Manual Técnico

SISTEMA DE GESTIÓN DE SEGUROS AUTOMOVILÍSTICOS

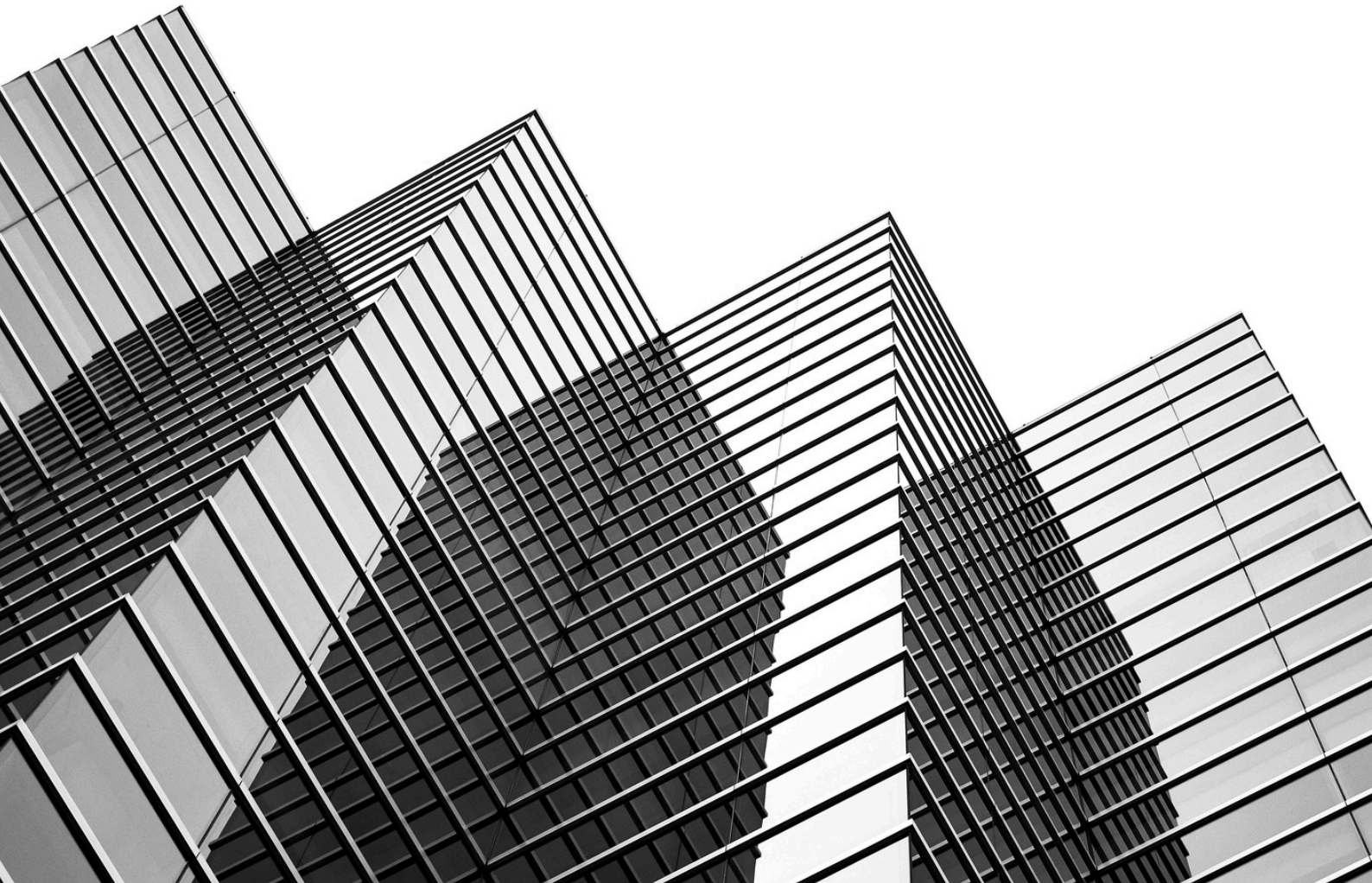

Tabla de contenido

INTRODUCCIÓN

"GESTIONAR SEGUROS AUTOMOVILÍSTICOS NO SOLO ES ASEGURAR VEHÍCULOS, SINO TAMBIÉN BRINDAR TRANQUILIDAD A QUIENES CONFÍAN EN NOSOTROS. CADA REGISTRO QUE MANEJAS ES UN PASO HACIA UN FUTURO MÁS SEGURO Y CONFIABLE. TU DEDICACIÓN Y PRECISIÓN TRANSFORMAN DESAFÍOS EN OPORTUNIDADES Y ASEGURAN LA TRANQUILIDAD EN CADA VIAJE."

Este manual está diseñado para ayudar a los asesores de seguros automovilísticos a utilizar el sistema de gestión desarrollado con Python y Flask. El sistema permite realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre los datos de automóviles para tener mas accesibilidad al cotizar un seguro automovilistico.

Requisitos del sistema

Requisitos de hardware:

- **Servidor: CPU Quad-core, 16 GB RAM, 100 GB SSD**
- **Cliente: Navegador moderno con soporte para JavaScript (Chrome, Firefox)**

Requisitos de software:

- **Backend:**
  - **Lenguaje: Python, Java, o Node.js**
  - **Framework: Flask**
  - **Base de datos: MySQL**
- **Frontend:**
  - **Framework: React.js o Vue.js**
  - **HTML5, CSS3, JavaScript**

Requisitos adicionales:

- **Sistema operativo: Linux, Windows o MacOS**

Descripción funcional del sistema

1. El sistema permite la gestión completa de los seguros automovilísticos, cubriendo las siguientes entidades principales:

- Vehículos: Registro de vehículos asegurados.

2. Características principales:

- Crear: Permite crear nuevas entradas en la base de datos.
- Leer: Permite leer y mostrar información de una o varias entidades.
- Actualizar: Permite actualizar los datos existentes en el sistema.
- Eliminar: Permite borrar entradas del sistema.
  -

Modelo de Base de Datos

Estructura básica de las tablas:

Table: automovil

Columns:

id	int(11) AI PK
id_automovil	int(11)
modelo_automovil	varchar
id_marca	int(11)
id_lineavehiculo	int(11)
id_tipovehiculo	int(11)

Descripción de las Funcionalidades CRUD

Crear vehículo: Añadir un nuevo vehículo a un cliente.

```
"marca": "Toyota",
"modelo": "Corolla",
"año": 2020,
"placa": "ABC-1234",
```

Leer vehículos: Listar todos los vehículos o uno en específico.

```
GET /vehiculos/
```

Actualizar vehículo: Actualizar detalles de un vehículo.

```
PUT /vehiculos/{id}
{
    "modelo": "Camry"
}
```

Eliminar vehículo: Eliminar un vehículo registrado.

```
DELETE /vehiculos/{id}
```

EndPoints y Estructura de API

Vehículos:

- POST /vehiculos/: Crear vehículo.
- GET /vehiculos/: Leer vehículos.
- PUT /vehiculos/{id}: Actualizar vehículo.
- DELETE /vehiculos/{id}: Eliminar vehículo.

Consideraciones de Seguridad

- Autenticación: Implementación de OAuth2 o JWT para proteger el acceso a las APIs.
- Validación de entradas: Todos los datos enviados al servidor deben ser validados.
- Cifrado: El tráfico de datos debe ser cifrado mediante HTTPS.
- Manejo de errores: Implementar respuestas de errores claras y concisas (códigos de estado HTTP apropiados).

Pruebas y Validación

- Pruebas unitarias: Para cada funcionalidad CRUD.
- Pruebas de integración: Asegurar que todas las entidades interactúan correctamente.
- Pruebas de carga: Evaluar la capacidad del sistema bajo distintas condiciones de tráfico.

Mantenimiento y Actualizaciones

- Documentación: Mantener actualizada la documentación del API.
- Backups: Programar respaldos automáticos de la base de datos.
- Monitorización