809-865-3062
Perez.frank@gmail.com
SANTO DOMINGO, D.N.

F/P FRANK PÉREZ

# EXPLICACION DEL EXAMEN PRACTICO PARA CANDIDATOS A LA POSICIÓN DE BACK-END

Saludos Equipo evaluador de Concentra,

En primer lugar, quiero expresar mi agradecimiento por la oportunidad de demostrar mis habilidades para el puesto de Back-End.

La intención de este documento es proporcionar una explicación detallada de cómo se realizó el examen práctico y por qué se tomaron las decisiones que se tomaron al implementar la solución. También se incluye una breve guía sobre cómo descargar y utilizar el proyecto.

Gracias de antemano.
Frank Javier Pérez Veloz

## Proceso de Instalación:

1. Descargar el Código de Github.
2. Configurar la cadena de conexión (ConnectionString) en la ubicación deseada para la base de datos y asegurarse de proporcionar las credenciales adecuadas para el acceso a la misma. Este proceso debe realizarse en el archivo denominado "appsettings.json".
3. Correr proyecto en visual studio.

**NOTAS:** Este proyecto se ha desarrollado para utilizar Microsoft SQL Server como base de datos. Cuando se ejecute el programa, se creará automáticamente la base de datos junto con todas las entidades y relaciones necesarias para asegurar el funcionamiento correcto del mismo.

Es importante destacar que, para utilizar cada uno de los endpoints creados, es necesario generar un token. Este requisito surge del uso de JWT para las peticiones. Por lo tanto, se debe generar un token utilizando el endpoint de Login. Posteriormente, este token debe ser incluido como un "Bearer Token" en cada una de las solicitudes a los endpoints expuestos.

## Solución de Problema CASO 1:

## PRIMERA PARTE DEL PROBLEMA

**1.- Creación de una base de datos, normalizada en sql server**

- Creación de tabla de Usuario, para login
- Creación de tabla de Placas de vehículos, para el registro CRUD
- Creación de tabla de Registro de vehiculos, para el registro CRUD

## SOLUCION:

Cuando el aplicativo se ejecuta, se encuentra un archivo ubicado en **"./Migrations/20230930152507_DBInit.cs"**. Este archivo se ejecuta automáticamente al iniciar la aplicación y tiene la responsabilidad de crear las siguientes estructuras con el propósito específicado:

- Creación de tabla de Usuario, para login "La tabla creada **Users**"
- Creación de tabla de Placas de vehículos, "La tabla creada **Plates**"
- Creación de tabla de Registro de vehiculos, "La tabla creada **Vehicles**"

## SEGUNDA PARTE DEL PROBLEMA

Caso 1
Yo: Test
Quiero: Registrar solicitudes de Placas
Para que: Poder evaluar, vender una placa a un cliente X Criterios de aceptación:

**PRIMERA PARTE**

- Capturar toda la información requerida para el registro de una placa
  - Nombre
  - Apellidos
  - Cédula
  - Fecha de nacimiento
  - Tipo de Placa
  - Tipo Personas (Persona Física o Persona Jurídica)
  - Tipo automóvil (Público, Privado, Transporte y Pesado)
  - Valor Total Pagado Por placas

**SOLUCIÓN:**

El capturar toda la información requerida para el registro de una placa en una base de datos Normalizada implica la creación de varias entidades, para tal fin se crearon las siguientes entidades:

1. Clients (Esta entidad almacenara la información de los clientes).
2. PersonTypes (Esta entidad es de configuración que indicara tipos de persona).
3. Vehicles (Esta entidad llevara el registro de todos los vehiculos).
4. VehicleTypes (Esta entidad es de configuración para indicar los tipos de vehiculos).
5. Plates (Esta entidad llevara el registro de todas las placas emitidas por vehiculos).

6. PlatesRecords (Esta entidad se encargará de llevar el registro de todos los cobros por concepto de placa en distintos periodos).
7. Brands (Esta entidad es para registrar todas las marcas de vehículos que pueda aceptar el sistema).
8. Models (Esta entidad es para registrar todos los modelos de las marcas previamente definidas).
9. Users (Esta entidad es para almacenar un registro de los usuarios que tienen acceso al sistema).
10. Status (Esta entidad es para almacenar todos los estatus que se pueden manejar en el sistema).

Ahora que fueron explicadas todas las estructuras para capturar la información del registro de placa debemos dar los siguientes pasos para capturar el registro de Placa:

1. Buscar Cliente o Crearlo en caso de no existir, para esto se creó el endpoint: http://localhost:5157/api/clients
2. Buscar Vehículo o Crearlo en caso de no existir, para esto se creó el endpoint: http://localhost:5157/api/vehicleTypes
3. Buscar o Crear Placa en caso de no existir para esto se creó el endpoint: http://localhost:5157/api/plates

**Nota:** El uso de los endpoints anteriores implica la necesidad de emplear otros endpoints para proporcionar los datos requeridos en las solicitudes que los métodos esperan. Por ejemplo, al crear un cliente en el sistema, es necesario especificar el tipo de persona, ya sea física o jurídica, para estos casos se crearon los siguientes endpoints:

1. Buscar Tipos Persona o Crearlo en caso de no existir, Para esto se creó el endpoint: http://localhost:5157/api/personTypes
2. Buscar Tipo Auto o crearlo en caso de no existir, para esto se creó el endpoint: http://localhost:5157/api/vehicleTypes
3. Buscar Marca Auto o crearla en caso de no existir, para esto se creó el endpoint: http://localhost:5157/api/brands

4. Buscar Modelo Auto o crearlo en caso de no existir, para esto se creó el endpoint:

   http://localhost:5157/api/models

5. Buscar Usuario o crearlo en caso de no existir, para esto se creó el endpoint:

   http://localhost:5157/api/users

**NOTA: PARA PROBAR EL BACKEND DE MANERA MÁS RÁPIDA, SE HA CREADO UNA COLECCIÓN DE ENDPOINTS EN POSTMAN, LO QUE FACILITARÁ LA REALIZACIÓN DE PRUEBAS RÁPIDAS EN TODOS LOS ENDPOINTS.**

**SEGUNDA PARTE**

- Evaluar si procede la solicitud. En caso de proceder se debe generar el número de placa del cliente y el valor a cobrar. En caso contrario notificado la razón de porque no procedió.

**SOLUCIÓN**

Cada uno de los endpoints mencionados anteriormente cuenta con validaciones, ya sea a nivel de restricciones de la base de datos o validaciones específicas del endpoint, con el propósito de llevar a cabo sus respectivas funciones. Si todas las validaciones de cada uno de los endpoints son superadas con éxito, se habilita la posibilidad de llegar al punto final de la creación de una placa en el sistema y la generación del valor a cobrar.

El ultimo Endpoint a llamar es el de generar placa a un vehículo y se encuentra en la siguiente URL: http://localhost:5157/api/plates

**TERCERA PARTE**

- Un cliente puede tener varios tipos de placas, pero un vehículo solo puede tener una placa asignada.

**SOLUCIÓN**

El enfoque actual para abordar este punto se ha diseñado de manera diferente para aprovechar al máximo el análisis. Esto se debe a que, en realidad, el cliente no posee placas; en su lugar, posee vehículos, y cada vehículo puede tener una única placa asociada. Por lo tanto, un cliente puede tener varios vehículos, pero cada vehículo está vinculado a una sola placa, por lo que podemos obtener el mismo resultado que se solicita.

Esto fue manejado a través de constrains a nivel de base de datos.

**CUARTA PARTE**

- Cada Placa debe contener una letra y 6 números aleatorios que no se repitan.

**SOLUCIÓN**

Para abordar esta parte del problema se realizo a nivel un storeprocedure a nivel de base de datos para garantizar que se cumpla con este requisito, el store procedure se llamó “USP_GenerateUniquePlate”

**QUINTA PARTE**

- Las placas emitidas deben ser de acuerdo con el tipo de automóvil
    - Publico Comienzan con la letra A
    - Privado con la Letra F
    - Transporte con la T
    - Pesado con la Letra P

**SOLUCIÓN**

Para abordar este requisito, se trató de la siguiente manera: dado que el tipo de automóvil determina la letra inicial de las placas, se agregó una columna a la tabla "**VehicleTypes**" llamada "**PlateType**". Esta adición permitirá de manera sencilla determinar qué tipo de placa se asigna a un vehículo al seleccionar su tipo correspondiente.

## Solución de Problema CASO 2:

## PRIMERA PARTE DEL PROBLEMA

Caso 2
Yo: Test
Quiero: Consulta de Placas
Para que: Poder obtener la o las placas de un cliente para conocer cuales tiene registrado. Criterios de aceptación:

- Poder buscar por identificación del cliente.
- Debe retornar los siguientes datos
    - Tipo de Placa
    - Número de Placa
    - Datos del Cliente
    - Fecha de Venta
    - Valor Pagado **por placa**

## SOLUCION

Para cumplir con este requisito, se creó un procedimiento almacenado llamado "USP_getPlates". Este procedimiento toma como argumento el número de identificación de un cliente y devuelve toda la información solicitada. Además, se implementó un endpoint llamado "plateLookUp" que consume este procedimiento y proporciona la información requerida a través de la siguiente URL:

http://localhost:5157/api/plateLookUp

## REQUERIMIENTOS TECNICOS

**2.- Creación de proyecto back end, en C#. net core Api**

- Utilizar jwt para las peticiones
- Utilizar patrón de diseño
- Utilizar Automapper
- Creación de mantenimiento CRUD
- Utilizar paradigmas de programación, buenas prácticas "Clean code"

## SOLUCION

Cada uno de los requerimientos técnicos ha sido cumplido de acuerdo con las especificaciones solicitadas. Por favor, siéntase libre de verificar el código y no dude en ponerse en contacto si tiene alguna pregunta o solicitud adicional. Estamos a su disposición para cualquier consulta.

**NOTA: PARA PROBAR EL BACKEND DE MANERA MÁS RÁPIDA, SE HA CREADO UNA COLECCIÓN DE ENDPOINTS EN POSTMAN, LO QUE FACILITARÁ LA REALIZACIÓN DE PRUEBAS RÁPIDAS EN TODOS LOS ENDPOINTS.**