# Prueba Técnica desarrollador Front-End

## Primer Bloque

1) Realizar maquetación del diseño suministrado y tomar en cuenta:
   a. Todos los campos son requeridos.
   b. El editar los campos de tarjetas, fecha y nombre modifica el diseño de la tarjeta.
   c. El campo tarjeta solo puede contener números y un máximo de 16 caracteres.
   d. El campo fecha de vencimiento debe tener formato mm/yy.
   e. El campo fecha solo puede aceptar valores válidos para mes (01 a 12) y año (22 hasta año actual + 5).
   f. El campo Nombre titular solo puede contener letras y letras con tildes y máximo de 20 caracteres.
   g. En caso de que algún campo no sea válido se debe colocar texto en rojo debajo del campo indicando la causa de la invalidez.

2) Al pulsar botón de Agregar tarjeta.
   a. Debe agregar la tarjeta a un bloque en el cual se mostrarán los campos de tarjeta, nombre y fecha vencimiento.
   b. Para cada registro se debe contener un identificador único.
   c. Debe validar que el formulario sea válido y que al no ser válido despliegue los textos en cada campo que no es válido.
   d. El campo de número de tarjeta se debe mostrar enmascarado, solo mostrar los 12 primeros y 4 últimos dígitos (ej. 41********1234).

3) Al pulsar botón Cancelar se deben limpiar todos los campos.

## Segunda Bloque

1) Crear RESTful API para el procesamiento de agregado/visualización de tarjetas.
   a. Debe contener todos los métodos de CRUD. (aunque no los use todos)
   b. Puede utilizar cualquier Web API de su preferencia (ej. Firebase, ASP.Net Core o MVC 5 en local host, Express, etc.)
   c. Para la prueba no se tomará en cuenta ningún mecanismo de autenticación.

2) Conectar el primer bloque con el segundo bloque.
   a. Al momento de agregar tarjeta esta debe ser almacenada en alguna base de datos o archivo de texto (ej. Firebase).
   b. Lo único que se debe validar al momento de agregar es que los campos requeridos sean enviados.
   c. Los API deben de devolver correctamente respuestas HTTP para cada caso, (ej. 404, 500, 302, 200, etc.).

# Documentación

## Información de lo realizado.

| Bloque | Sección | Realizado |
|---|---|---|
| Primer Bloque | **1** | **A, b, c, d, e, f, g** |
| | **2** | **A, b, c, d** |
| | **3** | **3** |
| Segundo Bloque | **1** | **A, b, c** |
| | **2** | **A, b, c** |

Para el desarrollo de esta aplicación se tomó en cuenta ReactJS que es uno de los Frameworks de JavaScript para realizar la parte del frontend, Asp.NET Web APIs para el backend y SQL Server para la base de datos.

## Frontend

Se utilizo React JS que es uno de los Frameworks de JavaScript que facilita la creación de interfaces usuarios en una sola página.

El desarrollo está organizado en los siguientes módulos:

- **Component**
    - La carpeta **image**: contiene las imágenes utilizadas para la vista de la tarjeta.
    - **Style.css:** para los estilos css.
    - **AgregarTarjeta.js:** en el cual contiene:
        - El formulario para agregar las tarjetas.
        - validationsForm, que valida el formulario. Para la validación se utilizó las expresiones regulares y un poco de código para validar los campos.
        - initialForm, donde se declaran las variables que se utilizaran para almacenar la información.

    - **ListarTarjeta.js:** muestra el listado de las tarjetas gradadas en la base de datos.
        - componentDidMount: para llamar los datos de un api cuando se carga la página. Se utilizo Fetch para guardar los campos en el api.

- **VistaTarjeta.js:** muestra una vista previa de la tarjeta al momento de llenar los campos. Esta recibe el numeroTarjeta, fechaVencimiento, nombreTitular y cvv.

- **Hooks:** esta carpeta contiene
  - **useForm:** para controlar los estados, los errores y el funcionamiento del formulario que se encuentra en el archivo AgregarTarjeta.js. Este contiene los siguientes métodos:
    - handleChange: para el cambio del valor de las variables.
    - handleBlur: para manejar el foco de un elemento.
    - handleSubmit: para guardar los datos del formulario al api. Se utilizo Axios para guardar los campos en el api.
    - handleCancel: para cancelar el envió del formulario.

## Backend

En la parte del backend se utilizó Asp.NET Web APIs que es un marco para compilar api web en . NET Framework. El desarrollo está organizado en los siguientes módulos:

- **Controller**
  - **TarjetasController.cs:** es el controlador que contiene los métodos del crud. Este contiene los métodos:
    - GetTarjeta: el cual devuelve la lista de tarjetas. Además, si se le agrega el parámetro id, devuelve una tarjeta.
    - PostTarjeta: el cual agrega la información de los campos a la base de datos.
    - PutTarjeta: modifica los datos de una tarjeta en la base de datos.
    - DeleteTarjeta: elimina una tarjeta de la base de datos.
- **Models:** contiene la conexión a la base de datos. Este modelo fue generado con Entity Framework.
- **Validaciones:** esta carpeta contiene todas las validaciones del api.
  - **ValidacionTarjeta.cs:** valida los datos de los métodos del controlador TarjetasController.cs. Los métodos son : ValidarNumeroTarjeta,

ValidarFechaVencimiento, ValidarNombreTitular y ValidarCVV, los cuales reciben un parámetro de tipo string y devuelven una respuesta.

- **ViewModel:** contiene los objetos que se le muestra al usuario final.
    - **TarjetaViewModel.cs** esta clase contiene cuatro variables tipo string: numeroTarjeta, fechaVencimiento, nombreTitular y cvv.

## Instrucciones de como correr y probar el proyecto.

### Backend

Para correr el api primero debe tener la base de datos guardada en SQL Server, la cual podrá encontrar en la carpeta TarjetaDB y luego ejecutar el proyecto.

### Frontend

- Debe abrir la carpeta TarjetaApp en Visual Studio Code.
- Abrir el terminal
- Ejecutar el comando NPM start