Memoria de Trabajo - API REST, Servidor y Cliente

Javier Moleón
DAM 2ºB
10/03/2025
PSEP

1. Introducción

Esta memoria describe el desarrollo e implementación de una API REST, un servidor y un cliente, todos desarrollados en C# utilizando .NET 8.0. La API gestiona datos de Fórmula 1, proporcionando endpoints para consultar información sobre equipos, pilotos y carreras. El servidor maneja las conexiones y la lógica de negocio asociada, mientras que el cliente permite consultar los datos de la API mediante una interfaz de consola.

2. Estructura del Proyecto

2.1 API REST

El directorio de la API incluye los siguientes archivos clave:

- ***ApiRest.csproj**: Archivo de configuración del proyecto.*
- ***Program.cs**: Punto de entrada de la aplicación.*
- ***F1Controller.cs**: Controlador que maneja las solicitudes HTTP.*
- ***F1Data.cs**: Clase que gestiona la estructura de datos.*
- ***f1_data.json**: Archivo de datos que almacena información en formato JSON.*

2.2 Servidor

El directorio del servidor incluye los siguientes archivos clave:

- ***servidor.csproj**: Archivo de configuración del proyecto.*
- ***Program.cs**: Punto de entrada del servidor.*
- ***Binarios y configuraciones** en `/bin/Debug/net8.0/`.*

2.3 Cliente

El cliente es una aplicación de consola en C# que permite interactuar con la API. Sus archivos clave son:

- ***cliente.csproj**: Archivo de configuración del proyecto.*
- ***Program.cs**: Contiene la lógica principal para consultar la API y mostrar datos en consola.*

3. Funcionamiento

3.1 API REST

La API utiliza un controlador llamado `F1Controller`, que proporciona endpoints para obtener información de Fórmula 1 desde el archivo `f1_data.json`.

Endpoints principales

- `GET /api/f1/equipos`: *Devuelve la lista de equipos.*
- `GET /api/f1/pilotos`: *Devuelve la lista de pilotos.*
- `GET /api/f1/carreras`: *Devuelve la lista de carreras.*

3.2 Servidor

El servidor se encarga de manejar la lógica de negocio y gestionar las conexiones con los clientes y la API REST.

3.3 Cliente

El cliente es una aplicación de consola que permite consultar los datos de la API mediante un menú interactivo. Sus principales funcionalidades incluyen:

- *Listar pilotos, equipos y circuitos.*
- *Consultar récords históricos de F1.*
- *Buscar pilotos, equipos y circuitos específicos.*

El cliente utiliza `HttpClient` *para realizar peticiones a la API, y* `JsonSerializer` *para procesar las respuestas.*

3.4 Encriptación

La seguridad de la comunicación entre el cliente y el servidor se implementa mediante encriptación AES (Advanced Encryption Standard). Esta implementación asegura que todos los datos transmitidos entre el cliente y el servidor estén protegidos.

3.4.1 Implementación de la Encriptación

La encriptación se implementa en ambos lados (cliente y servidor) mediante la clase `EncryptionHelper`*, que proporciona dos métodos principales:*

- `Encrypt`*(string plainText): Encripta el texto plano usando AES*

- `Decrypt`*(string cipherText): Desencripta el texto cifrado*

La clase utiliza las siguientes claves de encriptación:

- `Key`*: "YourSecretKey123" (16 bytes)*

- `IV` *(Vector de Inicialización): "YourIVVector1234" (16 bytes)*

3.4.2 Flujo de Comunicación Encriptada

1. El cliente inicia una conexión TCP con el servidor en el puerto 11000

2. Cuando el cliente envía una solicitud:

- El endpoint se encripta usando `EncryptionHelper.Encrypt()`

- Los datos encriptados se envían al servidor

3. El servidor recibe la solicitud:

- Desencripta los datos usando `EncryptionHelper.Decrypt()`

- Procesa la solicitud y obtiene la respuesta de la API

- Encripta la respuesta antes de enviarla al cliente

4. El cliente recibe la respuesta:

- Desencripta los datos recibidos

- Procesa y muestra la información al usuario

3.4.3 Seguridad

La implementación de la encriptación proporciona:

- Confidencialidad: Los datos transmitidos no pueden ser leídos por terceros

- Integridad: Los datos no pueden ser modificados durante la transmisión

- Autenticación: La comunicación solo es posible entre el cliente y el servidor autorizados

3.4.4 Manejo de Errores

El sistema incluye manejo de errores para casos de:

- Fallos en la encriptación/desencriptación

- Problemas de conexión

- Datos corruptos o malformados

4. Configuración y Ejecución

4.1 API REST

Para ejecutar la API, se deben seguir estos pasos:

1. *Abrir el proyecto en Visual Studio o utilizar la CLI de .NET.*
2. *Restaurar dependencias con `dotnet restore`.*
3. *Compilar con `dotnet build`.*
4. *Ejecutar con `dotnet run`.*

4.2 Servidor

Para ejecutar el servidor, se deben seguir los mismos pasos anteriores aplicados al directorio del servidor.

4.3 Cliente

Para ejecutar el cliente:

1. *Asegurarse de que la API REST esté en ejecución.*
2. *Abrir el proyecto del cliente en Visual Studio o la terminal.*
3. *Restaurar dependencias con `dotnet restore`.*
4. *Compilar con `dotnet build`.*
5. *Ejecutar con `dotnet run`.*

5. Pruebas

Se recomienda utilizar Postman o cURL para probar los endpoints de la API y verificar su correcto funcionamiento. Además, se pueden realizar pruebas de conexión al servidor y cliente mediante herramientas de depuración en Visual Studio.

6. Conclusión

Este proyecto integra una API REST, un servidor y un cliente de consola, todos en C# con .NET 8.0. La API proporciona datos de F1, el servidor maneja la lógica de negocio y el cliente permite la consulta de la información de manera interactiva.
