* + Descripción de los métodos implementados.
  + Descripción de las estructuras de datos desarrolladas (no utilizar la definición de las estructuras de datos desarrolladas).

Con el fin de integrar la base de datos utilizada con Angular y sus respectivos componentes. Se utiliza tanto Array y Json. Los Arrays se utilizan principalmente dentro de angular, se utilizan estas estructuras ya que nos permiten iterarlas y de esa manera obtener los valores que se necesitan mostrar al usuario mediante la pagina web. Por otro lado, los Jsons se utilizan para el servicio de REST/API, lo anterior debido a que la forma en que la base de datos recibe una solicitud es por medio de esta estructura de datos.

* + Problemas conocidos: Corresponde a cualquier problema que no se ha podido solucionar en el trabajo.

Los problemas que no se han podido solucionar se deben principalmente a la falta de tiempo para la tarea corta, cabe mencionar que la implementación de la base de datos consumió mucho tiempo, debido a que la curva de aprendizaje de las herramientas de EntityFrameWork, Bootstrap, HTML y CSS fue significativamente alta.

En la sección de productores, hizo falta un comando PUT para editar la cantidad de producto que tienen los productores, en el momento de realizar una compra este atributo debería de disminuirse.

* + Problemas encontrados: Descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

Con el fin de dar una mejor explicación de los problemas encontrados, se dividen estos por herramienta.

* Bootstrap: las dependencias de esta herramienta deben de ser incluidas, en un documento específico (index.html). Esto con el fin de utilizar todas las funciones.
* HTML: se pretende mostrar elementos de una lista que puede variar en tamaño, por esta razón se utiliza \*ngFor, para iterar una lista (Array)
* CSS: de momento no carga
* EntityFramework: gracias a esta herramienta se puede hacer la conexión con la base de datos. Su curva de aprendizaje es bastante alta, pero las funcionalidades que ofrece son de gran ayuda para realizar los métodos de protocolo https (Get, Post, Put, Delete).
* SQL: El principal problema fue crear una estructura de base de datos que fuera 100% funcional. Se tenía el problema que un cliente solamente podía comprar un único producto por compra, esto se solucionó agregando una nueva tabla con una llave foránea que relaciona la llave primaria de la tabla de usuarios.
  + Documentación que evidencie el trabajo en equipo:
    - Actividades planeadas y su responsable (Plan de trabajo).
    - Minutas de sesiones de trabajo (Seguimiento al plan de trabajo).
    - Actividades realizadas por cada estudiante. Bitácora en digital, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, entre otros. Se debe describir todo por más insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo. Imaginen que si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.
  + Conclusiones y recomendaciones del proyecto.
  + Bibliografía consultada para el proyecto.

<https://www.entityframeworktutorial.net/efcore/querying-in-ef-core.aspx>

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Nombre función | Parámetros | Explicación |
| Login-general | authorization |  | Verifica si el nombre de usuario y contraseña coinciden con los datos de la base de datos |
| Tramo-Producto | Producers | i | Dirige al cliente a una dirección específica de cada productor |
| Tramo-Producto | editDelete |  | Dirige al cliente a la página donde puede editar sus datos o eliminar la cuenta |
| Producers | refresh | I | Calcula el total de la orden parcial que lleva el Cliente |
| Producers | delete | j | Elimina los productos de la lista que lleva la orden |
| Producers | addOrderP | k, customer | Verifica si existe una orden activa, en el caso que no exista, se crea una |
| Checkout | calcTotal |  | Calcula el monto total de la orden, suma los valores de las ordenes de productos |
| Checkout | add | I | Suma un producto a la compra |
| Checkout | less | i | Resta un producto a la compra |
| Sign-up | addUserC |  | Agrega un nuevo usuario a la base de datos por medio del API, hace una verificación para ver si se trata de un cliente o un productor |
| Sign-up | editUser |  | Permite editar el usuario con los nuevos datos que se le brindan |
| Sign-up | deleteUser |  | Elimina un usuario de la base de datos. |
|  |  |  |  |
|  |  |  |  |

La mayoría de componentes utilizan las funciones llamadas getUser(), getProducers(), getOrders(), entre otras, para facilitar la comunicación con la base de datos por medio del API

Para realizar la conexión con la base de datos se utilizan varios controladores los cuales poseen los mismos métodos para los protocolos https, dichos métodos se describen a continuación

|  |  |  |
| --- | --- | --- |
| Nombre de la Función | Parametro | Explicación |
| IActionResult Get |  | Retorna los datos de una tabla en específico |
| IActionResult Post | Request | Se hace un pedido de para que se inserte nuevos elementos a una tabla de la base de datos |
| IActionResult Put | Request | Pedido para editar elementos de una tabla de la base de datos |
| IActionResult Delete | id | Elimina de manera horizontal algún elemento de una tabla de la base de datos, es elemento es reconocido por medio de su id |