Prueba Yamaha

Punto 1:

En la carpeta llamada “Punto 1” del repositorio se encuentra el código para crear las tablas.

La base de datos resultante sería esta (la imagen también se puede observar en el repositorio):

Figura 1. Base de Datos.

Hay más relaciones que se pueden hacer, por ejemplo tiendas con vendedores, ya que una tienda puede tener varios vendedores, pero en el contexto de la prueba se hace este análisis básico.

PUNTO 2:

Frontend:

Se usó código solo para crear un formulario simple y una tabla básica para las ventas. El código está en el repositorio en la carpeta llamada “Punto 2”. El estilo del frontend se puede mejorar usando librerías y frameworks como Material y Bootstrap.

Figura 2. Formulario Ventas.

Lista de Ventas

NumeroFactura	Ciudad	Tienda	Precio	NumeroMotor	Modelo	Vendedor	Cliente	
F-0001	Cali	Avenida 80 #15-67	1500000	NM123456	Yamaha MT-03	1034567890	1009876543	Editar
F-0002	Bogotá	Carrera 12 #8-45	2300000	NM654321	Yamaha R3	1035678901	1010987654	Editar
F-0003	Medellín	Calle 50 #20-30	1800000	NM987654	Yamaha XSR700	1036789012	1012098765	Editar
F-0004	Barranquilla	Diagonal 33 #14-60	2700000	NM567890	Yamaha FZ-25	1037890123	1013209876	Editar
F-0005	Bucaramanga	Transversal 22 #10-98	2000000	NM112233	Yamaha Tracer 900	1038901234	1014320987	Editar
F-0006	Cartagena	Calle 18 #5-77	2200000	NM445566	Yamaha Tenere 700	1039012345	1015432098	Editar
F-0007	Pereira	Avenida 45 #9-20	2800000	NM778899	Yamaha YZF-R6	1040123456	1016543209	Editar
F-0008	Manizales	Carrera 9 #23-11	1600000	NM334455	Yamaha FZ-S	1041234567	1017654320	Editar
F-0009	Santa Marta	Calle 7 #45-33	1900000	NM556677	Yamaha WR450F	1042345678	1018765431	Editar

Figura 3. Tabla ventas.

El resto de formularios se podrían implementar de manera similar. Para ahorrar tiempo se diseñó el formulario de clientes en Draw.io (ver figura 4).

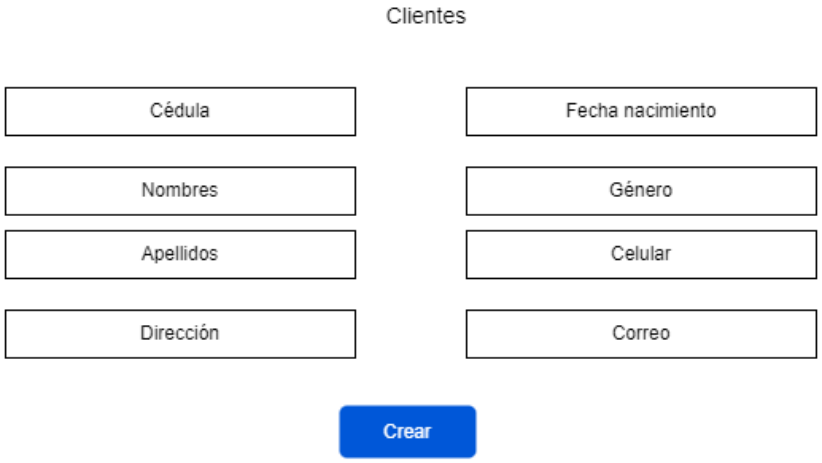


Figura 4. Formulario de clientes.

Y la lógica es la misma que en el código de ventas: usar axios o fetch para implementar los métodos get, post, put, etc.

Backend:

Se implementa el backend con .NET. El código se puede encontrar en la carpeta llamada “Backend”.

Nota: el código no es funcional.

Se utiliza la estructura de la figura 5:

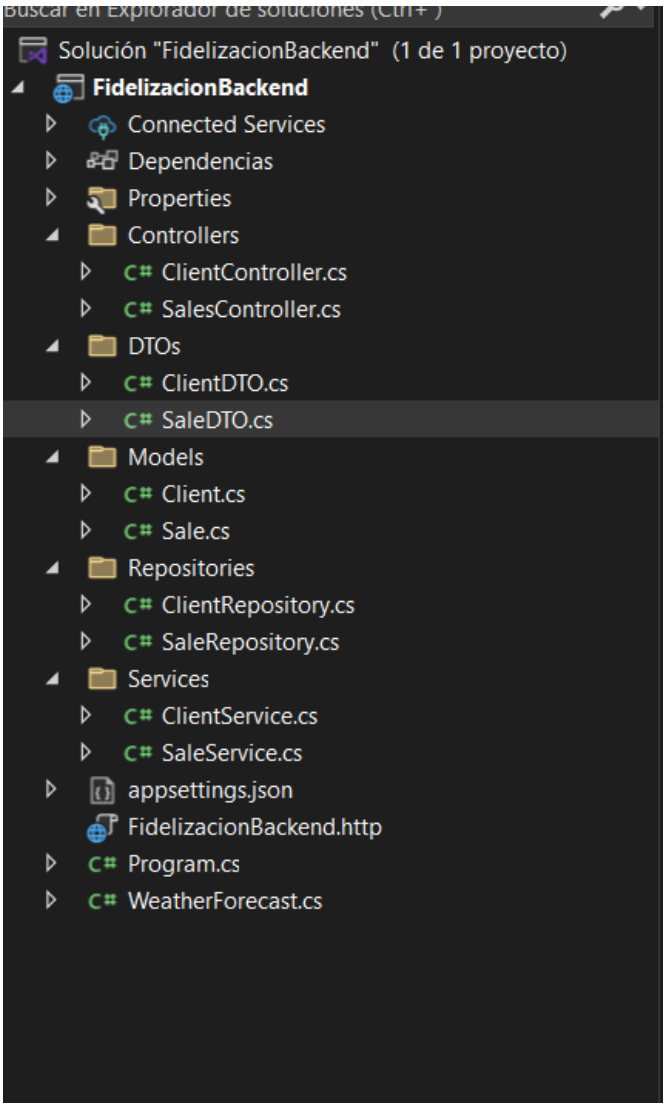


Figura 5: estructura backend

Ahora bien. El aplicativo es fundamental para analitica de datos (segmentación de clientes, comportamiento de clientes, etc). Para obtener la información de ventas de los últimos 10 años se debe tener en cuenta la integración que se realice con otras bases de datos y software que use la compañía (ERP, software propio, servicio de terceros, etc), donde esté la información necesaria que se guardará en la base de datos centralizada. Primero se tendría que realizar un proceso de limpieza de datos de lo que se obtenga de las demás fuentes de datos para así luego agregar la información en la aplicación.

Una vez organizada la información, la consulta sql sería similar a la que se encuentra en el “archivo procedimientos ventas” de la carpeta “procedimientos almacenados backend”, solo agregando los filtros respectivos (en este caso filtro de fecha). Y simplemente sería agregar **WHERE v.fecha >= DATEADD(YEAR, -10,**

GETDATE()). Lo ideal sería agregar paginación a la consulta para optimizar la aplicación.

PUNTO 3:

El frontend se diseñó de esta manera:

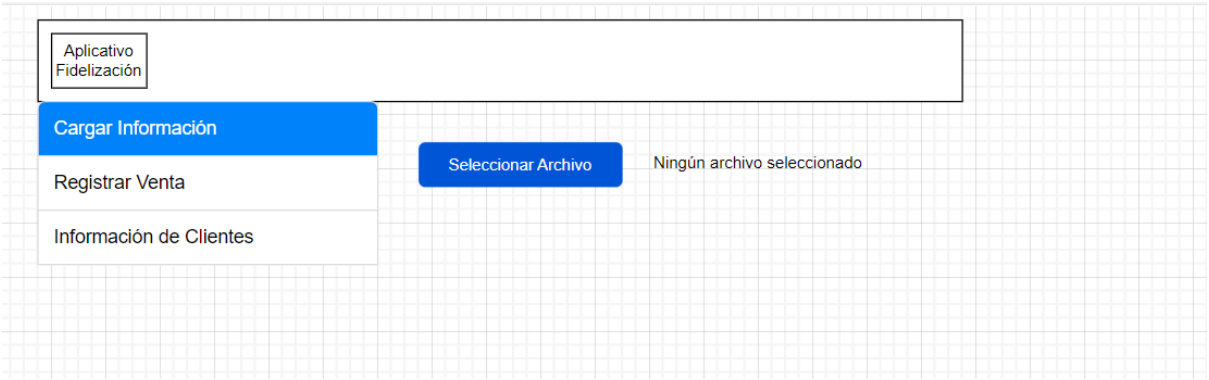


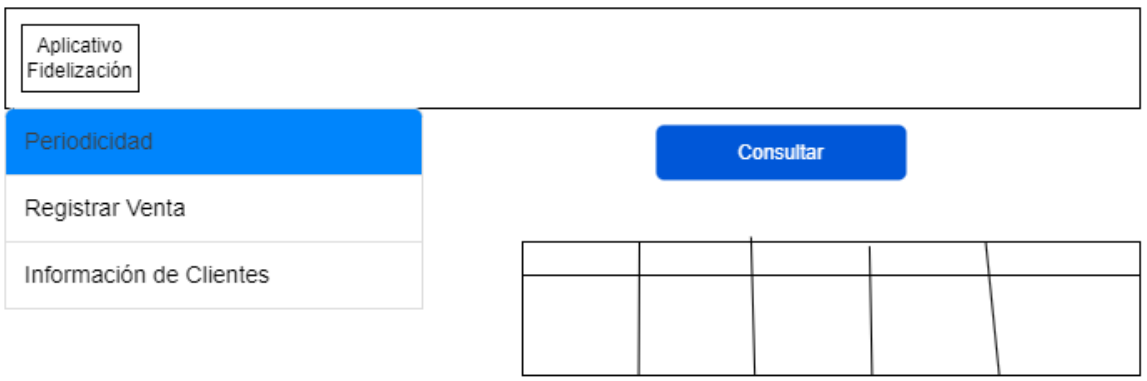
Figura 6. Frontend: selección de archivo

Se usaría un input de tipo “file” <input type=”file”/> y se enviaría el archivo con método post.

El código del backend está en el archivo “Método carga archivo”, en la carpeta punto 3. Es básicamente un método de controlador .NET.

PUNTO 4:

Este sería el diseño del frontend:



El código backend está en los archivos de la carpeta llamada “Punto 4”. Una archivo es para el código C# y el otro para el procedimiento almacenado.

PUNTO 5:

El backend se aloja usando EC2. Sin embargo, otra alternativa llamativa es usar Elastic Beanstalk, el cúal facilita bastante el despliegue de aplicaciones desarrolladas con .NET. Elastic Beanstalk ya crea por sí una instancia EC2 (esto también es configurable). Así mismo, dependiendo del tráfico que vaya a tener el sitio se puede considerar implementar autoscaling y loadbalancer. El número de instancias que se necesiten se saca de un análisis realizado con JMeter, configurando los recursos de las máquinas a usar. También se puede usar RDS para alojar la base de datos, ya que AWS cuenta con motor para SQL Server. Se hace la conexión RDS con EC2 (esto es posible incluso al momento de crear la base de datos en AWS). Para alojar archivos estáticos se usa S3; el frontend, por tanto, lo alojamos haciendo uso de CloudFront con S3. Por último, usamos Route 53 para manejar el dominio (si es necesario) y AWS Certificate Manager para el certificado HTTPS.

PUNTO 6:

- Buena implementación de JWT
- Prevenir inyecciones SQL: usando un ORM con entity framework o dapper en .NET. Si no se usa un ORM, dar buen manejo al paso de parámetros.
- Usar secrets en AWS para no exponer información sensible.
- Limitar intentos al iniciar sesión.
- Asegurar la implementación de HTTPS.
- Buscar un servicio que proteja de ataques DDos.

Prueba de conocimiento

1. **¿Cuál es la diferencia entre sitio web, servidor web y motor de búsqueda?**

Sitio web: páginas asociadas con un dominio en específico.

Servidor: Máquina que aloja los sitios web. Por lo general tienen software para manejar peticiones HTTP, manejar concurrencia y exponer los recursos (usando protocolo HTTP).

Motor de búsqueda: Sirve para buscar en la web.

2. **¿Cuáles son los lenguajes y/o componentes que se requieren para crear un sitio WEB?**

   - Lenguajes que puedan ser ejecutados en el navegador como JavaScript o WebAssembly.
   - Lenguajes backend (si el sitio no es solo estatico) como C#, Java, JavaScript (NojeJS), etc.
   - Servidor donde alojar el sitio.
   - HTML para la estructura y CSS para los estilos.
   - Una base de datos (de nuevo, si no es un sitio estático simple).
   - Un dominio.

3. **Describa la estructura básica de un servidor WEB.**

Es una máquina con software especial para alojar sitios web y exponer los recursos. Como cualquier computador, tiene sistema operativo, sistema de ficheros, manejo de redes, etc.

4. **Describa la estructura de directorios recomendada para una aplicación WEB.**

Depende del lenguaje y framework que se utilice. En frontend carpeta donde esté el código fuente principal (src), carpeta public para archivos estáticos accesibles en el navegador. En src se puede agregar carpetas de: services para lógica, rutas para el manejo de rutas, helpers para funciones que no se pueden clasificar en services, models y demás.

En backend tener services, controllers, models, DTOs, repositories, helpers, etc.

5. **Nombre 5 lenguajes que puedan ser usados para páginas web.**

PHP, JavaScript, C#, Python, Java.

6. **Elija un lenguaje web, y escriba 10 palabras reservadas**

C#: int, string, class, private, using, abstract, virtual, override, internal, return

7. **¿Cuál es el uso de HTML y CSS en programas web?**

HTML estructura del sitio. CSS estilos.

8. **¿Cuál es la diferencia entre bases de datos relacionales y no relacionales?**

Relacionales: usan tablas las cuales se relacionan con llaves. Dichas tablas tienen una estructura fija por lo general.

No relacionales: los datos se pueden guardar en otro tipo de estructuras como grafos, documentos, etc. No es necesaria la relación.

9. Describa los siguientes componentes y su posible uso: AWS Lambda, AWS Bucket S3, RDS.

Lambda: ejecutar código sin necesidad de un servidor, se cobra el tiempo de cómputo. Posible uso: una API.

S3: alojar archivos estáticos como imágenes, videos, archivos, etc. posible uso: alojar imágenes que se van a usar en el sitio.

RDS: alojar bases de datos, tiene motores de MySQL, SQL Server, etc.

10. Defina qué es OWASP y por qué es importante en el desarrollo de software.

Intenta hacer la web más segura. Es importante porque al momento de desarrollar se puede tener en cuenta las recomendaciones que dan para sitios web más seguros.