

Nombre: Ángel Gabriel Martínez Moreno Ana Daniela Torrero Jasso Pablo Torres Doria Luis Mario Quintanilla Álvarez	Matrícula: 3004203 2953691 3003910 2886747
Nombre del curso: Proyecto integrador de la metodología DevOps	Profesor: Julio Antonio García Moreno
Módulo: 3	Fecha: 18 de mayo de 2025
Bibliografía: <ol style="list-style-type: none">1. Sahin, K. (2025, 20 enero). Python Web scraping: full tutorial with examples (2025). <i>ScrapingBee</i>. https://www.scrapingbee.com/blog/web-scraping-101-with-python/2. Python For ETL: How to Build ETL Pipelines With Examples. (2024, 25 junio). Airbyte. https://airbyte.com/data-engineering-resources/python-etl3. Lumigo. (2024, 21 agosto). How to Deploy AWS Lambda with Terraform. https://lumigo.io/aws-lambda-deployment/aws-lambda-terraform/4. <i>El comparador de autos nuevos más completo en México</i> my-car.mx. (s. f.). My Car México. https://my-car.mx/comparador5. Autocosmos. (s. f.). Comparador de autos. Autocosmos.com. https://www.autocosmos.com.mx/catalogo/comparar	

Fase VI: Resultados.

Enlace al repositorio: <https://github.com/PabloDoria/Car-Wizard>

Nombre del proyecto: CarWizard

Descripción: Página web de consulta de datos financieros, técnicos y de propiedades de los autos de las marcas más comunes a nivel mundial, donde podrás filtrar por marca, precio, financiamiento o datos específicos como consumo.

El proyecto hasta el momento tiene un fuerte desarrollo en la parte backend y de infraestructura, el trabajo ha tenido muchas extensiones y ha escalado en gran medida, para explicar de forma general como funciona la arquitectura que tenemos hasta el momento, comenzaremos por definir sus partes.

1. Infraestructura en AWS (Terraform)

Networking (networking.tf):

- VPC dedicada para el proyecto
- 4 subredes:
 - 2 públicas para el ALB (Application Load Balancer)
 - 2 privadas para RDS y ECS
- Internet Gateway para acceso a internet
- NAT Gateway para que los servicios en subredes privadas accedan a internet

- Tablas de rutas para gestionar el tráfico

Base de Datos (rds.tf):

- Instancia RDS SQL Server
- Configuración:
 - Clase: db.t3.micro
 - 20GB de almacenamiento GP2
 - Grupo de parámetros personalizado para UTF-8
 - Ubicada en subredes privadas
- Credenciales:
 - Usuario: admin
 - Contraseña: generada automáticamente y almacenada en Secrets Manager

Lambda Functions (lambda.tf):

Tres funciones Lambda para gestionar datos:

1. schema_generator:
 - Crea la estructura de la base de datos
 - Define tablas: years, makes, models, engines, bodies, etc.
2. db_initializer:
 - Espera a que RDS esté disponible
 - Ejecuta el schema SQL generado
3. data_loader:
 - Obtiene datos de la API de coches
 - Procesa y carga datos en las tablas
 - Se ejecuta diariamente mediante EventBridge

Contenedores (ecs.tf, ecr.tf):

- Cluster ECS para la aplicación Laravel
- Task Definition con:
 - Contenedor Laravel
 - Variables de entorno para DB
 - Configuración de red y memoria
- ECR Repository para almacenar imágenes Docker

Balanceador de Carga (alb.tf):

- Application Load Balancer público
- Target Group para los contenedores ECS
- Reglas de enrutamiento HTTP/HTTPS

Seguridad:

- Security Groups específicos para:
 - ALB (entrada HTTP/HTTPS)
 - ECS Tasks (tráfico desde ALB)
 - RDS (tráfico desde ECS y Lambda)
- IAM Roles y Políticas para:
 - ECS Tasks
 - Lambda Functions
 - GitHub Actions

2. Gestión de Secretos**AWS Secrets Manager (secrets.tf):**

- Almacena credenciales de RDS
- Almacena tokens y claves de API
- Accesible por Lambda y ECS

GitHub Secrets:

- AWS_ROLE_ARN: Rol para GitHub Actions
- RDS_PASSWORD: Contraseña de la base de datos
- Otros secretos de configuración

3. Pipeline de Despliegue (GitHub Actions)**Workflow Principal (deploy-steps.yml):**

Pasos secuenciales:

1. **Networking:**
 - Crea VPC y subredes
 - Configura rutas y gateways
2. **Database:**
 - Despliega RDS
 - Crea grupos de parámetros
 - Configura seguridad
3. **Lambda:**
 - Empaqueta código Python
 - Despliega funciones
 - Configura triggers
4. **ECS:**
 - Construye imagen Docker

- Publica en ECR
- Despliega servicios

5. **App:**

- Configura Laravel con Ansible
- Ejecuta migraciones
- Actualiza servicios ECS

4. Configuración de Aplicación (Ansible)

Roles:

1. **Laravel** (ansible/roles/laravel):

- Configura .env
- Establece permisos
- Gestiona caché y storage

2. **Docker** (ansible/roles/docker):

- Configura PHP
- Optimiza configuraciones

6. Scripts de Base de Datos

Estructura (scripts/database/):

- generate_schema.py: Genera estructura de tablas
- db_initializer.py: Inicializa la base de datos
- table_create.py: Funciones auxiliares para SQL
- db_connector.py: Gestión de conexiones

Tablas Principales:

- years: Años de vehículos
- makes: Fabricantes
- models: Modelos de vehículos
- engines: Especificaciones de motores
- bodies: Características de carrocería
- mileages: Datos de consumo
- make_models: Relación make-model
- trims: Versiones específicas

7. Monitoreo y Logs

CloudWatch (cloudwatch.tf):

- Logs de Lambda
- Métricas de RDS
- Alertas de ECS
- Eventos programados

8. Seguridad y Acceso

IAM (iam.tf, team-access.tf):

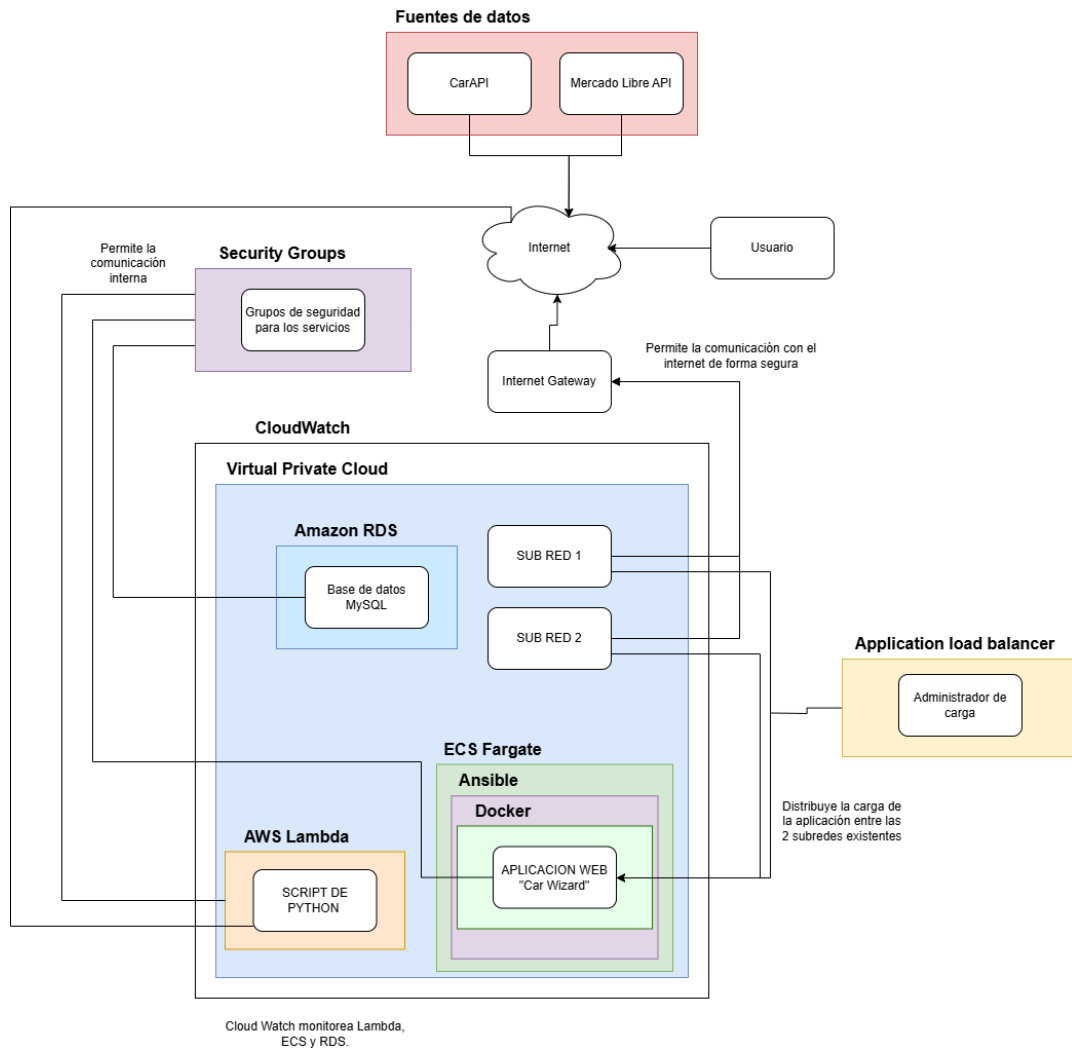
- Roles por servicio
- Políticas mínimas necesarias
- Acceso por equipo
- MFA y rotación de credenciales

Infraestructura

La infraestructura, ya utiliza todas las tecnologías que contemplamos para el proyecto, estas son las tecnologías que implementamos:

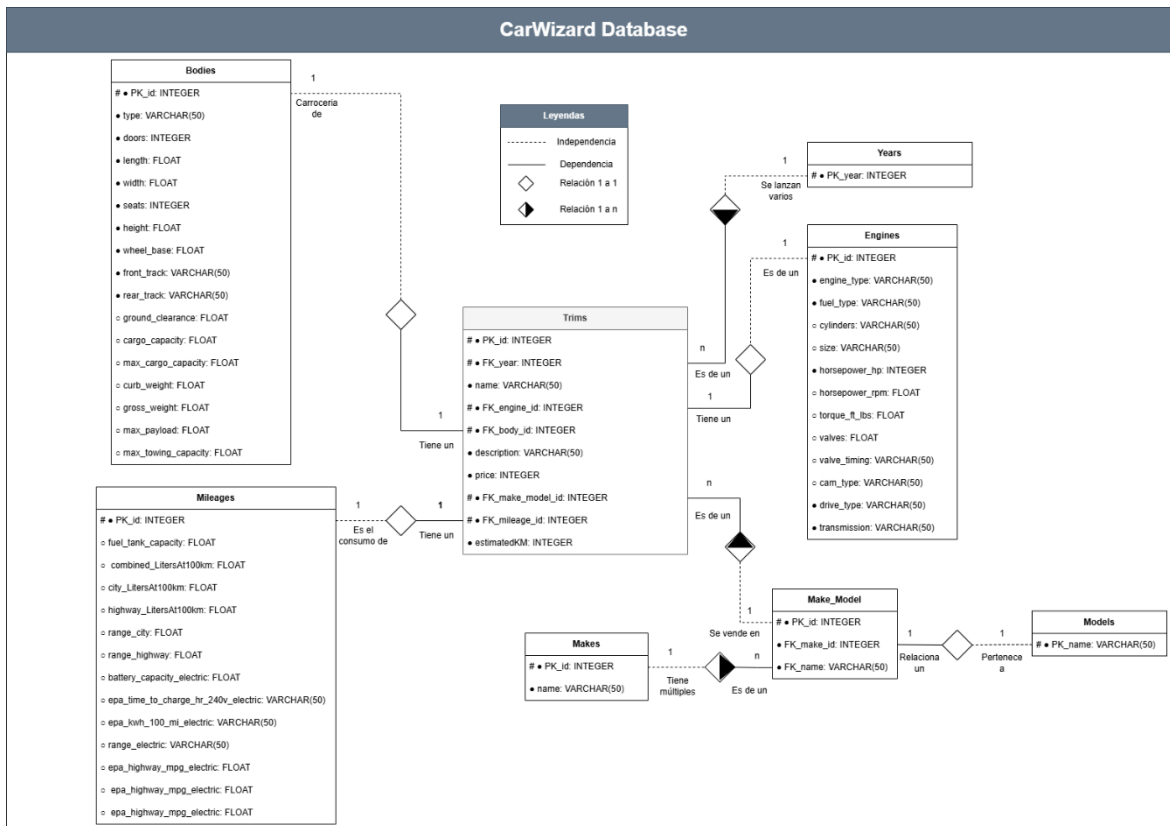
- **Terraform:** Para levantar toda la infraestructura y los recursos de AWS.
- **Python:** Para el sistema de scripts que obtienen los datos, los procesan, crean las tablas y la base de datos.
- **Ansible:** Para configurar el ambiente de la aplicación en Amazon ECS, donde se alojará la aplicación web.
- **Github actions:** Para la integración y automatización, utilizándolo podemos levantar la infraestructura, aplicar cambios específicos, etc.
- **CloudWatch (Monitoreo):** Usamos CloudWatch para monitorear nuestros recursos, no solo para ver cuanto están consumiendo sino también la carga económica.
- **Amazon Secrets (Sustituto de Vault):** Usamos Amazon Secrets Manager para guardar el token de la API, credenciales de RDS, variables de entorno para la aplicación, credenciales de AWS. Usamos AWS Secrets Manager porque:
 - Se integra automáticamente con nuestros servicios AWS (Lambda, ECS, RDS)
 - No requiere mantener servidores adicionales como Vault
 - La rotación de credenciales es automática
 - El costo es más predecible (pagas por secreto)
 - En resumen: es más simple y se ajusta perfectamente a nuestra arquitectura AWS.

Una visión general de la arquitectura luce así:



Base de datos

La base de datos ya está realizada en base a los datos que fuimos capaces de obtener de los vehículos, ya es utilizable, como hemos montado todo de tal forma que sea un sistema completamente automático, tenemos un script que genera un archivo llamado "Scheme.sql" que contiene el diseño de la base de datos con sus relaciones. Este es el diagrama de entidad relación:



De la misma forma, creamos un diccionario de datos con las definiciones de cada uno de los datos que tenemos en nuestra base:

Makes		
Column	Data type	Description
PK_id	Whole number	Unique identifier of the make
name	Text (50)	Name of the car manufacturer (e.g., Honda, Toyota)

Make_Model			
Column	Data type	Description	Notes / Relationships
PK_id	Whole number	Unique identifier of the make-model combination	Primary Key
FK_make_id	Whole number	Foreign key referring to the make	Foreign Key → Makes.PK_id
FK_name	Text (50)	Foreign key referring to the model name	Foreign Key → Models.PK_name

Models			
Column	Data type	Description	Notes
PK_name	Text (50)	Unique name of the model (e.g., Civic, Corolla)	Primary Key

Years		
Column	Data type	Description
PK_year	Whole number	Unique identifier for a car year

Bodies			
Column	Data type	Description	Units (metric)
PK_id	Whole number	Unique identifier of car body	—
type	Text (max 50)	Type of vehicle body (e.g., Sedan, SUV)	—
doors	Whole number	Number of doors	units
length	Decimal	Vehicle length	centimeters (cm)
width	Decimal	Vehicle width	centimeters (cm)
seats	Whole number	Number of seats	units
height	Decimal	Vehicle height	centimeters (cm)
wheel_base	Decimal	Distance between front and rear axles	centimeters (cm)
front_track	Text (max 50)	Width between front wheels (nullable or textual)	—
rear_track	Text (max 50)	Width between rear wheels (nullable or textual)	—
ground_clearance	Decimal	Height from ground to undercarriage	centimeters (cm)
cargo_capacity	Decimal	Standard cargo volume	liters (L)
max_cargo_capacity	Decimal	Maximum cargo volume with seats folded	liters (L)
curb_weight	Decimal	Weight of the empty vehicle	kilograms (kg)
gross_weight	Decimal	Maximum total vehicle weight	kilograms (kg)
max_payload	Decimal	Max additional weight the vehicle can carry	kilograms (kg)
max_towing_capacity	Decimal	Max weight the vehicle can tow	kilograms (kg)

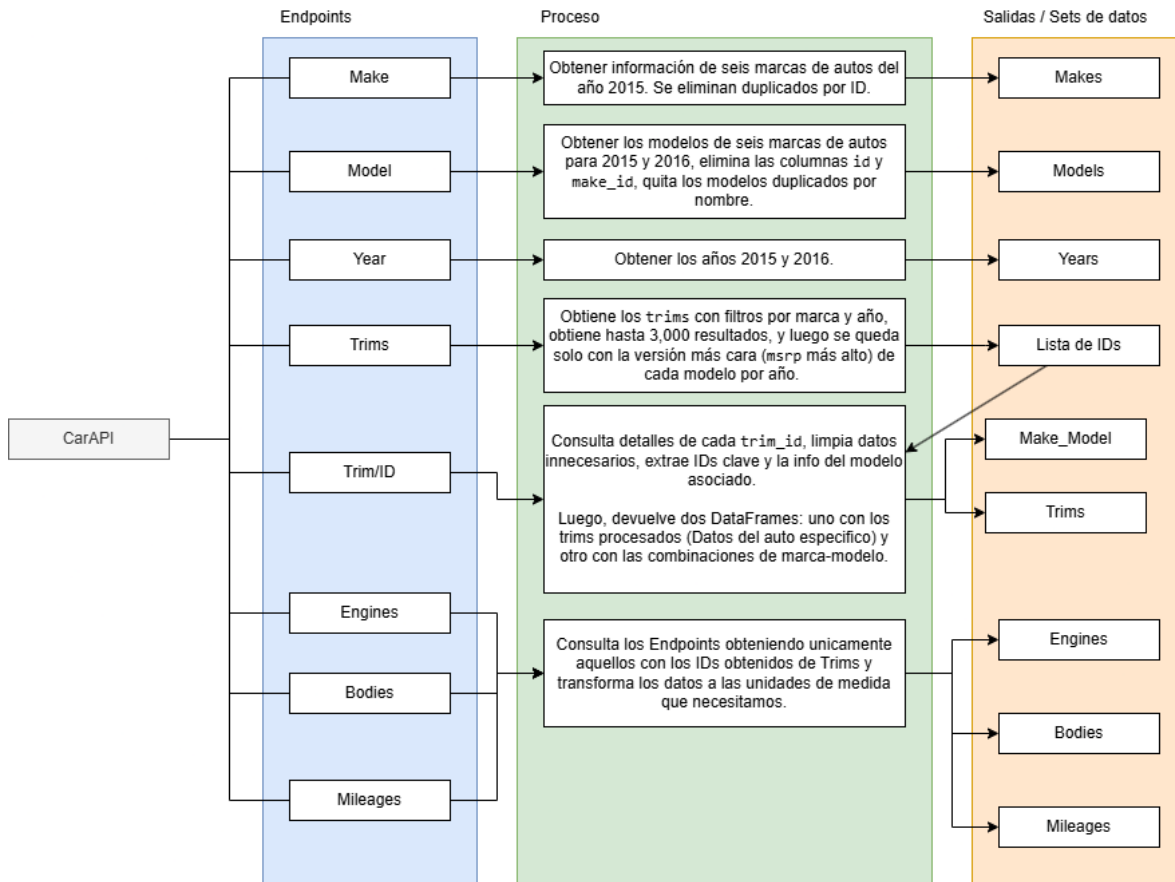
Trims			
Column	Data type	Description	Notes / Relationships
PK_id	Whole number	Unique identifier of the trim	Primary Key
FK_year	Whole number	Model year	Foreign Key → Years.id
name	Text (50)	Name of the trim or version (e.g., SE, XLE, GT-Line)	—
FK_engine_id	Whole number	Linked engine	Foreign Key → Engines.PK_id
FK_body_id	Whole number	Linked body type	Foreign Key → Bodies.PK_id
description	Text (50)	Short description of trim (may include doors, transmission)	—
price	Whole number	Manufacturer's suggested retail price (MSRP)	—
FK_make_model_id	Whole number	Linked model to which this trim belongs	Foreign Key → Make_Models.id
FK_mileage_id	Whole number	Linked mileage entry (fuel efficiency)	Foreign Key → Mileages.PK_id
estimatedKM	Whole number	Estimated kilometers driven (for market data or resale info)	Kilometers (km)

Mileages			
Column	Data type	Description	Units (metric)
PK_id	Whole number	Unique identifier of the mileage entry	—
fuel_tank_capacity	Decimal	Fuel tank capacity	liters (L)
combined_LitersAt100km	Decimal	Combined fuel consumption	L/100 km
city_LitersAt100km	Decimal	City fuel consumption	L/100 km
highway_LitersAt100km	Decimal	Highway fuel consumption	L/100 km
range_city	Decimal	Estimated driving range in the city	kilometers (km)
range_highway	Decimal	Estimated driving range on highways	kilometers (km)
battery_capacity_electric	Decimal	Capacity of electric battery	kilowatt-hours (kWh)
epa_time_to_charge_hr_240v_electric	Text (50)	Time to charge using 240V outlet	hours (text)
epa_kwh_100_mi_electric	Text (50)	Energy usage over 100 miles	kWh/100mi (textual)
range_electric	Text (50)	Estimated electric driving range	kilometers (km) (text)
epa_highway_mpg_electric	Decimal	EPA-rated electric efficiency on highway (if not converted)	MPGe
epa_city_mpg_electric	Decimal	EPA-rated electric efficiency in city (if not converted)	MPGe
epa_combined_mpg_electric	Decimal	EPA-rated combined electric efficiency (if not converted)	MPGe

Engines			
Column	Data type	Description	Units (metric)
PK_id	Whole number	Unique identifier of the engine	—
engine_type	Text (50)	Type of engine (e.g., gas, diesel, electric)	—
fuel_type	Text (50)	Type of fuel used	—
cylinders	Text (50)	Engine cylinder configuration (e.g., I4, V6)	—
size	Text (50)	Engine size or displacement	liters (L) (text)
horsepower_hp	Whole number	Engine power	horsepower (hp)
horsepower_rpm	Decimal	RPM at which max horsepower is delivered	revolutions/minute
torque_ft_lbs	Decimal	Torque	pound-feet (lb-ft)
valves	Decimal	Total number of engine valves	units
valve_timing	Text (50)	Valve timing type (e.g., Variable)	—
cam_type	Text (50)	Camshaft configuration (e.g., DOHC, SOHC)	—
drive_type	Text (50)	Drivetrain configuration (e.g., FWD, AWD, RWD)	—
transmission	Text (50)	Type of transmission (e.g., 6-speed automatic)	—

Obtención de datos

En la obtención de datos hemos diseñado un sistema de scripts se encargan de realizar un proceso de ETL, este proceso se encarga de obtener todos los datos de la API de automóviles, hicimos un modelo de datos y adaptamos los resultados a él, transformamos los datos de medidas imperiales a internacionales y también carga la base de datos en RDS desde Lambda. Adjuntamos un diagrama de cómo funciona:



Aplicación Web

Por la parte del front-end creamos una aplicación web que es muy sencilla e intuitiva de utilizar y además es responsiva a distintos tipos de dispositivos y muestra los datos que se requieren de nuestra base de datos en la nube dependiendo del tipo de vehículo seleccionado con los filtros.



Adjuntamos algunas capturas de nuestra aplicación:

CarWizard

your trusted pre-owned car comparison tool

Vehicle 1	Vehicle 2
<input type="text" value="Honda"/>	<input type="text" value="Toyota"/>
<input type="text" value="Civic"/>	<input type="text" value="Corolla"/>
<input type="text" value="2016"/>	<input type="text" value="2016"/>

COMPARE VEHICLES

Feature	Vehicle 1	Vehicle 2
Photo	 \$ 26,500	 \$ 23,125
Make	Honda	Toyota

Feature	Vehicle 1	Vehicle 2
Photo	 \$ 26,500	 \$ 23,125
Make	Honda	Toyota
Model	Civic	Corolla
Year	2016	2016
Trim	Touring	S Premium
Engine Type	gas	gas
Fuel Type	regular unleaded	regular unleaded
Cylinders	I4	I4
Displacement	1.50	1.80
Horsepower (HP)	174	132
Torque (ft-lbs)	162.00	128.00
Transmission	continuously variable-speed automatic	continuously variable-speed automatic
Drivetrain	front wheel drive	front wheel drive
Doors	4	4
Seats	5	5
Length	463.04	465.07
Width	179.83	177.55