se guardan los archivos de estilos de fuentes.
Scripts/	Este directorio se guardan los archivos de javascripts, que contienen librerias de Bootstrap, Modernizr y Jquery.
Content/	Este directorio se guardan los archivos de estilos css, que contienen las librerias de Bootstrap.
Controllers/	Directorio que contiene los archivos que son responsable de manejar las operaciones relacionadas con la obtención de datos a través de solicitudes POST, y sincronizar estos cambios con múltiples bases de datos.
Web.config	Archivo de configuración de seguridad, manejo de errores, acceso a bases de datos, y ajustes específicos del servidor.
AppStart/RouteConfig.cs	Archivo de configuración que interpreta y maneja el enrutamiento en tu aplicación ASP.NET MVC.
AppStart/BundleConfig.cs	Archivo organizador que agrupa varios archivos JavaScript y CSS juntos para que la API cargue más rápido.
AppStart/FilterConfig.cs	Archivo necesario para configurar los filtros que se aplicarán de manera global en la aplicación. Se está añadiendo un filtro de manejo de errores para capturar y gestionar adecuadamente las excepciones no controladas en las acciones de los controladores.
AppStart/WebApiConfig.cs	Archivo necesario para maneja las solicitudes a la API Web, incluyendo habilitar CORS, configurar enrutamiento y definir patrones de URL para acceder a los controladores y acciones de la API.

Cuadro 13: Directorio y archivos principales del Proyecto .NET

6.4. Cadena de Conexión a la Base de Datos MEGACDA

La conexión a las bases de datos se realiza en varios pasos. Primero se establece la conexión con la base de datos central SQL Server `MEGACDA`, y luego se procede a conectarse a las bases de datos PostgreSQL de cada centro. A continuación se describe este proceso:

Configuración de Conexión en Web.config: La configuración de conexión a la base de datos SQL Server `MEGACDA` se define en el archivo `Web.config`, como se muestra a continuación:

```
<add key="SQL_MEGACDA" value="Data Source=
   megasqlqas;Initial Catalog=MEGACDA;
user id=megacentro\servicio_gestioncda;
   password=!DFRG45fcvc.!;
Integrated Security=SSPI;
   MultipleActiveResultSets=true" />
```

Esta configuración especifica todos los parámetros necesarios para establecer la conexión con la base de datos `MEGACDA`, incluyendo el nombre del servidor, el nombre de la base de datos, las credenciales de usuario y otros ajustes relevantes.

6.4.1. Consulta a la Base de Datos SQL Server:

Se realiza una consulta SQL para obtener los detalles de conexión para cada centro activo. Esta consulta se efectúa en la base de datos SQL Server llamada `MEGACDA`.

```
SqlCommand command = new SqlCommand("Select *
   from CDA_Centros where activo = 1", conn);
SqlDataAdapter da = new SqlDataAdapter(command
   );
DataSet ds = new DataSet();
da.Fill(ds);
```

6.4.2. Iteración sobre los Resultados:

Se recorren los resultados de la consulta para obtener los detalles específicos de conexión a PostgreSQL de cada centro.

```
foreach (DataRow lect in ds.Tables[0].Rows)
{
    dynamic fila = new ExpandoObject();
    // ... asignacion de valores a fila ...
}
```

6.4.3. Establecimiento de la Conexión a PostgreSQL:

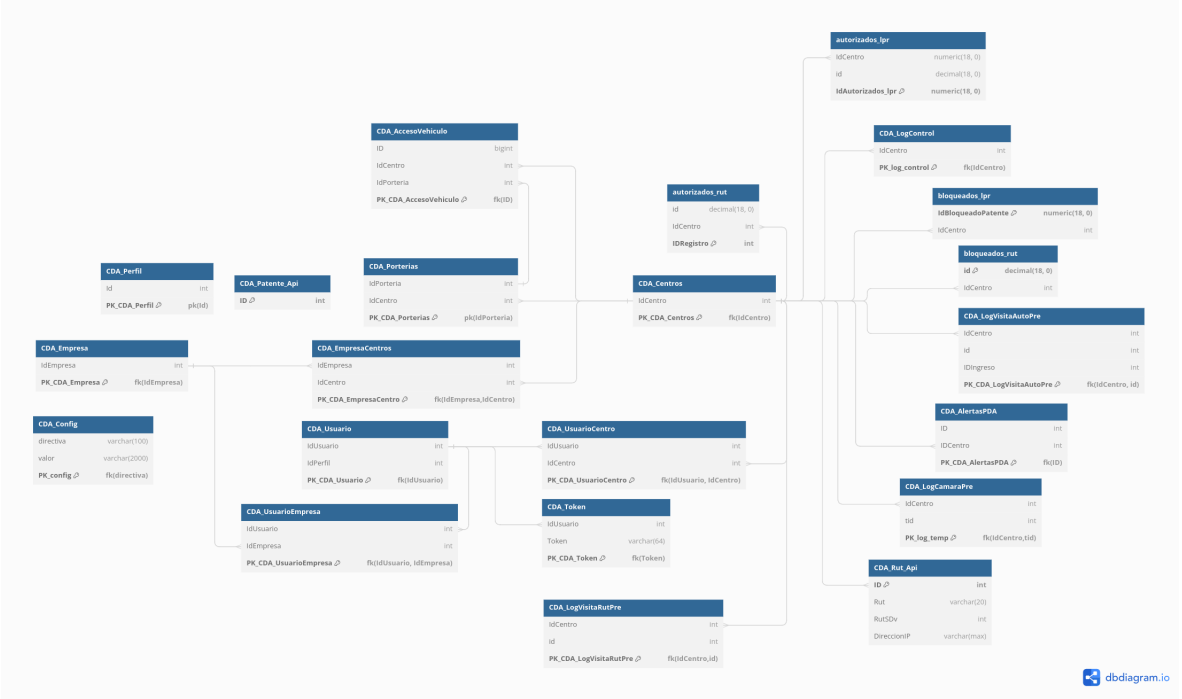
Para cada centro, se utiliza la información recopilada para crear una nueva conexión a PostgreSQL.

```
NpgsqlConnection conn2 =
    new NpgsqlConnection("Server=" + fila.
       ServidorPosgress +
         ";Port=" + fila.PuertoPosgress +
         ";User Id=" + fila.UsuarioAuto +
         ";Password=" + fila.ContrasenaAuto +
         ";Database=" + fila.BaseDatosAuto +
         ";CONNECTIONLIFETIME=10; MaxPoolSize=1
           000;");
conn2.Open();
```

Este método permite la conexión a las bases de datos PostgreSQL específicas de cada centro, utilizando los detalles de conexión almacenados y gestionados centralmente en SQL Server.

7. Base de Datos MEGACDA

La base de datos 'MEGACDA' esta construida usando SQL Server. En esta se definen varias tablas esenciales para el almacenamiento y gestión de la información relacionada con el control de acceso y administración de centros.



7.1. Descripción de las Tablas en MEGACDA

A continuación, se describen algunas de las tablas clave:

- **autorizados_lpr**: Guarda información de vehículos autorizados por patente.
- **autorizados_rut**: Guarda información de personas autorizadas por RUT.
- **bloqueados_lpr**: Almacena información sobre patentes bloqueadas.
- **bloqueados_rut**: Registra información de RUTs bloqueados.
- **CDA_AccesoVehiculo**: Controla el acceso de vehículos a los centros.
- **CDA_AlertasPDA**: Registra alertas generadas por dispositivos PDA.
- **CDA_Centros**: Contiene información sobre los centros operativos.
- **CDA_Config**: Almacena configuraciones y directivas del sistema.
- **CDA_Empresa**: Contiene información sobre las empresas asociadas.
- **CDA_EmpresaCentros**: Relaciona empresas con centros.
- **CDA_LogCamaraPre**: Guarda registros previos de cámaras.
- **CDA_LogControl**: Almacena registros de control para los centros.
- **CDA_LogVisitaAutoPre**: Registra visitas previas de vehículos.
- **CDA_LogVisitaRutPre**: Almacena visitas previas basadas en RUT.
- **CDA_Patente_Api**: Contiene datos obtenidos de una API de patentes.
- **CDA_Perfil**: Almacena información de perfiles de usuario.
- **CDA_Porterias**: Registra datos de las porterías de los centros.
- **CDA_Rut_Api**: Almacena datos obtenidos de una API de RUTs.
- **CDA_Token**: Gestiona tokens de autenticación de usuarios.
- **CDA_Usuario**: Almacena información de los usuarios del sistema.
- **CDA_UsuarioCentro**: Relaciona usuarios con centros específicos.
- **CDA_UsuarioEmpresa**: Relaciona usuarios con empresas específicas.

7.2. Procedimientos Almacenados en MEGACDA

Los procedimientos almacenados en SQL, conocidos como *'stored procedures'*, son fundamentales para operar colecciones de comandos SQL que se conservan en la base de datos. Pueden ejecutarse según se requiera para realizar diversas tareas automatizadas y de mantenimiento.

7.2.1. Identificar Procedimientos en SQL

Los procedimientos almacenados en SQL son identificables por su estructura y sintaxis específicas. En general, un procedimiento almacenado comienza con la palabra clave `CREATE PROCEDURE` seguida del nombre del procedimiento y una lista de parámetros, si los hay. A continuación, se define el cuerpo del procedimiento, que contiene la lógica y las sentencias SQL que se ejecutarán cuando se llame al procedimiento.

Un ejemplo típico de la declaración de un procedimiento almacenado es el siguiente:

```
CREATE PROCEDURE NombreDelProcedimiento
    @Parametro1 TipoDeDato,
    @Parametro2 TipoDeDato,
    ...
AS
BEGIN
    -- Cuerpo del procedimiento
    -- Sentencias SQL
END
```

Los procedimientos almacenados son una herramienta poderosa para encapsular lógica de negocio, realizar operaciones complejas y mejorar el rendimiento de las aplicaciones que interactúan con bases de datos SQL.

7.2.2. Listado de procedimientos en MEGACDA

A continuación, se describen algunos de los procedimientos almacenados más relevantes en el sistema:

Procedimiento Sql	Descripción
spGet_EstadoCentros	Obtiene información del estado de los centros a partir de CDA_LogControl y CDA_Centros, ordenados por nombre.

Procedimiento Sql	**Descripción**
spGet_KpiGeneral	Realiza consultas para reunir datos de ingresos, utilizados en reportes y dashboards personalizados.
AUTO_FIX_AUTOSCAN	Verifica y actualiza el bloqueo en cda_config si ha pasado más de una hora desde la última operación.
AUTO_MIGRATE_RECORDS	Migración y procesamiento de registros de acceso de vehículos y gestión de duplicados.
AUTO_MIGRATE_SISALIO	Actualiza registros en CDA_AccesoVehiculo marcando si los vehículos han salido.
spDel_Porteria	Elimina una porteria específica por su ID.
spGet_AlertasMonitor	Obtiene alertas recientes de ingreso para monitoreo basadas en segundos y prioridad.
spGet_APK_NoEsListaNegra	Verifica si una patente está en lista negra por centro y devuelve 'OK' o información asociada.
spGet_Centros	Recupera información de centros con su log de control, filtrados por diversos criterios.
spGet_EmpresasPorCentro	Lista empresas asociadas a un centro, opcionalmente por destino o nombre fantasía.
spGet_EstadoCentros	Provee información sobre el estado de los centros, incluyendo nombre y mensaje del servidor.
spGet_KpiFuncional	Genera datos de ingresos para KPI Funcional, detallando tipo de ingreso y tiempo de atención.
spGet_KpiOperativo	Ofrece datos para KPI Operativo, sobre ingresos por centro y correcciones de lista blanca.
spGet_ListaBlanca	Recupera patentes de la lista blanca por centro, PPU, nombre y rut.
spGet_ListaNegra	Extrae patentes de la lista negra por centro de origen y PPU.
spGet_LoginUsuario	Identifica usuarios por AD o localmente, retornando nombre, perfil y estado.
spGet_Perfiles	Recupera información de perfiles de la aplicación.
spGet_Porterias DisponiblesPorCentro	Obtiene porterías disponibles por centro para agregar o existentes.
spGet_PorteriasPorCentro	Recopila información de porterías por centro, enfocado en 'approaching'.
spGet_Usuarios	Proporciona detalles sobre usuarios, con filtros por estado, ID de usuario y más.
spIns_AlertaPDA	Inserta alertas desde una PDA con detalles como prioridad y mensaje.

Procedimiento Sql	**Descripción**
spPro_Centros	Crea o actualiza información de centros, incluyendo cambio de estado.
spPro_ListaBlanca	Permite crear o actualizar registros en la lista blanca.
spPro_ListaBlancaRut	Gestiona RUTs en la lista blanca, para creación o actualización.
spPro_ListaNegra	Administra la lista negra, permitiendo añadir o modificar patentes.
spPro_Porterias	Crea o modifica información de porterías, incluyendo nuevas adiciones.
spPro_Usuarios	Gestiona la creación o actualización de usuarios, perfiles y asociaciones.

Cuadro 14: Descripción de los Procedimientos Almacenados en MEGACDA

Cada uno de estos procedimientos es crucial para el funcionamiento integral del sistema, automatizando tareas y proporcionando información vital para la toma de decisiones.

8. Responsabilidad del documento

Elaborado por:

Firma	Nombre	Cargo	Fecha
Carlos A.	Carlos Arias	Dev. Full Stack	27-11-23
nombre firma	nombre	cargo	fecha

Revisado por:

Firma	Nombre	Cargo	Fecha
Fabián A.	Fabián Aguero	Arquitecto de Sistemas	27-11-23
nombre firma	nombre	cargo	fecha

Aprobado por:

Firma	Nombre	Cargo	Fecha
Richard C.	Richard Castillo	Líder Técnico	27-11-23
nombre firma	nombre	cargo	fecha