

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

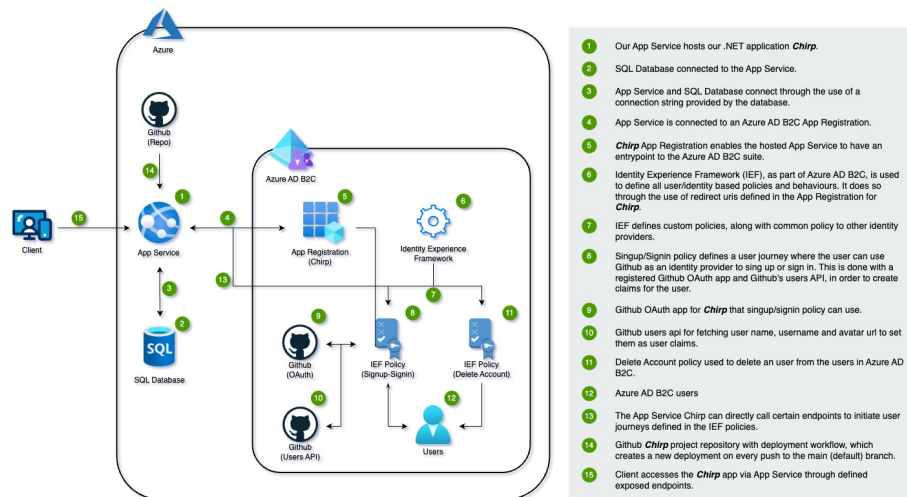


Figure 1: Cloud Architecture

1.4 User activities

1.5 Sequence of functionality/calls through *Chirp!*

2 Process

2.1 Build, test, release, and deployment

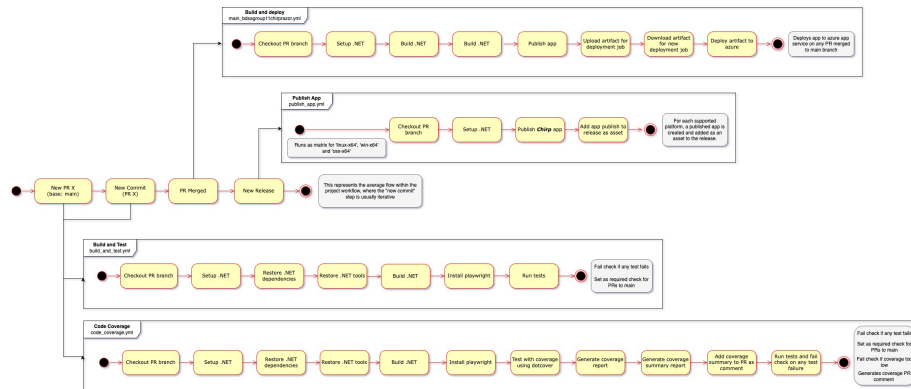


Figure 2: Workflows

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 azure-sql-server \
  -d mcr.microsoft.com/azure-sql-edge
```

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 as 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