Contents

1 Tools we Use

- Linting through Flutter
- Automated tests using GitHub actions
- GitGaurdian for secret scanning
- Codecov for code coverage
- Unit testing using Flutters built in tools
- Unit testing using XUnit with .Net

2 File Structure

BeakPeek is made using the MVC architecture as well as the domain driven design and for this reason on top of that we use flutter for our frontend and .Net for our backend we have separated our files into two directories those being ‘beakpeek‘ (due to how flutters project naming works) and ‘dotnet‘ to make it descriptive of the software bein used. For these reasons all future sections will be refered to under subsections for ‘Frontend‘ for all of the flutter related topics and ‘Backend‘ for all of the dotnet related topics.

2.1 Frontend

For the ‘beakpeek‘ directory the only directories that are of importence are the ‘lib‘ and the ‘test‘ directories.

2.1.1 Lib

The ‘lib‘ directory contains all of the actual logic and styling of the frontend this is also split up int appropriate subdirectories. These being ‘Controller‘, ‘Model‘ and ‘View‘ to match the MVC pattern. There are other directories as well that have uses for styling.

2.1.2 Test

The ‘test‘ direcotry is for the unit tests that are written for testing the flutter frontend and logic.

2.2 Backend

For the ‘dotnet‘ directory this is split up into two direcoties, one for the actual logic of the api this being called ‘BeakPeekApi‘ and then another for the unit tests of the api is this being called ‘BeakPeekApi.Tests‘.

2.2.1 BeakPeekApi

This directory also follows the MVC pattern however it does not have any views so there are only directories for model and controller.

2.2.2 BeakPeekApi.Tests

This direcotry mirrors the directory structure of BeakPeekApi exactly as for each file in BeakPeekApi there is also a unit test.

2.2.3 Entire Project

The rest of the direcoties should be self explanatory however here is a quick rundown

- ‘doc‘ is for all documentation for this BeakPeek.
- ‘res‘ is for all resources and contains all of the csv’s needed to run the api and load all of the birds
- ‘scripts‘ is for any scripts that are used for automation.

3 Formatting

3.1 Frontend

For formatting we use the built in tools and linting that flutter provides and is enforced by the flutter language server which we all use either through vscode or through neovim.

3.2 Backend

For formatting the backend we use the omnisharp C-sharp langauge server which has its own linting rules that are enforced on save either in vscode or in neovim.

4 Unit Testing

To ensure that all of our code is of a good standard and to make the development cycle faster we use unit tests written in either flutter or dotnet C-sharp. We aim for atleast a 70-80 percent coverage for the code.

5 Pull Request Restrictions

To ensure that the our code standards are maintained we have set up branch rules and pull request rules. The branch rules are that only the DevOps engineer

is able to push to the main branch and that any pull requests into the main branch require atleast one approval from someone else to push it.

To ensure that all pull requests adhere to our standards we have made it so that only the DevOps engineer should be allowed to approve pull requests into or from development and also for any pull requests into or from main. Any pull requests going into main have to first pass all workflows and tests.