TEC | Tecnológico de Costa Rica

El Mercadito

Integrantes:

Pablo Azofeifa González
María Lucía Monge Golcher
Maesly Velásquez Bone
Viviana Villalobos Valverde

Profesor:
Luis Diego Noguera Mena

Fecha:
21 de Octubre de 2020

Índice

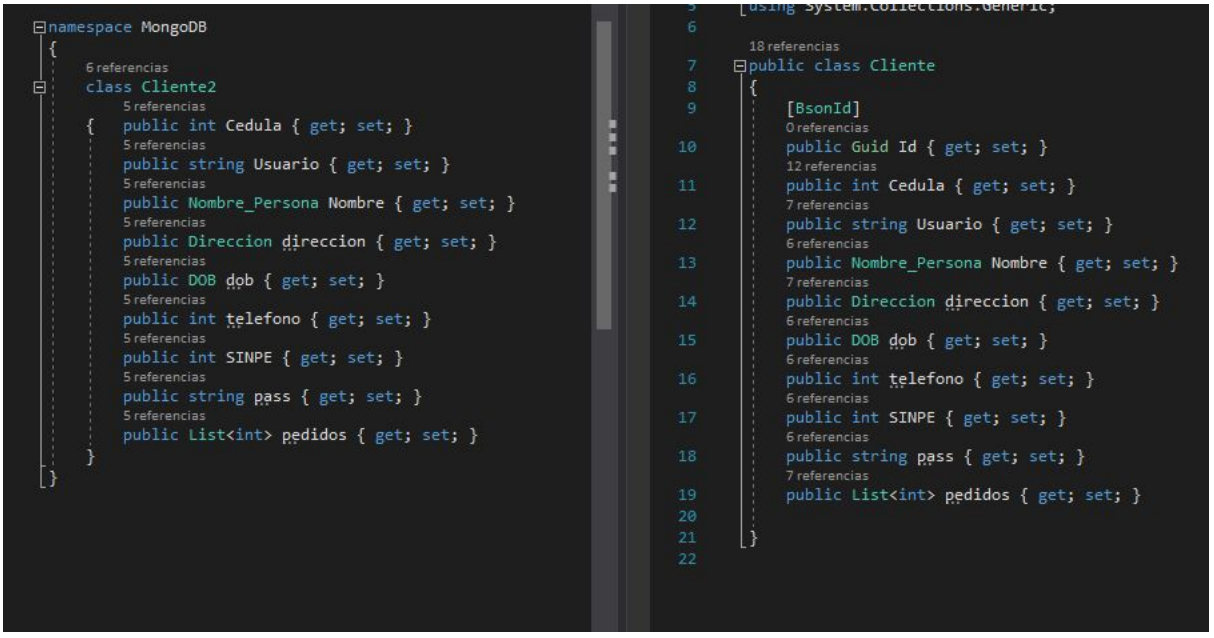
Descripción de los métodos implementados

Del lado del servidor en C# se crearon dos métodos importantes los cuales son el manejo del GET y del POST según corresponda.

Cuando llega un GET se tienen 11 solicitudes generales de donde se derivan el resto. Los gets siguen una estructura básica donde /instrucción/filtro/, donde la instrucción es lo que se desea del servidor y el filtro es lo que filtra y obtiene según lo deseado.

Por ejemplo cuando se pide la información de un cliente en específico según su cédula, la solicitud es /CedulaCliente/117480511.

Este get devolverá el cliente con el numero de cedula 117480511 y si no hay lo devuelve vacío. Así mismo las clases cliente, productor, producto y pedido se duplicaron sin el valor de Mongodb como id dado que de esta forma no se logra recibir en el cliente, por lo que se se envía duplicado sin el valor de id que solo se utiliza en Mongo. Por ejemplo:



Del lado izquierdo es el objeto que le llega al cliente y del lado derecho es lo que se guarda en el servidor de Mongodb y de la misma forma se trabaja con las clases antes mencionadas.

Por el lado de los POST funciona de la misma manera llega una solicitud junto con un body de una clase que corresponde a la duplicada por lo que se toman esos valores y se guardan en uno original.

Descripción de las estructuras de datos desarrolladas

Para el presente trabajo se optó como estructura de datos la utilización de los archivos JSON, ya que entre los beneficios que ofrece encontramos de entrada que es un formato ligero de intercambio de datos; además que lo que corresponde a leerlo y escribirlo es simple para los usuarios y para las máquinas es simple interpretarlo y generarlo. Otra ventaja, es que su estructura se está constituido por 2 partes: la primera una colección de pares nombre/valor y una lista ordenada de valores, lo cual ayudaba a que la utilización de esta estructura se acoplara de gran forma para los datos que se iban a necesitar.

Descripción detallada de los algoritmos desarrollados

Para el desarrollo del presente trabajo, se utilizaron lo que fueron reglas sencillas para lograr funcionamiento oportuno. Dichas reglas las mencionaremos en un tono general, ya que su lógica es la misma para muchas funciones de la página como por ejemplo la acción de los botones de la página.

En la parte visual básicamente contamos las siguientes reglas:

1) Botones: al dar click en un botón, este debe desplegarse o redirigir hacia otra vista de la aplicación o bien aplicar un cambio en la página donde se encuentra.

2) Campos de texto: al dar click en alguno de estos campos, se permite escribir tanto letras como números según se ocupe sin restricción alguna.

3) Desplegar información: al haber cambios (ya sea que se editó, eliminó o se agrego) en los archivos donde se guarda la información (archivos JSON), el programa deberá mostrar los respectivos cambios.

Además, se cuentan con otra serie de algoritmos en el frontend, que nos ayudan en la comunicación que respecta a la base de datos, estas se dividen como en 2 partes: los services forman la base de la comunicación con el API y en los typescript de los componentes o vistas, tenemos otras que se encargan de la implementación de estos servicios para usar la información que proveen.

En el lado de los services, para darse una idea tenemos por ejemplo:

1) Para el cliente: getClients() para obtener todos los clientes, addCliente(client:Client2) para agregar un nuevo cliente, authCliente(username:string, password:string) para la verificación validadción del cliente, getClienteCedula(cedula:string) para obtener un

cliente en específico por su cedula, addPedido(pedido:Pedido2) para conectar un pedido hecho a un cliente.

Estas funciones son las principales y su estructura es la misma para lo que corresponde a productores y productos.

En los typescript lo que se maneja es como se manipula la información que se obtiene de los JSON, ya sea para mostrarla en las vistas o para enviar, para esto último se obtiene la información de los “placeholder” o campos para escribir de las vistas y se acomodan según la estructura que ya viene dada desde la base de datos para procesar los datos.

Por parte del API, como su principal función es la comunicación con la base de datos (archivo JSON) este tiene básicamente 2 funciones que es la de enviar y captar información (get y post), por lo cual se implementaron distintas funciones.

Para los clientes contamos con:

1) public IActionResult Get(): al hacer una solicitud del tipo get, esta función se encarga de recorrer la base de datos y coloca todos los clientes junto sus datos para agregarlos a un JSON que luego serán enviado.
2) public IActionResult AgregarCliente(Cliente2 nuevoCliente): esta es la que se encarga de acomodar los nuevos datos en un JSON para luego mandarlos a la base de datos.
3) public IActionResult AuthClient(string username, string password): este es para hacer un get específico de datos, ya que solicita el usuario y contraseña de un cliente en específico.
4) public IActionResult GetCCedula(string cedula): esta función está diseñada para obtener solamente un cliente de la base de datos.

Para los productores tenemos:

1) public IActionResult Get(): esta función se encarga de recorrer la base de datos, toma cada productor con sus datos y los almacena en un

JSON que luego será enviado.

2) public IActionResult AgregarProductor(Productor2 nuevoProductor): con esta función lo que se logra es poder ingresar un nuevo productor en la base de datos, ya que capta los nuevos datos en un JSON que luego se envía.
3) public IActionResult AuthProductor(string username, string password): este es para hacer un get específico en la base de datos, ya que solicita el usuario y contraseña de un productor en específico.

Para la dirección tenemos:

1) public IActionResult GetPDireccion(string provincia, string canton, string distrito): este obtiene los valores de la provincia, cantón y distrito que se encuentran ya precargados para evitar datos falsos o incorrectos.

Para los productos tenemos:

1) public IActionResult GetProductos(string usuario): este sería para obtener los productos ligados a un usuario en específico.
2) public IActionResult AgregarProducto(Producto2 nuevoProducto): este sería para agregar nuevos productos a la base de datos.

Para la base de datos tenemos:

1) public void InsertRecord<T>(string table, T record): este se encarga de insertar un valor en la tabla que se solicite.
2) public List<T> LoadRecords<T>(string table): esta se encarga de obtener los valores de una base de datos.
3) public T LoadRecordbyId<T>(string table, Guid id): obtiene el valor de un ID en específico y los devuelve.
4) public void DeleteRecord<T>(string table, Guid id): se encarga de eliminar un dato en específico.
5) public Client ObtenerCliente(int id): para obtener un cliente en específico.
6) public void Agregar2(Cliente2 nuevoClient): para poder agregar un

nuevo cliente.

7) public List<Cliente2> getAllClientes(): se encarga de obtener todos los clientes de la tabla.
8) public List<Productor2> getAllProductores(): se encarga de obtener todos los productores de la tabla.
9) public List<Producto2> getAllProductos(): se encarga de obtener todos los productos de la tabla.
10) public User auth(string username, string password): se encarga de hacer como la validación o verificación del usuario y la contraseña.

Problemas conocidos

1) Algunos botones no se les pudo encontrar la falla exacta del problema y no ofrecen funcionalidad.
2) No se pudo implementar la parte de guardar las imágenes de los productos.
3) No se logró la implementación de los reportes, ya que no sirvió ni Crystal Report ni Reporting Services.
4) No se logró la implementación de una aplicación nativa para dispositivos móviles, se buscaron opciones como Ionic y PWA pero no se logró el cometido.
5) Se quería implementar en los pedidos un botón que mostrará los pedidos entregados, provocando que el pedido no se pudiera cambiar y poniendo un fondo gris, sin embargo no se logró su desarrollo.
6) Por parte del administrador se deseaba que este aceptara las solicitudes o las negara, esto no fue implementado a tiempo.
7) Los botones de las vistas referentes al productor carecen de funcionalidad.

Problemas encontrados

Bootstrap

Descripción: con respecto a la estética no se veían que se aplicaran los cambios en las vistas de la aplicación web.

Intentos sin solución: se cambió y se probó con distintos datos en el CSS sin mayor cambio alguno.

Solución Encontrada: luego de revisar bien el código, se encontró que el problema no era el css que se estaba utilizando, sino que, por errores humanos, no se había instalado bootstrap de manera correcta con lo cual se procedió a instalarlo.

Recomendaciones: verificar que todas aquellas librerías que se necesitan estén instaladas correctamente.

Conclusiones: antes de empezar a buscar soluciones, es mejor verificar que se cuenta con todo lo que se necesita para la correcta tramitación del programa y no entrar en pánico.

Bibliografía: Bootstrap team. (2018). Bootstrap. octubre 13, 2019, de Bootstrap team Sitio web: https://getbootstrap.com/

Carrito de compras

Descripción: Se buscaba tener botones para agregar o eliminar productos del carrito de compras, así como agregar mayor cantidad de los mismos o en caso de error disminuir la cantidad solicitada.

Intentos sin solución: se intentó modificar la lista del carrito de compras con ayuda del producto seleccionado, sin embargo esto generaba trabas para devolverse en la lista del carrito, los productos debían ser adyacentes con respecto al índice ya que no funcionaba si no era el caso, así mismo al llegar a usar el último producto la lista quedaba inhabilitadas las opciones anteriores.

Solución encontrada: se colocó un atributo extra en la lista del carrito de compras, conocido como id, el cual tiene el valor del índice del producto dependiendo de su

posición en la lista, de forma que no es necesario generar condiciones ni bucles. Al seleccionar el botón que hace referencia a un producto este cambia por índice lo necesario, es decir precio y cantidad.

Recomendaciones: buscar soluciones fuera de lo que está establecido, ya que la base o los atributos se pueden modificar según la necesidad enfrentada, así mismo tener presente la estructura que se utiliza para la información ya que puede facilitar su administración y abstracción.

Conclusiones: lo ideal a la hora de realizar es funciones es mantener el código simple y legible.

Rutas de Acceso

Descripción: a la hora de correr la aplicación había algunas vistas que no se lograban observar, ya que no existía la ruta hacia la misma.

Intentos sin Solución: se intentaba recargar la página y el proyecto en especial.

Solución encontrada: se revisó el proyecto y se encontró que algunas rutas habían sido escritas de manera incorrecta, por lo cual se procedió a cambiarlas dando como resultado que ya funcionan las rutas que se tenían.

Recomendaciones: revisar el archivo donde se encuentran las rutas para verificar que las mismas se encuentren correctamente escritas.

Conclusiones: ser cuidadoso para evitar malos manejos en los nombres de las rutas.

Bibliografía: Angular. (2018). Routing & Navigation. noviembre 3, 2019, de Angular Sitio web: https://angular.io/guide/router

Documentación de bitácora

Bitácora 1

miércoles, 23 de septiembre de 2020 20:03

Integrantes presentes

- Viviana Villalobos Valverde
- Ma. Lucía Monge Golcher
- Pablo Azofeifa González
- Maesly Velásquez Bone

Requerimientos

Tareas

1. Aplicación Web
- Vista Administración
1. Gestión de productos
- Num de cedula
- Nombre
- Apellidos
- Dirección (Provincia, Cantón, Distrito) todos estos por defecto
- Fecha de nacimiento
- Teléfono
- Número de SINPE móvil
- Lugares de entrega
- Crear, actualizar y eliminar el productor
2. Administración de afiliciaciones
- Aceptar o denegar al productor
- Comentario de porque no se puede
3. Gestión de categoría
- Creción, actualización y eleminación de categorías
- Identificador único
- Nombre (verduras, lácteos, legumbres, etc)
4. Vista de reportes
- 10, más vendidos productos
- 10, productos más ganancias
- 10, productos más vendidos por productor
- 10, clientes con más compras

Productor y cliente
Productor o cliente
Varios productores en la misma orden o mezclado

- Vista Público General
1. Gestión de clientes
- Creación, actualización, eliminación de clientes
- Número de cédula
- Nombre
- Apellidos
- Dirección (Provincia, Cantón, distrito)
- Fecha de nacimiento
- Teléfono
- Usuario
- Password
2. Log In
- Productores que entregan en el distrito
3. Entrar al tramo del productor
- Productos del productos
- Cantidad disponible
- Precio
4. Administrador del carrito de compras
- Eliminar productos
- Modificar cantidades
- Línea por producto
- Total de monto a pagar al productor
5. Realizar Compra
- Comprobante de pago

- Vista del productor

1. Solicitud de Afiliación
- Número de cédula
- Nombre
- Apellidos
- Lugar de Residencia (Provincia, Cantón, Distrito) por defecto
- Fecha de nacimiento
- Teléfono
- Número de SINPE móvil
2. Gestión de productos
- Creación, actualización y eliminación de productos
- Nombre
- Categoría
- Foto
- Precio
- Modo de venta (kilo, paquete, caja, entre otros)
- Disponibilidad
3. Gestión de pedidos
- Compras anteriores
- Listado la compra
- Comprobante de pago
- Dirección de entrega

5. Realizar Compra
- Comprobante de pago
6. Feedback
- Cuando haya recibido la compra
- Califica el producto
- Califica el productor

1. Aplicación Móvil

Se implementa la Vista Web Público General

Bitácora 2

domingo, 4 de octubre de 2020 22:17

Integrantes presentes

- Ma. Lucía Monge Golcher
- Pablo Azofeifa González

Inicio del desarrollo en Frontend

Teniendo el diseño de las vistas en una página de figma, el cual se puede encontrar en el siguiente enlace:
https://www.figma.com/file/XcPQjbK1xnkssfIPYOQV5M/Bases?node-id=0%3A1
Se inicia la programación de las vistas de la página web, para esto se tuvo una reunión con las personas encargadas del desarrollo, se pueden ver en integrantes presentes.

Division de trabajo

Se cuentan con 18 vistas en página web y 8 en app movil. Por lo que se dividen 9 vistas cada desarrollador web y 4 vistas cada desarrollador para aplicación en android

Temas desarrollados

- Se estipula el uso de git para manejo de versiones de las paginas.
- Se ve la necesidad de contar con el figma.
- El uso de flexbox en el proyecto.
- Se cuenta con las herramientas necesarias para el desarrollo.

Bitácora 3

lunes, 5 de octubre de 2020 19:44

Integrantes presentes

- Viviana Villalobos Valverde
- Ma. Lucía Monge Golcher
- Pablo Azofeifa González
- Maesly Velásquez Bone

Temas

- Se revisa el avance de los equipos tanto Backend como Frontend.
- Dudas sobre casos particulares de la tarea.

Avance

Se ven cuatro vistas que se han avanzado desde que se asignó la división de trabajo.
Se tiene la estructura de la base de datos desarrollada en MongoDB.

Próxima reunión

El Viernes 9 a las 9:00 pm con el propósito de revisar el avance del proyecto, con las vistas realizadas y la base de datos completa.

Bitácora 4

Saturday, 10 October 2020 10:27 AM

Integrantes presentes

- Viviana Villalobos Valverde
- Ma. Lucía Monge Golcher
- Pablo Azofeifa González
- Maesly Velásquez Bone

Temas

Revisión de lo estipulado en la bitácora anterior

Avance

Se revisan las vistas finalizadas, así como la estructura realizada en Mongodb tanto con get y post de este.
Se inicia la documentación del proyecto en proceso.

Faltantes

Se piden algunos cambios para las vistas
Se necesita recibir y enviar la información al frontend
Realizar la investigación de la aplicación móvil

Próxima reunión

Los días Lunes y Martes se realizarán reuniones para dejar los entregables listos.

Bitácora 5

Tuesday, 13 October 2020 11:30 AM

Integrantes presentes

- Viviana Villalobos Valverde
- Ma. Lucía Monge Golcher
- Pablo Azofeifa González
- Maesly Velásquez Bone

Temas

Errores al conectar backend y frontend

Problemas

Se tuvieron errores de codigo, se solucionaron, sin embargo no se concretaron soluciones por errores técnicos en el equipo de una de las compañeras.
Al solucionar los problemas técnicos se dieron otros, por lo cual tuvimos que migrar el servidor y realizar pruebas con solicitudes.

Bitácora 6

Friday, 16 October 2020 12:08 AM

Integrantes presentes

- Viviana Villalobos Valverde
- Ma. Lucía Monge Golcher
- Pablo Azofeifa González
- Maesly Velásquez Bone

Temas

Los faltantes de los entregables

Faltantes

- Por parte de la parte de la documentación, el manual de usuario además de los problemas tanto conocidos como encontrados
- Correcciones en: el carrito de compras, ortografía, la página de home
- Conectar con la base de datos con el finde que los valores salgan en pantalla
- Completar las solicitudes del API
- Revisar el flujo de la aplicación
- Hacer el drop down menu
- Capacidad de subir imágenes
- Servicio para obtener reportes

Se dividieron las tareas faltantes

Próxima reunión

Se realizarán las reuniones necesarias para la implementación del entregable

Conclusiones y recomendaciones del proyecto

Las aplicaciones web llegaron para quedarse, de ahí la importancia por buscar mejoras y una expansión hacia aquellas áreas que todavía no cuenten o brinden un servicio de este tipo, ya que siempre se busca agilizar o facilitar las tareas del diario vivir.

Estos servicios, en su versión final o funcional, deben ser fáciles de aprender y/o entender para el usuario final, ya que lo que se busca es facilidad y rapidez por parte de las empresas que lo adquieren para no ver pérdidas en la parte económica y lo correspondiente a la información, esto con el fin de que no exista un sobrecosto de la aplicación y solamente se deba pensar en su mantenimiento o en alguna actualización/mejora en el futuro pero que no implique una realización o reestructuración desde cero.

Hay cantidades inmensas de información sobre aplicaciones web, con las cuales se puede tener una muy buena base para su implementación y creación. Se puede encontrar desde detalles tan pequeños como la estética de una botón hasta la conexión con una base de datos (en el apartado del servidor y el web API), todo esto siendo de ayuda para tener una fluidez durante el tiempo de trabajo en el cual se implemente la creación de la aplicación, aunque siendo conscientes y sin dejar de lado que pueden surgir errores o dudas en el camino que encontrar su solución se complique o bien que no se le pueda dar una solución completa.

Bibliografía

1) Linux Foundation. (). *About Node.js*. Linux Foundation. Node.js Foundation Recuperado de https://nodejs.org/en/about/
2) Lahoud, P.. (5 de diciembre de 2017). *Getting Started with Node.js, Angular, and Visual Studio Code*. Premier Developer. Microsoft Recuperado de https://devblogs.microsoft.com/premier-developer/getting-started-with-node-js-angular-and-visual-studio-code/
3) (). *One framework. Mobile & desktop.*. . Google Recuperado de https://angular.io/
4) W3Schools. (). *HTML The language for building web pages*. . W3Schools Recuperado de https://www.w3schools.com/
5) Bootstrap team. (). *Bootstrap*. . Bootstrap team Recuperado de https://getbootstrap.com/
6) CSS-Tricks. (28 de septiembre de 2020). *A Complete Guide to Flexbox*. Recuperado de https://css-tricks.com/snippets/css/a-guide-to-flexbox/
7) *Introduction to the Angular Docs*. . Angular Recuperado de https://angular.io/docs
8) Daniel Donbavand. (2018, 25 junio). ASP.NET Core 2.1: Building a Simple Web API [Vídeo]. YouTube. https://www.youtube.com/watch?v=J_MEscBWJYI&feature=youtu.be&ab_channel=DanielDonbavand
9) Intro to MongoDB with C# - Learn what NoSQL is, why it is different than SQL and how to use it in C#. (2019, 6 mayo). [Vídeo]. YouTube. https://www.youtube.com/watch?v=69WBy4MHYUw&t=183s&ab_channel=IAmTimCorey
10) Cómo arreglar el error CORS Angular. (2020, 13 abril). [Vídeo]. YouTube. https://www.youtube.com/watch?v=iXbsn-qaqsw&feature=youtu.be&ab_channel=DominiCode
11) Se puede acceder al repositorio del trabajo presentado en el siguiente enlace, https://github.com/ViviVillalobos79/Tarea-1---Bases