# Bland

1. You will need to create the "BlandGroup" database with "blandgroup" password.

Change the connection strings for your database and Azure Storage for yours in appsettings.Development:

"ConnectionStrings": {

"DefaultConnection": "Server=tcp:localhost,1433;Initial Catalog=BlandGroup;User ID=sa;Password=blandgroup;MultipleActiveResultSets=True;Encrypt=false;TrustServerCertificate=false;",

"AzureStorage": "Your connection string"

}

There is another connection string for your database in Program inside BlandGroupCamera project. Change it for your connection sting:

**var contextOptions = new DbContextOptionsBuilder<ApplicationDbContext>().UseSqlServer("Server=tcp:localhost,1433;Initial Catalog=BlandGroup;User ID=sa;Password=blandgroup;MultipleActiveResultSets=True;Encrypt=false;TrustServerCertificate=false;)**

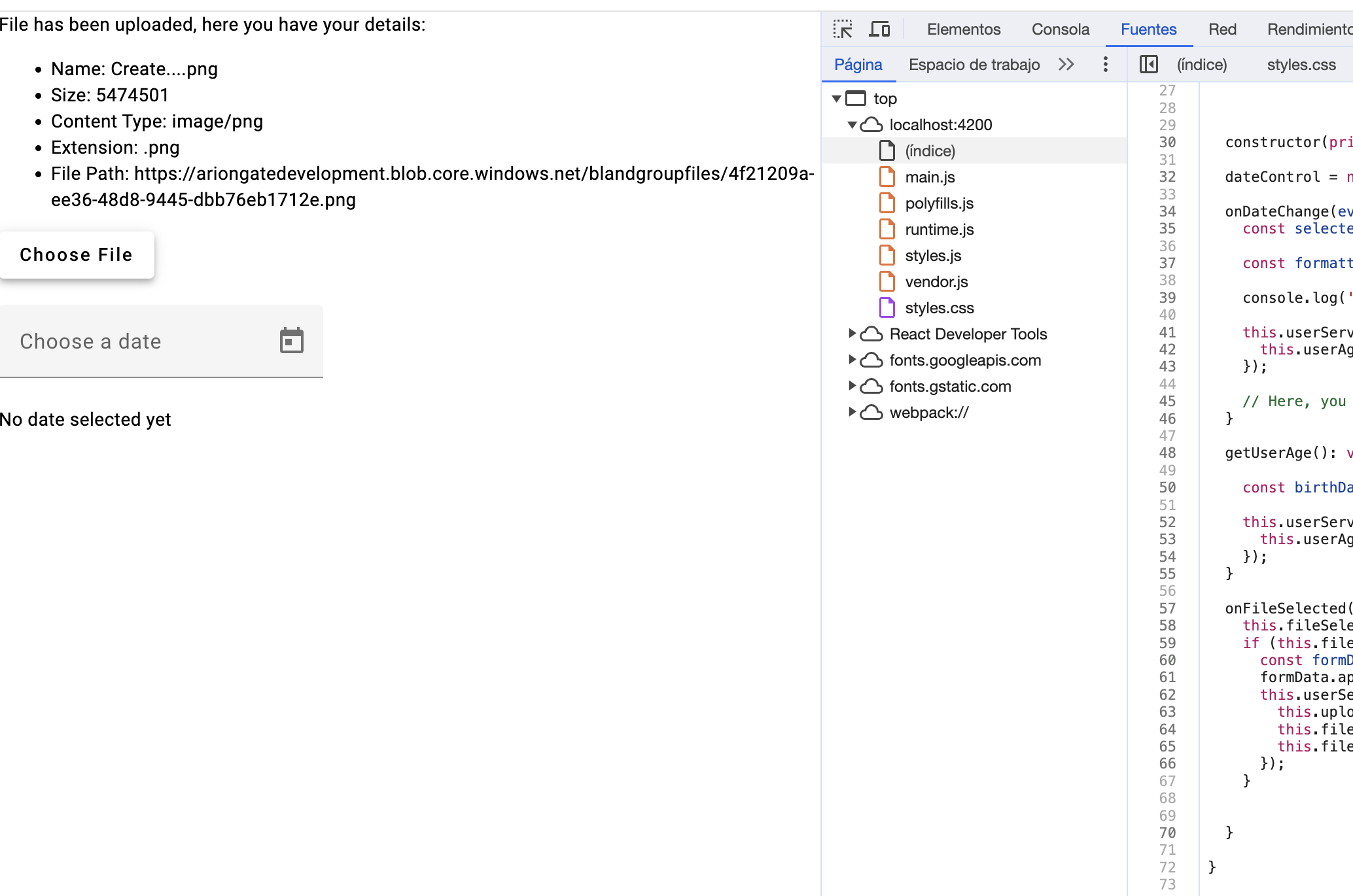
Remove the migrations and add run ef tools in a console as I indicate here:

- **dotnet ef migrations add Initial.**

Then to run the migrations type:

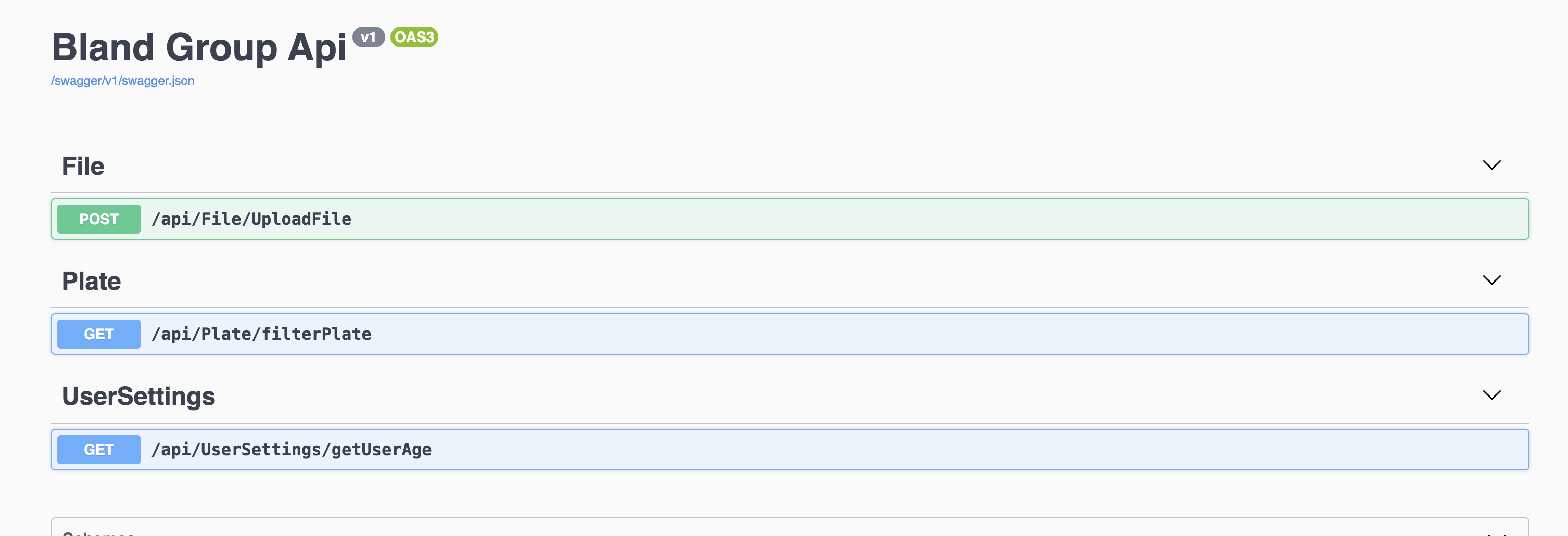
- **dotnet ef database update**.

2. To run the Angular Project **you must install angular and node**, go in the console to the folder **BlandGroupAgeAngular and run ng serve --open**. Then you must run the BlandGroupApi from visual studio as well.

Then inside the service of the Angular app, **you might want to change the port in which your Angular app is running**:

By default in mine is: **private apiUrl = 'https://localhost:7167/api';** From the Angular App you can upload a file for the exercise 4 and see the details and also test the age with the calendar for exercise 3.

- BlandGroup project contains a console app with the 2 first exercises. **Palindrome and Quicksort.**

- T**he BlandGroupApi provides Services for the Exercises 3, 4 and 5.** **From the controller, you can query the Plates, accept the files input and also has its endpoint for the Angular app for the age and the file upload.**

- BlandGroupCamera contains the console app that reads the camera files and stores the info in db.

I only wrote the feature for the real time drop off the files, but did not write the functionality to iterate through the existing ones. To test it **change the path to your local CameraPlates folder inside this project** and drop any ltr file inside:

**string relativeFilePath = "/Users/egoitz/Projects/Bland/BlandGroupCamera/CameraPlates"; // Replace with your relative file path**

**It will automatically detect the new file**. **Then you can query the camera with the dates it by running the Api with Swagger.**

I am a Mac user so I did not create a windows service, since I can only run it from a Windows Machine, unless I would install some windows virtual machine. Instead I created a console app that watches for the files. Other solutions in Mac would be to create a .net Background service, that simulates a windows service.

- Also there is not local Azure Storage for Mac, so I provided the configuration of the connection string to a real Azure Storage account.

- Unit tests project contains the unit tests for all the exercises.

- I could also implemented the Repo patter for EF, and return some genericApiResponses from the controllers but I did not want to over complicate the infrastructure.