Recomendaciones de Mejora

1) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Program.cs&version=GBmain&line=14&lineEnd=21&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

La política CORS permite peticiones desde cualquier origen, método y header. En producción esto expone la API a ataques CSRF desde sitios maliciosos.

2) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Program.cs&version=GBmain&line=23&lineEnd=23&lineStartColumn=1&lineEndColumn=15&lineStyle=plain&_a=contents

Se utiliza dotenv.net para cargar configuración, que es un patrón de Node.js. En .NET debería usarse User Secrets para desarrollo y Azure Key Vault para producción. El archivo .env puede terminar accidentalmente en el repositorio exponiendo credenciales.

3) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Program.cs&version=GBmain&line=13&lineEnd=13&lineStartColumn=1&lineEndColumn=40&lineStyle=plain&_a=contents

El HttpClient se registra sin timeout, políticas de retry ni circuit breaker. Si TMDB responde lento o se cae, la aplicación se quedará esperando indefinidamente consumiendo recursos del servidor.

4) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Program.cs&version=GBmain&line=1&lineEnd=43&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

No existe implementación de health checks. Azure no puede determinar si las instancias están sanas para el balanceo de carga y escalado automático.

5) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Services/FilmService.cs&version=GBmain&line=27&lineEnd=27&lineStartColumn=1&lineEndColumn=100&lineStyle=plain&_a=contents

La API Key se envía como query parameter en la URL. Queda expuesta en logs del servidor, proxies intermedios, Application Insights y cualquier herramienta de monitoreo.

6) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Services/FilmService.cs&version=GBmain&line=24&lineEnd=27&lineStartColumn=1&lineEndColumn=100&lineStyle=plain&_a=contents

La API Key se envía duplicada: en el header Authorization y también en el query string. Esto es redundante y duplica los vectores de exposición de la credencial.

7) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Services/FilmService.cs&version=GBmain&line=27&lineEnd=27&lineStartColumn=50&lineEndColumn=95&lineStyle=plain&_a=contents

La URL base de TMDB está hardcodeada en el código. Si se necesita cambiar el endpoint, apuntar a un mock en desarrollo o usar un proxy, hay que modificar código y redesplegar.

8) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Services/FilmService.cs&version=GBmain&line=29&lineEnd=35&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

Cuando la petición a TMDB falla, solo se retorna null sin ningún logging. No hay forma de saber qué código de error devolvió ni cuál fue el mensaje, lo que hace imposible diagnosticar problemas en producción.

9) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Controllers/FilmController.cs&version=GBmain&line=23&lineEnd=29&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

El mensaje de error es genérico y no aporta información útil. El frontend no puede mostrar detalles relevantes al usuario y el equipo de soporte no tiene pistas para identificar la causa del problema.

10) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Controllers/AuthController.cs&version=GBmain&line=17&lineEnd=17&lineStartColumn=1&lineEndColumn=50&lineStyle=plain&_a=contents

El endpoint de login utiliza el verbo HTTP GET. Las operaciones de autenticación deben usar POST para evitar que información sensible quede en logs, historial del navegador y sea vulnerable a ataques CSRF.

11) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Controllers/AuthController.cs&version=GBmain&line=18&lineEnd=20&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

El método de autenticación no recibe ningún parámetro de credenciales. Cualquier persona puede llamar al endpoint y obtener un token válido sin necesidad de identificarse.

12) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Services/AuthService.cs&version=GBmain&line=18&lineEnd=23&lineStartColumn=1&lineEndColumn=100&lineStyle=plain&_a=contents

Las variables de entorno se utilizan sin validar que existan. Si no están configuradas correctamente, la aplicación fallará con errores crípticos en vez de indicar claramente qué configuración falta.

13) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Services/AuthService.cs&version=GBmain&line=24&lineEnd=31&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

El manejo de errores en autenticación retorna null cuando falla. No hay forma de distinguir entre un token expirado, credenciales inválidas o que el servicio externo está caído.

14) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/appsettings.json&version=GBmain&line=7&lineEnd=7&lineStartColumn=18&lineEndColumn=21&lineStyle=plain&_a=contents

AllowedHosts está configurado como "*", lo que permite cualquier valor en el header Host. Esto facilita ataques de host header injection en escenarios de generación de URLs o redirecciones.

15) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/appsettings.Development.json&version=GBmain&line=1&lineEnd=8&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

La configuración de desarrollo es idéntica a la de producción. Debería tener niveles de logging más detallados (Debug o Trace) y URLs de servicios diferenciadas para facilitar el desarrollo y testing local.

16) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Interfaces/IFilmService.cs&version=GBmain&line=7&lineEnd=7&lineStartColumn=1&lineEndColumn=50&lineStyle=plain&_a=contents

Existe código comentado que no se eliminó. El historial de cambios debe gestionarse con Git, no dejando líneas muertas que generan confusión sobre cuál es la implementación vigente.

17) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Interfaces/IFilmService.cs&version=GBmain&line=8&lineEnd=8&lineStartColumn=1&lineEndColumn=50&lineStyle=plain&_a=contents

La interfaz retorna string en lugar de un tipo fuerte. Se pierde el tipado de C#, la validación en tiempo de compilación y Swagger no puede documentar correctamente la estructura de la respuesta.

18) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Model/MovieResponse.cs&version=GBmain&line=5&lineEnd=9&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

Las propiedades del modelo no tienen atributos de serialización JSON. TMDB devuelve campos en snake_case (total_results, total_pages) que no se mapearán correctamente a las propiedades en PascalCase.

19) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Model/MovieResponse.cs&version=GBmain&line=9&lineEnd=9&lineStartColumn=1&lineEndColumn=50&lineStyle=plain&_a=contents

La lista Results no está inicializada. Si la API devuelve null en ese campo o alguien instancia el objeto sin asignarla, se producirá un NullReferenceException al intentar iterar sobre ella.

20) https://dev.azure.com/francoiscalderon/_git/BancoGuayaquil?path=/bg-movies-back/Model/Movie.cs&version=GBmain&line=5&lineEnd=12&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents

El modelo Movie no tiene atributos JsonPropertyName y el campo Release no coincide con el nombre real de TMDB que es release_date. Los datos de poster_path, release_date y vote_average no se deserializarán correctamente.