Área académica de ingeniería en computadores.

Curso: Bases de datos.

Tarea corta 1: EnFeriaTec

Entrega: miércoles 21 de octubre.

Profesor: Msc.Marco Rivera Meneses

Miembros:

Jonathan García Ugalde.
Ronny Aguilar Barahona.
Roger Mora Gonzalez

Descripción de los métodos implementados.

Backend (Controller)

El la sección que respeta al API se manejó segmentando el encapsulamiento de los métodos divididos en clases que iban a permitir la comunicación con el protocolo HTTP y sus métodos get, post, put y delete.

Customer Controllers

IEnumerable<Customer> getCustomers(): Este método se encarga de obtener todos los valores de los clientes almacenados en un JSON .

IHttpActionResult SetCustomer(Customer model):Este método mediante el protocolo HTTP recibe una serialización de un objeto tipo,en este casos toman los valores y se insertan en los valores JSON mediante un Constructor. Este método devuelve un mensaje al frontend cuando se haya insertado de forma correcta.

IHttpActionResult Update(Customer ncustomer): Este método se encarga de actualizar los valores JSON de algún agricultor en específico, al finalizar mediante un mensaje le notifica que la solicitud se realizó con éxito.

Enumerable<string> CustomerNames(): método que devuelve los el nombre de todos los clientes que se insertaron.

Estos métodos pueden realizar interacción con la Clase Farmer, para acceder mediante el encapsulamiento de los datos. Estos métodos son los siguientes:

id { get; set; } : gestiona la inserción y obtención del Número Cédula del cliente

name { get; set; } : gestiona la inserción y obtención del Nombre del cliente.

lastName { get; set; }: gestiona la inserción y obtención del Apellidos

address { get; set; } : Dirección (Provincia, Cantón, Distrito deben estar por defecto)

DateTime birth; : gestiona la inserción y obtención del Fecha de nacimiento del cliente.

phone { get; set; } ;gestiona la inserción y obtención del Teléfono del cliente.

public string userName { get; set; } :gestiona la inserción y obtención del Usuario del cliente.

public int password { get; set; } : gestiona la inserción y obtención de Contraseña del cliente.

Farmer Controller:

IHttpActionResult Update(Farmers farmers):Este método se enca se encarga de poder realizar una actualización de datos de los agricultores en caso de que este
necesitará poder realizar algún cambio, este primero debe poder consultar si en este se encuentra registrado en la base de de datos en caso contrario este devuelve un mensaje de error debido a que el agricultor no se encuentra registrado

IEnumerable<Farmers> FarmerInfo();
haciendo uso de la "api/farmers", este método se encarga de enviar todos los agricultores registrados mediante el protocolo http para poder realizar diversas consultas

IHttpActionResult SetFarmer(Farmers model): Mediante la ruta de la ruta del post de protocolo se insertan los valores haciendo uso del constructor de farmer's el cual se carga y se actualiza, posterior a ello se serializa en formato JSON y se inserta.

Product Controller: Al igual que en la clase Farmer Controller, ésta cumple con la misma funcionalidad de insertar, actualizar y borrar clientes de la base de datos JSON, haciendo uso de la clase Product que funciona como cascarón para los objetos de tipo Product.

Entre los métodos encapsulados se encuentran :
ListAllProducts(): devuelve los productos almacenados.

Enumerable<string> Products(): devuelve el número de agricultores almacenados.

CategoryManagement: esta clase cumple con las funciones de añadir categoría, actualizar y borrar categoría, en donde mediante esta, se clasifican los productos que se insertan

Backend (Model)

Se creó un proyecto en Angular. La página se divide en varios componentes, donde cada componente corresponde a una vista o “sub-página”. Estos componentes tienen su propio directorio, un archivo css donde se ponen estilos únicos para esa vista, un archivo ts desde donde se consume la información desde el web service, y un web service encargado de consumir la información directamente desde el api en formato Json.

Además se tiene un módulo routing, encargado de trabajar el redireccionamiento interno de la página (urls y extensiones). Cada vez que se ingresa una extensión no reconocida, se devuelve a /home.

Descripción de las estructuras de datos desarrolladas

Se implementa el uso de una lista enlazada de forma genérica que nos brinda la oportunidad de poder hacer uso de diversos tipos de estructuras es decir que se usó para crear lista de objetos de tipo Cliente, Agricultor, transacción y Productos, donde estas se utilizaron para la serialización y deserialización de objetos JSON. para poder actualizarlos.

Esto ayudó a poder acceder de una forma más fácil a los datos de JSON en el momento de que estos se actualizaban o borraban. por ejemplo si se tuviera tres productos insertados en stock y se tuviera que vender el de en medio, en lugar de crear una estructura compleja para leer el JSON esta se encarga de darme el nodo con el atributo (id) que busco y lo borrar de forma eficiente y luego lo serializado y lo vuelvo a pasar a formato JSON.

.

Problemas conocidos:

Al momento de redacción de esta documentación, se presenta un error en el almacenamiento de los JSON, son de a la hora de insertar un nuevo elemento en cualquier de los archivos .json se borran algunos de los elementos que se insertan.

Problemas encontrados:

Error: connect ECONNREFUSED 127.0.0.1:4438: Luego de una serie de reiterados intentos fallidos de solución, el problema de los cores, iba más allá de haber programado mal el api, era una versión de c# que estaba teniendo incompatibilidad con la que se había creado inicialmente, para poder solucionarlo, fue necesario volver a crear otro proyecto que esta vez si tuviera contemplado el parche que solucionara el problema con los cores que en la sección del BackEnd(model) se estaba viendo reflejada, posterior a ello se migró toda la estructura que le daba forma al API (application Programming Interface) posteriormente se comprobó con postman su correcto funcionamiento.

Microsoft.CSharp.RuntimeBinder.RuntimeBinderException:

Documentación que evidencie el trabajo en equipo:
Bitácora de trabajo (reuniones en equipo):

Fecha	Duración	Asistencia	Temas
27/09	2.5 horas	Roger Mora Ronny Aguilar	Se discutió el diseño de la vista productores y login. Se probó el correcto funcionamiento de GitKraken como manejador de versiones. Se realizaron pruebas de acople entre model y view.
			Errores actuales: redireccionamiento de páginas internas.
8/10	1 hora	Roger Mora Ronny Aguilar	Merge de view/model: En esta reunión se trabajó en las plantillas html. Se lograron agregar al proyecto en Angular, correr las plantillas dentro del proyecto, y trabajar con el redireccionamiento interno de la página web.
11/10	3 horas	Jonathan Garcia Roger Mora	Problemas con el API En la reunión se observó el progreso de la construcción del API, se presentaban problemas de conexión. Inicialmente los endpoints estaban mal hechos, por lo tanto a la hora de abrir el localhost no se desplegaba ninguna información. Se logró acceder al JSON de los

			productos y los usuarios-productores y leer la información desde localhost:puerto/api/ruta.
14/10	3 horas	Jonathan Garcia Roger Mora	Se procedió reunirse debido a problemas con la versión del c# en cuanto al acceso de las carpetas donde se almacenaban los JSON en la base de Datos, el error encontrado fue a la hora de almacenar los valores de los JSON que formaban parte de las inserciones de nuevos valores.

Asignación de tareas:

Parte del proyecto	Tarea	Encargado
Model	Crear el proyecto en angular.	Roger Mora
	Añadir las plantillas, estilos y scripts al proyecto.	Roger Mora
	Redireccionar las páginas.	Roger Mora
	Crear el consumidor del API.	Roger Mora
	Acceso de cada persona a la página (users).	Roger Mora
View	Crear vista de Login	
	Crear modal para crear cuenta	Ronny Aguilar/Roger Mora
	Crear vista de Productos General	Ronny Aguilar

	Crear vista de Productos	Ronny Aguilar
	Crear Vista de Pago	Ronny Aguilar
	Crear vista de Confirmación Pago	Ronny Aguilar
	Crear Vista de Contáctenos	Ronny Aguilar
Controller	Crear proyecto de backend en c#	jonathan García
	Crear estructuras de métodos para comunicar web services(API) con api consumer	jonathan García
	Crear gestor de almacenamiento	jonathan garcía
	crear un webServices	Jonathan García
	almacenamiento de	Jonathan Garcia

Conclusiones y recomendaciones del proyecto.

Al dar inicio al proyecto, se tenía una expectativa muy grande de lo que consistía desarrollar una página web que permitiera al grupo de trabajo obtener o depurar las habilidades de desarrollo y tal y como se pensó

El uso de protocolo **HTTP** como método de comunicación hizo que la se abriera la curva de aprendizaje al saber que no solo se podía trabajar con *sockets* para comunicar programas entre los lenguajes de programación sino que con una herramienta como web services está permitió migrar datos desde un servidor a un cliente haciendo uso de métodos como get, post,put y delete.

Este proyecto, retó al grupo de desarrollo debido a que la mayoría de los miembros tuvo que adaptarse a un estilo de programación basado en el model view controller, del cual las herramientas de uso eran nuevas, por esa razón en este trabajo se aprendió a hacer uso de herramientas como **Angular, Bootstrap**, **HTML5, CSS3, Crystal Report**, las cuales en el quehacer laboral se están usando de forma masiva.

Entre las recomendaciones que se pueden dar de este proyecto, se destaca que, a futuras tareas se evalúe si es realmente necesario la implementación de un versión móvil, esto porque para este proyecto se solicitaba la implementación bootstrap y como es sabido este tiene la particularidad de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo, y en este caso podría ser redundante crear un app desde cero, sabiendo que gracias a este framework el consumo de recurso es mínimo y muy eficiente.

Bibliografía consultada para el proyecto.

AngularJS (2019-02-05). Recuperado de: https://angular.io/

Bootstrap Themes & Templates (2019-02-05). Recuperado de: https://getbootstrap.com/docs/4.1/getting-started/introduction/

Hasan, J. (2004). *Expert Service-Oriented Architecture in C#*. Apress.

RazorCX Technologies: https://github.com/razorcx/json-example/blob/master/JsonExample/JsonExampleForm.cs

Singer, A. (2011). Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, *30*(1), 20-36.