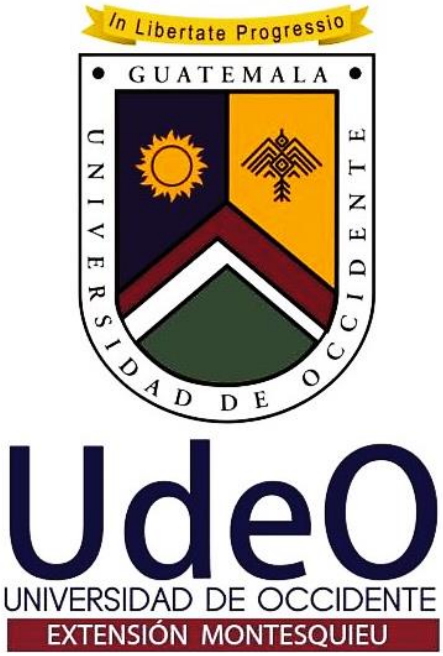


Manual Técnico- Parqueo AG

- Williams Coro 140402013
- Ludswing Cahuec 140402008
- Andersson Hernández 150402008

Contenido

Contenido

A continuación de describen todas las partes implicadas en el sistema.

Base de datos

La base de datos es del tipo relacional, se utilizó SQL Server 2017 en su versión 17.9., a continuación, se describen los detalles de la computadora utilizada.

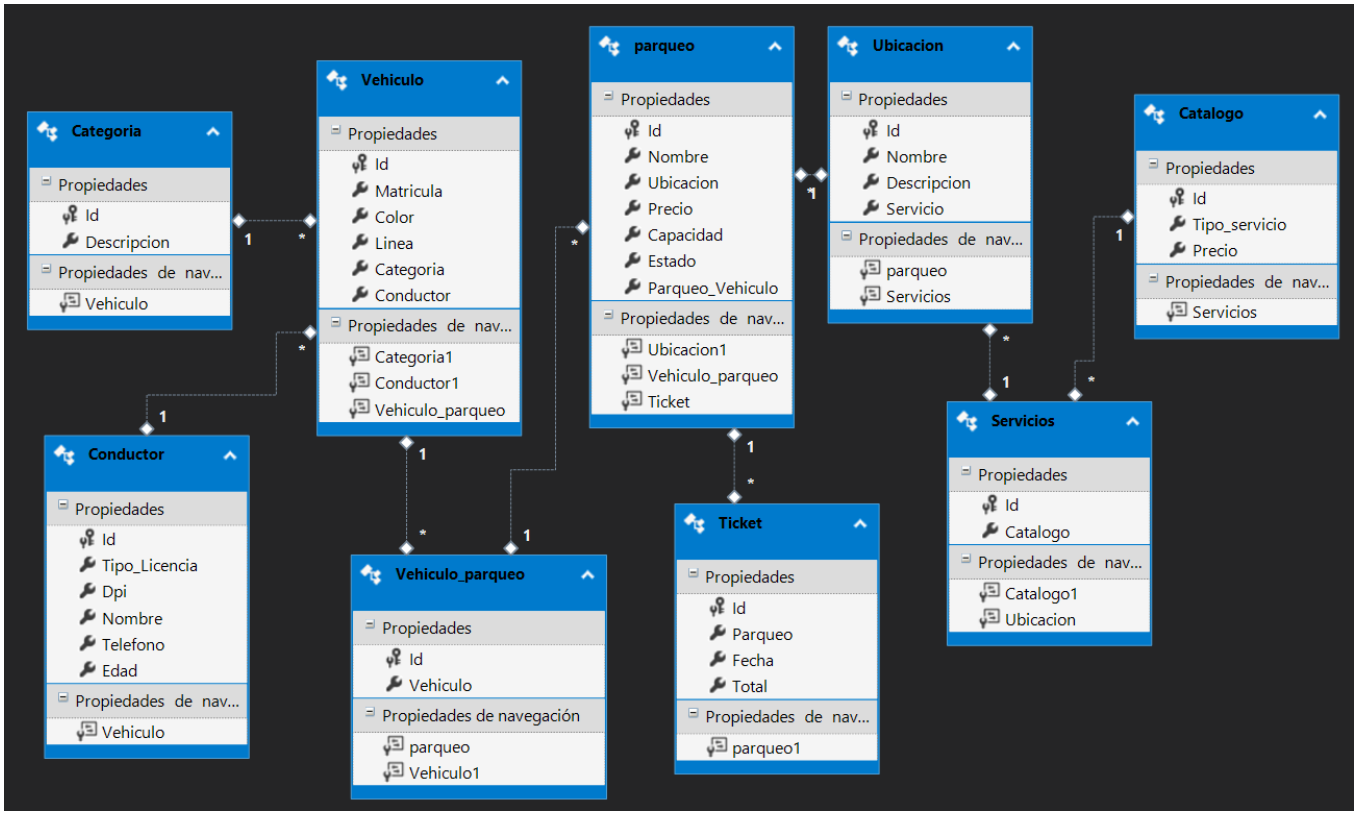
Disco Duro	1 TB
Memoria Ram	16 GB
Procesador	Intel i7 2.2 GHz
Tipo de Sistema	Sistema Operativo de 64 bits, procesador x64
Sistema operativo	Windows 10 Home Single Language

La base de datos cuenta con 9 tablas, las cuales se describen a continuación.

Categoría	Tabla donde se almacena las marcas de los vehículos.
Vehículo	Tabla donde se almacenan los detalles del vehículo, como placa, color, línea.
Vehículo_Parqueo	Tabla donde se le asigna un parqueo.
Parqueo	Tabla donde se almacena los detalles del parqueo tales como ubicación, precio, capacidad.
Ticket	Tabla donde se almacena los detalles del ticket como fecha, total, parqueo.

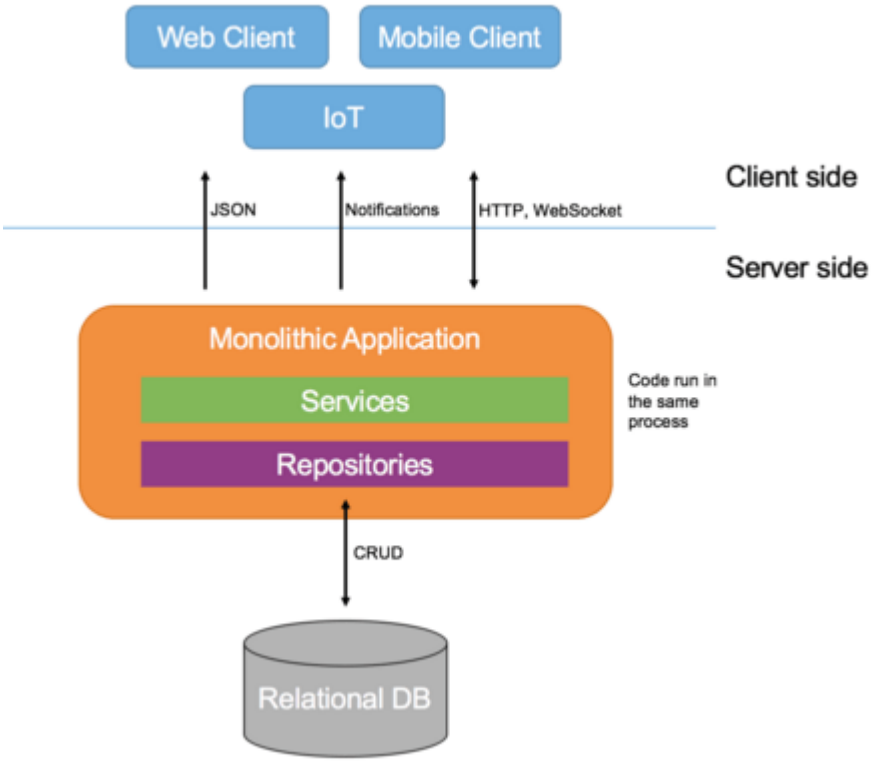
Conductor	Tabla donde se almacena la información del conductor como nombre, dpi, licencia, teléfono.
Ubicación	Tabla donde se almacenan los servicios dependiendo la ubicación.
Servicios	Tabla catalogo donde se almacena el tipo de servicios que existen.
Catalogo	Tabla catalogo donde se almacenan los precios de cada servicio.

A continuación, el diagrama de la Base de Datos:



API Parqueo

El api fue desarrollado utilizando el framework de programación .Net Core en su versión 2.1, el api es del tipo RESTful, a continuación, un pequeño diagrama de como se desplegó la aplicación.



Detalle de las funciones del api:

Tipo	URL	Headers	Descripción

GET	http://localhost:49822/api/VehiculoApi	public IQueryable<Vehiculo> GetVehiculo()
		{
		return db.Vehiculo;
		}
			Sirve para obtener todos los vehículos del sistema.

GET	http://localhost:49822/api/VehiculoApi/2	public IHttpActionResult GetVehiculo(int id)
		{
		Vehiculo vehiculo = db.Vehiculo.Find(id);
		if (vehiculo == null)
		{
		return NotFound();
		}
		
		return Ok(vehiculo);
		}
			Sirve para obtener un vehículo en específico recibiendo como parámetro el id.

PUT	http://localhost:49822/api/VehiculoApi/1	public IHttpActionResult PutVehiculo(int id, Vehiculo vehiculo)
		{
		if (!ModelState.IsValid)
		{
		return BadRequest(ModelState);
		}
		
		if (id != vehiculo.Id)
		{
		return BadRequest();
		}
		
		db.Entry(vehiculo).State = EntityState.Modified;
		
		try
		{
		db.SaveChanges();
		}
		catch (DbUpdateConcurrencyException)
		{
		if (!VehiculoExists(id))
		{
		return NotFound();
		}
		else
		{
		throw;
		}
			Sirve para poder cambiar un vehículo dentro del sistema.

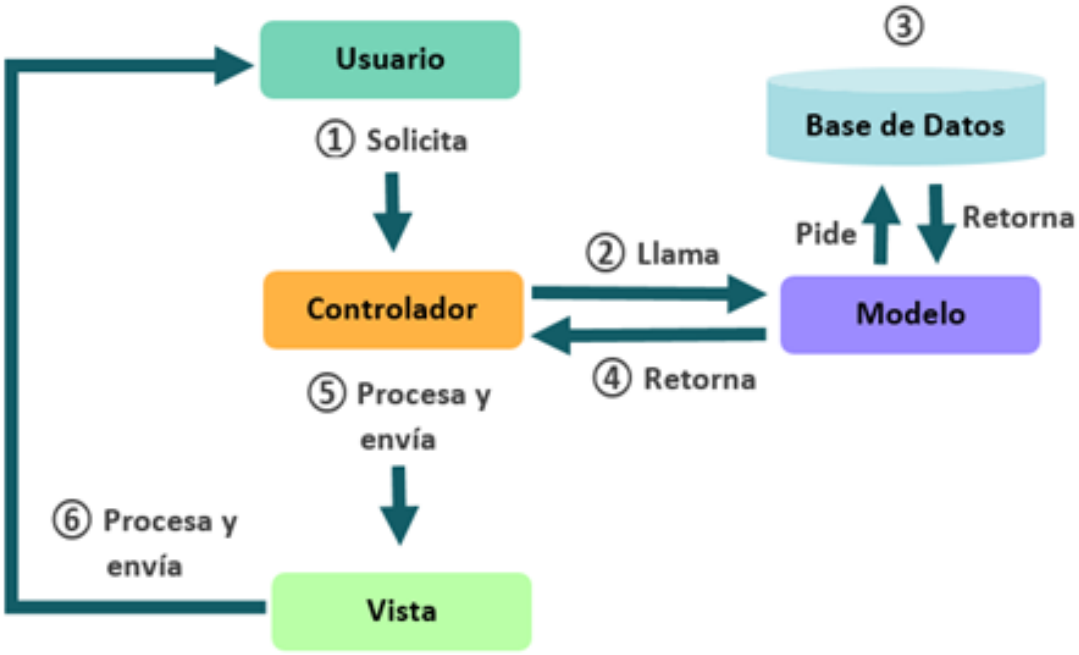
		}
		
		return StatusCode(HttpStatusCode.NoContent);
		}	

POST	http://localhost:49822/api/VehiculoApi	public IHttpActionResult PostVehiculo(Vehiculo vehiculo)
		
		{
		
		if (!ModelState.IsValid)
		
		{
		
		return BadRequest(ModelState);
		
		}
		
		db.Vehiculo.Add(vehiculo);
		
		db.SaveChanges();
		
		return CreatedAtRoute("DefaultApi", new { id = vehiculo.Id }, vehiculo);
		
		}
			Sirve para poder crear un nuevo vehículo en el sistema.

DELETE	http://localhost:49822/api/VehiculoApi/3	public IHttpActionResult DeleteVehiculo(int id)
		{
		Vehiculo vehiculo = db.Vehiculo.Find(id);
		if (vehiculo == null)
		{
		return NotFound();
		}
		
		db.Vehiculo.Remove(vehiculo);
		db.SaveChanges();
		
		return Ok(vehiculo);
		}
			Sirve para poder eliminar un vehículo en específico en el sistema

Web de Parqueo

La web fue desarrollada utilizando el framework de programación ASP.Net Core en su versión 2.1, con el patrón de diseño MVC (Modelo, Vista, Controlador), a continuación, un pequeño diagrama de como se desplegó la aplicación.

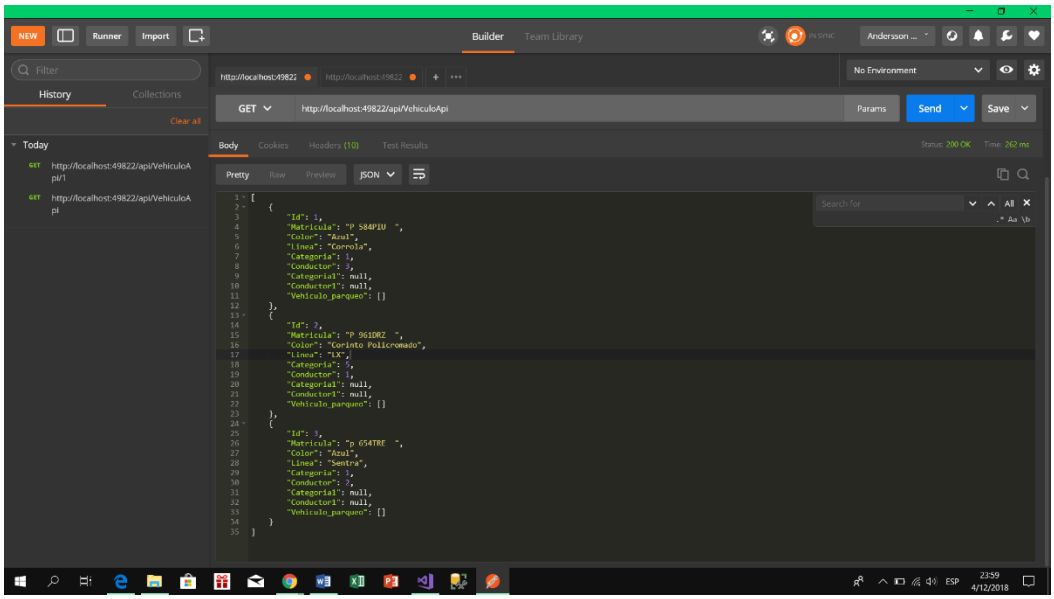


Herramientas utilizadas

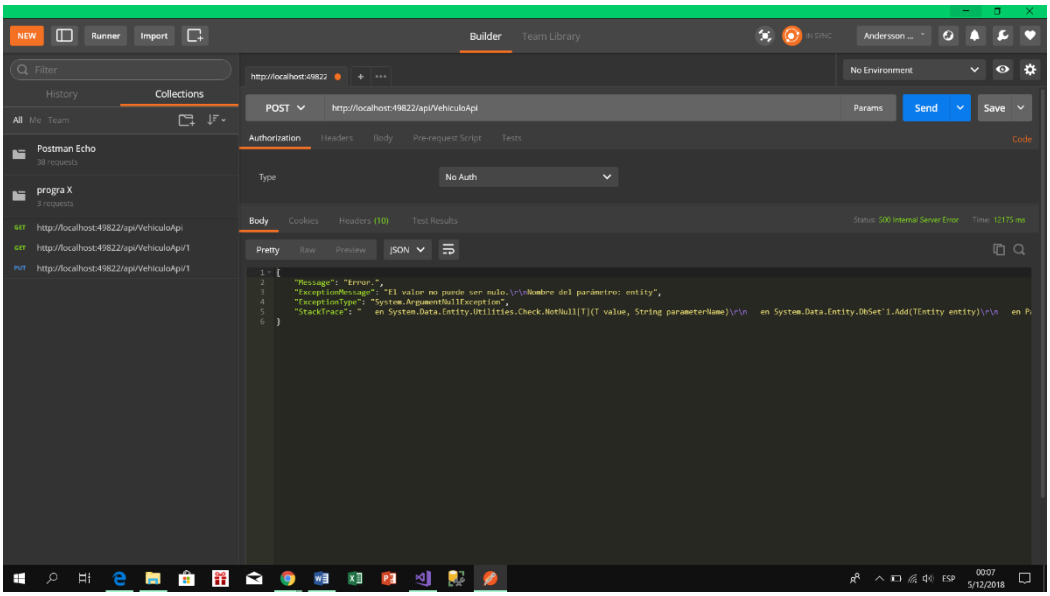
Postman:

Es una extensión del navegador Google Chrome, que permite el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente. Se utilizó para poder hacer las pruebas con el servidor. A continuación, las capturas con las pruebas.

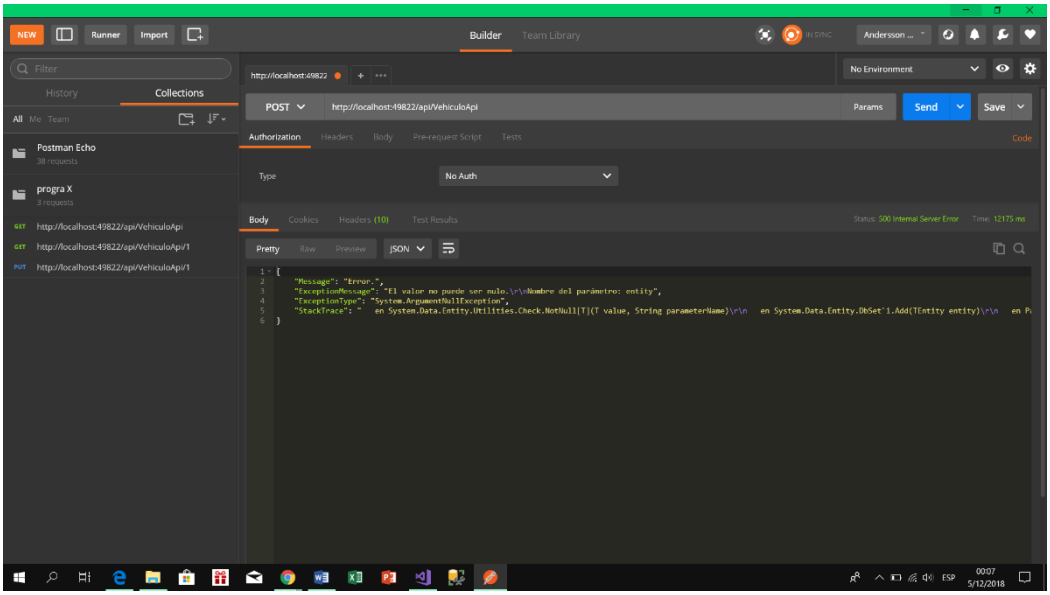
GET:



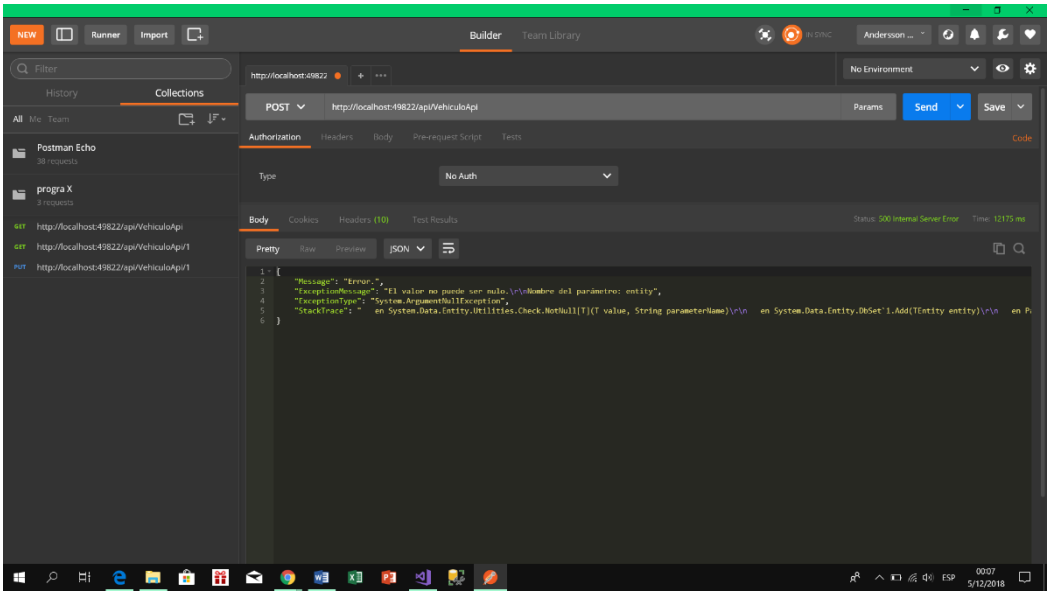
PUT:



POSTS:

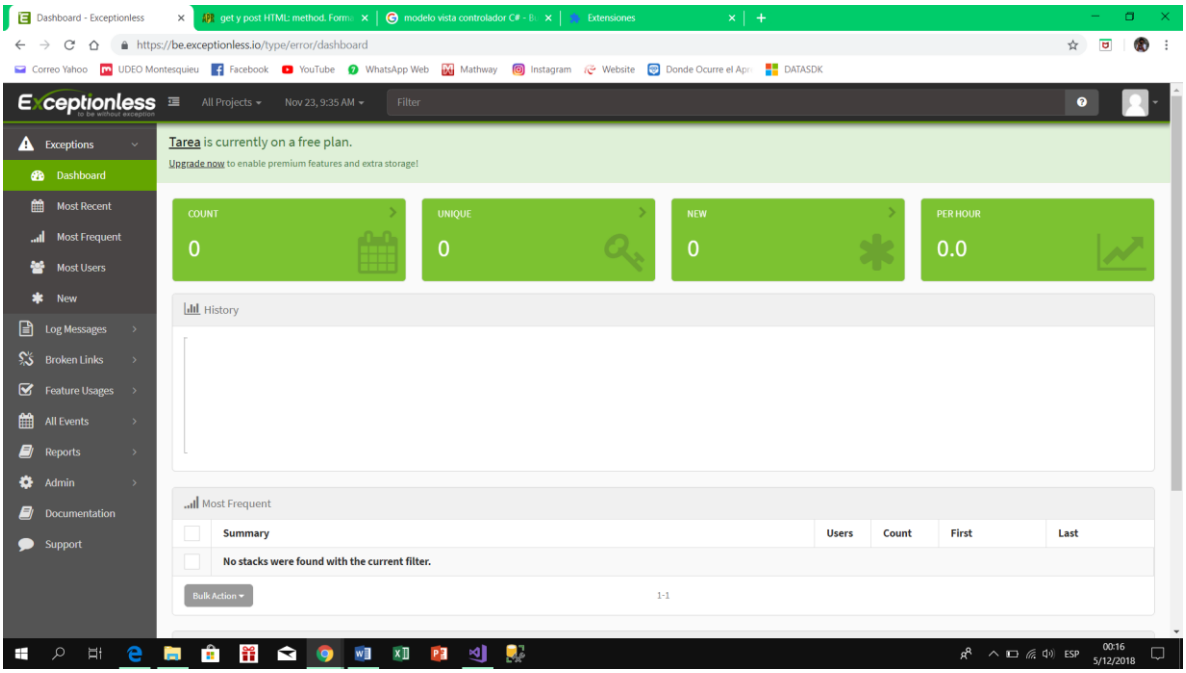


DELETE:



Exceptionless:

Es una técnica de programación que permite al programador controlar los errores ocasionados durante la ejecución de un programa informático, y para manejar esto se utilizó la herramienta Exceptionless en su versión Online.



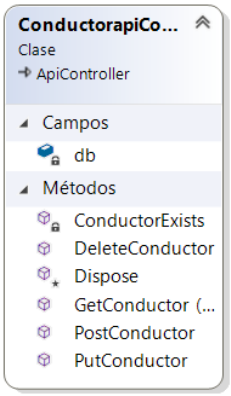
RabbitMQ:

Es un software de negociación de mensajes de código abierto, y entra dentro de la categoría de middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (AMQP). El servidor RabbitMQ está escrito en Erlang y utiliza el framework Open Telecom Platform (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores.

Diagrama de clases:

Generamos los diagramas de clases desde visual estudio, a continuación, los diagrama.

Este diagrama contiene la Api del conductor



Este diagrama contiene la Api del Parqueo

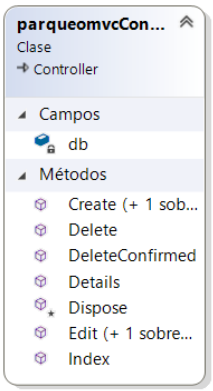


Diagrama con el Api del Parqueo/Vehículo

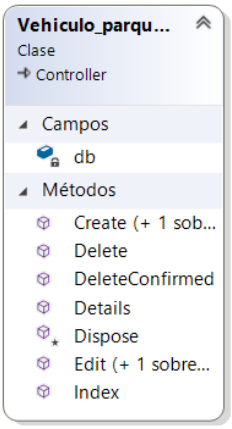


Diagrama que contiene el Api del Catalogo

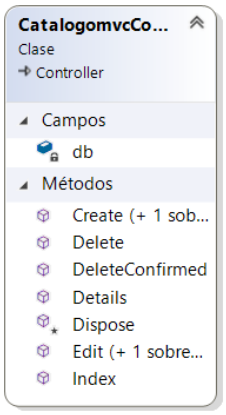


Diagrama que contiene el Api del Conductor

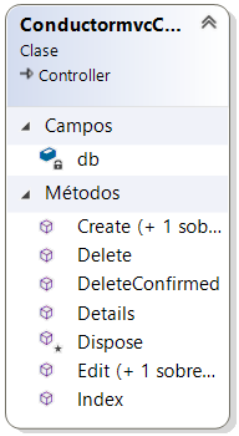


Diagrama que contiene el Api de los Servicios

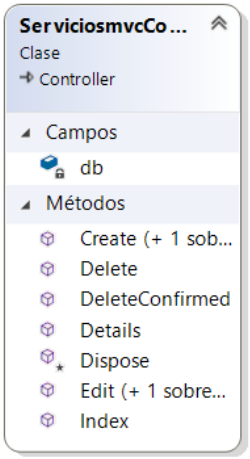


Diagrama que contiene el Api del Vehículo

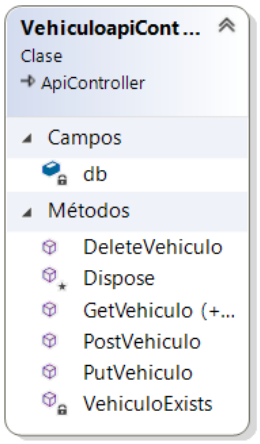


Diagrama que contiene el Api de la Categoría

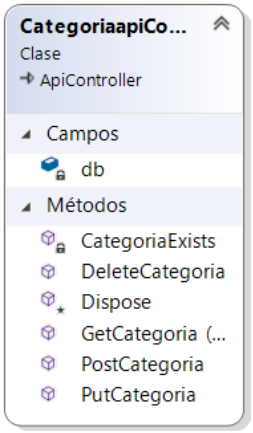
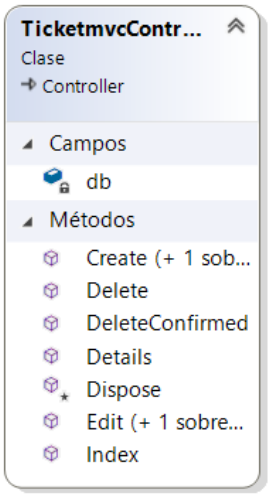


Diagrama con el Api del Ticket



Casos de uso:

Haremos un caso de uso para ver el ingreso del sistema al momento de registras un nuevo vehículo, a continuación, el diagrama.

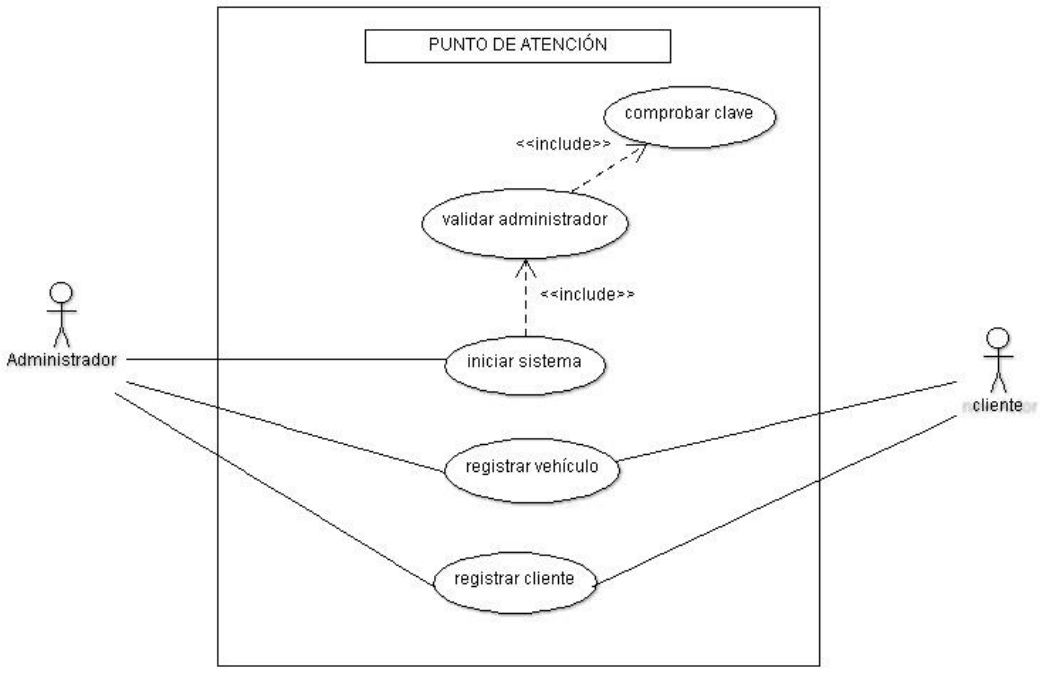


Diagrama de Flujo:

Se realizó un diagrama de flujo para ver de mejor forma la información que se manejar dentro del sistema.