Chirp! Project Report

ITU BDSA 2023 Group 11

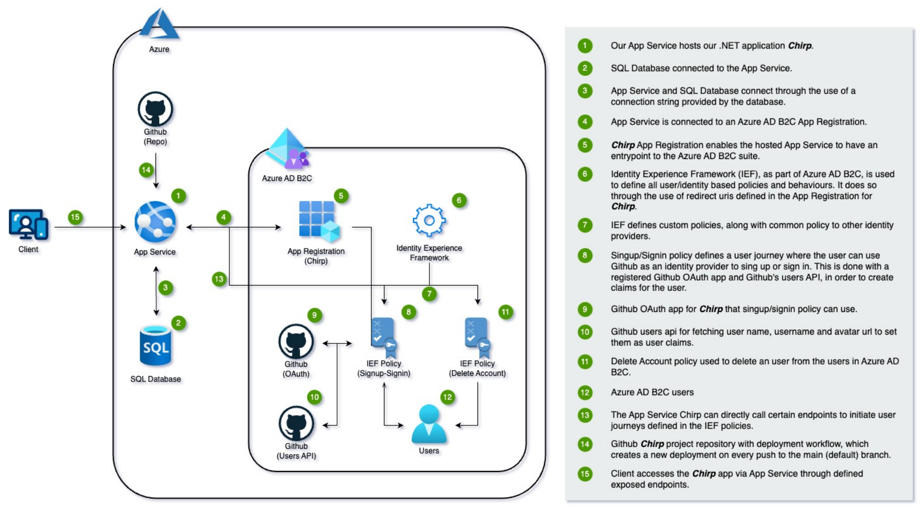
Andreas Bartholdy Christensen anbc@itu.dk
Marcus Andreas Aandahl maraa@itu.dk
Villads Grum-Schwensen vilg@itu.dk

1 Design and Architecture of *Chirp!*

1.1 Domain model

1.2 Architecture — In the small

1.3 Architecture of deployed application



1. Our App Service hosts our .NET application ***Chirp***.
2. SQL Database connected to the App Service.
3. App Service and SQL Database connect through the use of a connection string provided by the database.
4. App Service is connected to an Azure AD B2C App Registration.
5. ***Chirp*** App Registration enables the hosted App Service to have an entrypoint to the Azure AD B2C suite.
6. Identity Experience Framework (IEF), as part of Azure AD B2C, is used to define all user/identity based policies and behaviours. It does so through the use of redirect uris defined in the App Registration for ***Chirp***.
7. IEF defines custom policies, along with common policy to other identity providers.
8. Singup/Signin policy defines a user journey where the user can use Github as an identity provider to sing up or sign in. This is done with a registered Github OAuth app and Github's users API, in order to create claims for the user.
9. Github OAuth app for ***Chirp*** that singup/signin policy can use.
10. Github users api for fetching user name, username and avatar url to set them as user claims.
11. Delete Account policy used to delete an user from the users in Azure AD B2C.
12. Azure AD B2C users
13. The App Service Chirp can directly call certain endpoints to initiate user journeys defined in the IEF policies.
14. Github ***Chirp*** project repository with deployment workflow, which creates a new deployment on every push to the main (default) branch.
15. Client accesses the ***Chirp*** app via App Service through defined exposed endpoints.

Figure 1: Cloud Architecture

1.4 User activities

1.5 Sequence of functionality/calls trough *Chirp!*

2 Process

2.1 Build, test, release, and deployment

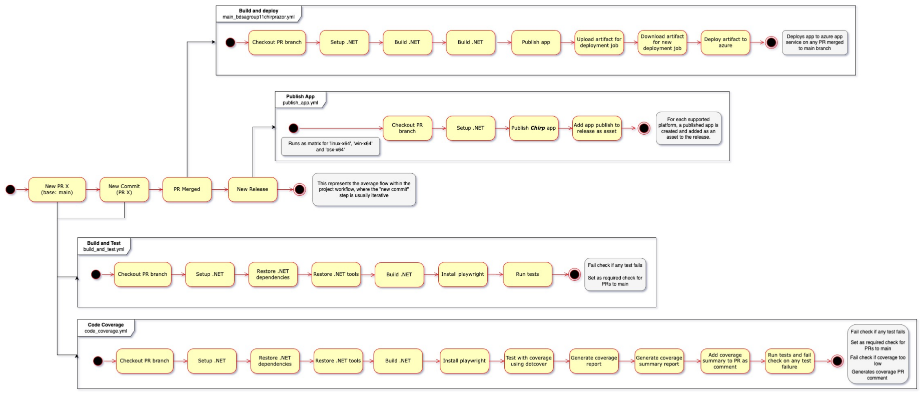

Build and deploy
main_bdsagroup11chirprazor.yml
Checkout PR branch
Setup .NET
Build .NET
Build .NET
Publish app
Upload artifact for deployment job
Download artifact for new deployment job
Deploy artifact to azure
Deploys app to azure app service on any PR merged to main branch
Publish App
publish_app.yml
Checkout PR branch
Setup .NET
Publish Chirp app
Add app publish to release as asset
For each supported platform, a published app is created and added as an asset to the release.
Runs as matrix for 'linux-x64', 'win-x64' and 'osx-x64'
New PR X (base: main)
New Commit (PR X)
PR Merged
New Release
This represents the average flow within the project workflow, where the "new commit" step is usually iterative
Build and Test
build_and_test.yml
Checkout PR branch
Setup .NET
Restore .NET dependencies
Restore .NET tools
Build .NET
Install playwright
Run tests
Fail check if any test fails
Set as required check for PRs to main
Code Coverage
code_coverage.yml
Checkout PR branch
Setup .NET
Restore .NET dependencies
Restore .NET tools
Build .NET
Install playwright
Test with coverage using dotcover
Generate coverage report
Generate coverage summary report
Add coverage summary to PR as comment
Run tests and fail check on any test failure
Fail check if any test fails
Set as required check for PRs to main
Fail check if coverage too low
Generates coverage PR comment

Figure 2: Workflows

2.2 Team work

2.3 How to make *Chirp!* work locally

2.3.1 Clone Github repository

To make *Chirp!* work locally, first clone the repository with the following command if you have SSH keys set up for Github:

```
git clone git@github.com:ITU-BDSA23-GROUP11/Chirp.git
```

otherwise, if you don't have SSH keys set up for Github, the following command can be used:

```
git clone https://github.com/ITU-BDSA23-GROUP11/Chirp.git
```

2.3.2 Install .NET

Thereafter, in order to set up the project, the main dependency you need is `.NET 7.0`. It can be downloaded from the from the *'Download .NET 7.0'* website. > *Make sure to download `.NET 7.0` and not `.NET 8.0`, as **Chirp** will not work otherwise.*

2.3.3 Set up Sql Server with Docker

Here are the steps to set up the sql server with docker:

2.3.3.1 1. Pull docker image

Run the following to pull the docker image

```
docker pull mcr.microsoft.com/azure-sql-edge
```

2.3.3.2 2. Run the image in a container

Replacing `<YOUR_DB_PASSWORD>` with a strong password (requires 1 upper case, 1 lower case, 1 number, and no special characters), run

```
docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=<YOUR_DB_PASSWORD>" -p 1433:1433 --name
```

2.3.3.3 3. Init secrets

If not done yet, run the following to create a secrets file

```
dotnet user-secrets init --project ./src/Chirp.WebService
```

2.3.3.4 4. Add DB password secret

Add the DB secret by running the following command, replacing `<YOUR_DB_PASSWORD>` with the strong password you generated earlier

```
dotnet user-secrets set "DB:Password" "<YOUR_DB_PASSWORD>" --project ./src/Chirp.WebService
```

2.3.4 Run the project

After the project is set up, it can now be run.

When running the profile, make sure to run with the https profile. This can be done with the following command:

```
dotnet run --launch-profile https --project src/Chirp.WebService
```

We would however recommend running it through an IDE, such a Rider, which automatically detects launch profiles.

Furthermore, since the project needs to be run on HTTPS, a certificate will be needed.

In our case, using Rider as our IDE, a HTTPS certificate was added after being prompted when running *Chirp* the first time. However, a certificate can also be added with:

```
dotnet dev-certs https
```

2.4 How to run test suite locally

2.4.1 Clone Github repository

To make *Chirp!* work locally, first clone the repository with the following command if you have SSH keys set up for Github:

```
git clone git@github.com:ITU-BDSA23-GROUP11/Chirp.git
```

otherwise, if you don't have SSH keys set up for Github, the following command can be used:

```
git clone https://github.com/ITU-BDSA23-GROUP11/Chirp.git
```

2.4.2 Install .NET

Thereafter, in order to set up the project, the main dependency you need is `.NET 7.0`. It can be downloaded from the from the *'Download .NET 7.0'* website. > *Make sure to download `.NET 7.0` and not `.NET 8.0`, as **Chirp** will not work otherwise.*

2.4.3 Install Playwright

Tests have only one requirement, which is needed to run end-to-end tests: Playwright.

First of all, the powershell dotnet tool is needed, which can be installed with the following command:

```
dotnet tool install PowerShell --version 7.4.0
```

After running the tests the first time, and failing, the cause will be due to playwright not be installed. This can can be solved by running the following command:

```
dotnet pwsh test/Chirp.WebService.Tests/bin/Debug/net7.0/playwright.ps1 install
```

Everything should now be set up in order to enable tests to run.

2.4.4 Run tests

To run tests, given the it is set up, simply run the following command:

```
dotnet test --verbosity normal
```

3 Ethics

3.1 License

3.2 LLMs, ChatGPT, CoPilot, and others