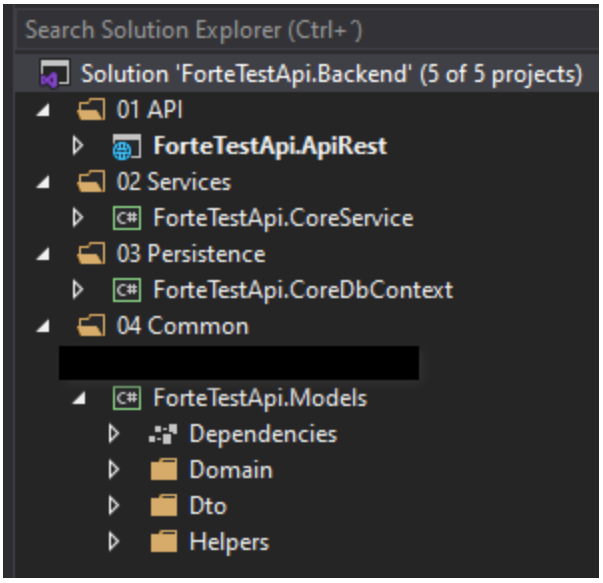
PRUEBA TÉCNICA DESARROLLADOR FULLSTACK .NET SR.

BACKEND

1. **Crear una solución de Visual Studio en raíz (C:\) con el siguiente nombre ForteTestApi.Backend que contendrá la siguiente arquitectura con proyectos en .NET Framework 4.6+ (4.7 de preferencia):**
    a. ForteTestApi.ApiRest
        i. De tipo ASP.NET Web API 2
            1. Crear una cadena de conexión a una base de datos local en el web.config.
    b. ForteTestApi.CoreService
        i. De tipo Librería de clases.
    c. ForteTestApi.CoreDbContext
        i. De tipo Librería de clases
    d. ForteTestApi.Models
        i. De tipo librería de clases que contenga las siguientes carpetas
            1. Domain
            2. Dto
            3. Helpers



2. **En el proyecto ForteTestApi.Models -> Helpers crear las siguientes interfaces**
    a. ISoftDeleted: La cual tendrá la siguiente propiedad:
        i. Eliminado (booleano)
    b. IAuditable: La cual tendrá las siguientes propiedades:
        i. CreadoPor (texto)
        ii. CreadoEl (fecha)
        iii. ModificadoPor (d texto)
        iv. ModificadoEl (fecha, y permita tener valor null)
        v. EliminadoPor (texto)
        vi. EliminadoEl (fecha, y que permita tener valor null)

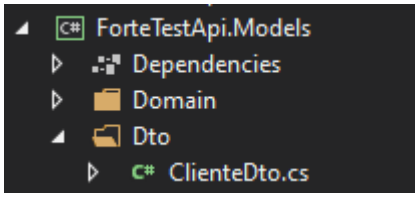
3. **En el mismo proyecto ForteTestApi.Models -> Helpers crear la siguiente clase:**
    a. PersonaFisica: La cual tendrá las siguientes propiedades:
        i. NombreCompleto (texto)
        ii. FechaDeNacimiento (fecha)
        iii. Edad (short, de solo lectura y calculada en base a la siguiente fórmula: Año Actual – Año de la fecha de nacimiento)

4. **En el proyecto ForteTestApi.Models-> Domain crear las siguientes clases**
    a. EstatusCliente: con las siguientes propiedades
        i. EstatusClienteId (byte)
        ii. Descripción (texto)
    b. Cliente: heredar de la clase PersonaFisica creada anteriormente, implementar las interfaces ISoftDeleted y IAuditable, además dar de alta las siguientes propiedades (ver diagrama UML adjunto).
        i. ClienteId (entero)
        ii. CorreoElectronico (texto)
        iii. Password (texto)
        iv. Domicilio (texto)
        v. LimiteCredito (número decimal)
        vi. EstatusClienteId (byte)
        vii. EstatusCliente (EstatusCliente)
    c. Regresar a la clase EstatusCliente y agregar una propiedad de tipo Lista, Colección o similar de Cliente indicando la relación uno a muchos (un estatus puede contener muchos clientes), inicializar esta lista en el constructor de EstatusCliente.

5. **En el proyecto ForteTestApi.CoreDbContext**
    a. Instalar EntityFramework.
    b. Crear el contexto de la aplicación a través de una clase llamada ForteDbContext (que herede de DbContext) y que apunte a la cadena de conexión creada en el proyecto ForteTestApi.ApiRest.

c. Dar de alta los DbSet de las entidades EstatusCliente y Cliente (recuerda agregar las referencias correspondientes)
d. Habilitar migraciones a través de la consola de NuGet
e. En el método seed insertar 3 registros para la tabla EstatusPersona que serán:
    i. 1 –ACTIVO
    ii. 2 –PENDIENTE ACTIVACIÓN
    iii. 3 -INACTIVO
f. Agregar una primera migración con el nombre InitialModel y ejecutarla en la base de datos a través del comando correspondiente.

6. **En el proyecto ForteBack.CoreService**
    a. Agregar referencia del proyecto ForteTestApi.CoreDbContext y a ForteTestApi.Models.
    b. Instalar de igual manera EntityFramework
    c. Crear una clase llamada ClienteDto dentro de una carpeta llamada Dto (dentro del proyecto ForteTestApi.Models) que tendrá las siguientes propiedades:
        i. ClienteId (int)
        ii. NombreCliente (string)
        iii. CorreoElectronico (string)
        iv. FechaNacimiento (DateTime)
        v. EstatusCliente (string)
        vi. LimiteCredito `(decimal)`



    d. Crear una clase llamada ClienteService en la raíz del proyecto.
    e. Dentro de ClienteService, inicializar un miembro (variable) de tipo ForteDbContext llamado `_forteDbContext` que sea **privado y de solo lectura** e inicializarlo a través del constructor creando una nueva instancia

```
_forteDbContext = new ForteDbContext();
```

    f. Crear los siguientes métodos
        i. AddCliente: publico, no devuelve nada y recibe como parámetro un objeto de tipo cliente.
            1. Validar que en la base de datos no exista un cliente con el mismo correo electrónico y en caso de que si, lanzar una Excepción de tipo InvalidOperationException cuyo mensaje

sea que ya existe un cliente registrado con ese correo electrónico.
2. Antes del guardado asignar la propiedad CreadoEl con la fecha actual del servidor.
3. El método debe estar anidado dentro de un **bloque try-catch**

ii. GetClientes: No recibe nada como parámetros y devuelve **una lista de clientes activos** a través de un IEnumerable de ClienteDto `IEnumerable<ClienteDto>`.

1. Con cliente activo nos referimos a todos aquellos cuyo valor Eliminado = false.
2. El método debe estar anidado dentro de un bloque try-catch.

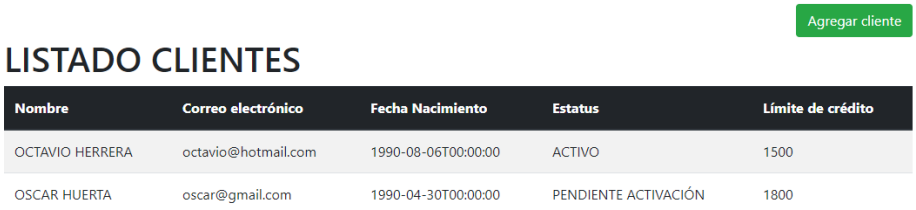
g. Dichos métodos realizarán las operaciones a través de EntityFramework y usando expresiones lambda (eager loading) o consultas LINQ

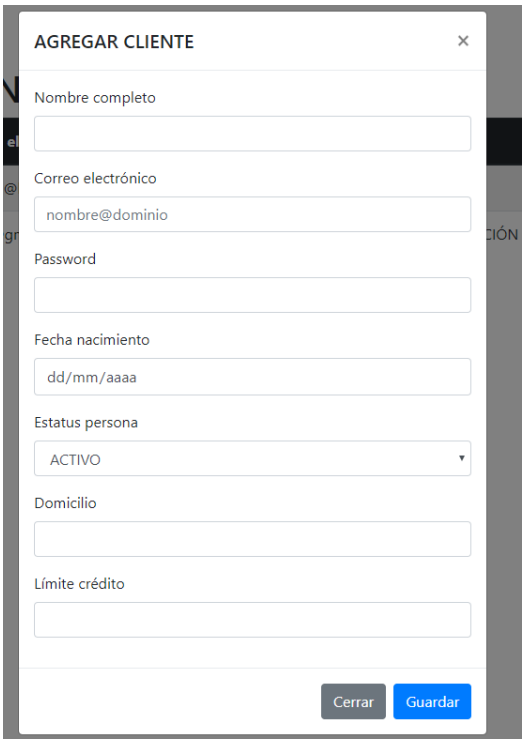
7. En el proyecto ForteTestApi.ApiRest

a. Referenciar el proyecto ForteBack.Services y ForteBack.Common
b. Instalar EntityFramework
c. Habilitar CORS (Cross-origin resource sharing) para todas las llamadas API desde otro dominio instalando el paquete de Nuget Correspondiente e insertando la configuración en la clase WebApiConfig.
d. Generar un controlador de tipo WEB API 2 en blanco para Cliente
   i. Crear un miembro (variable) de tipo ClienteService **privado de solo lectura** e inicializarlo dentro del constructor del controlador.
   ii. Generar un método GET llamado GetClientes que extraiga el listado de clientes de la base de datos y cuya ruta de llamado de la API sea: [url servidor]/api/clientes ejemplo: http://localhost:5503/api/clientes
   iii. Generar un método POST llamado AddCliente que agregue un cliente a la base de datos y cuya ruta de llamado de la API sea
   iv. [url servidor]/api/cliente ejemplo: http://localhost:5503/api/cliente
e. Los llamados de obtención y guardado de datos deberán ser a través del servicio ClienteService.
f. El controlador deberá notificar sobre errores en el servidor (500), acceso denegado (403) o similares.

FRONTEND

1. Crear una solución en Visual Studio Code o el editor de su preferencia que contenga una interfaz con **bootstrap 4** para la administración de los clientes.
2. Deberá contener un fichero index.html donde se despliegue una tabla con el listado de clientes con las siguientes columnas: Nombre completo, correo electrónico, Fecha de nacimiento, Estatus y límite de crédito y un botón para agregar en la parte superior.
3. Crear el formulario de clientes que contenga los siguientes campos:
    a. Nombre completo(texto), Correo electrónico (email), Password (password), Fecha de nacimiento (date), Estatus (dropdown hardcodeando los valores de EstatusPersona), domicilio (texto), Límite Crédito (número).
    b. Dentro del formulario de clientes, los siguientes campos son requeridos:
        i. Nombre
        ii. Correo Electrónico
        iii. Password
        iv. Fecha de nacimiento
        v. Estatus
4. El formulario se mostrará a través de un Modal.
5. Utilizar de preferencia Angular 2+ (8 deseable), caso contrario ÁngularJS, Vue o React para generar las peticiones al servidor y desplegar la información.
6. Las peticiones deberán ser a través de ajax y promesas indicando debidamente cualquier mensaje de error en la validación de los datos, mensaje de guardado con éxito o si ocurrió un error en la transacción, los errores se pueden mostrar a través de alerts, modals o como el desarrollador lo prefiera.





FOR TECHNOLOGY.... FOR LIFE

Blvd Jorge Vertíz Campero 1640 Int UIAC 305 H,I,J,K.
Col. Cañada de Alfaro C.P. 37238 León Gto.

Tel: +52 477 710 0600 ext 1518 y 1525
forteinnovation.mx