Chirp! Project Report

ITU BDSA 2023 Group 25

Silas Wolff siwo@itu.dk Sebastian Blok segb@itu.dk
Karl Gustav Løhr kagl@itu.dk Adam Nørgård Aabye aaab@itu.dk
Atila Arianpour atia@itu.dk

1 Design and Architecture of *Chirp!*

1.1 Domain model

Our domain model is built around the core concept of an Author, which is central to the *Chirp!* application's functionality. An Author represents a user of the application, encapsulating their identity and interactions within the system. The UML class diagram model the key entities that make up our application.

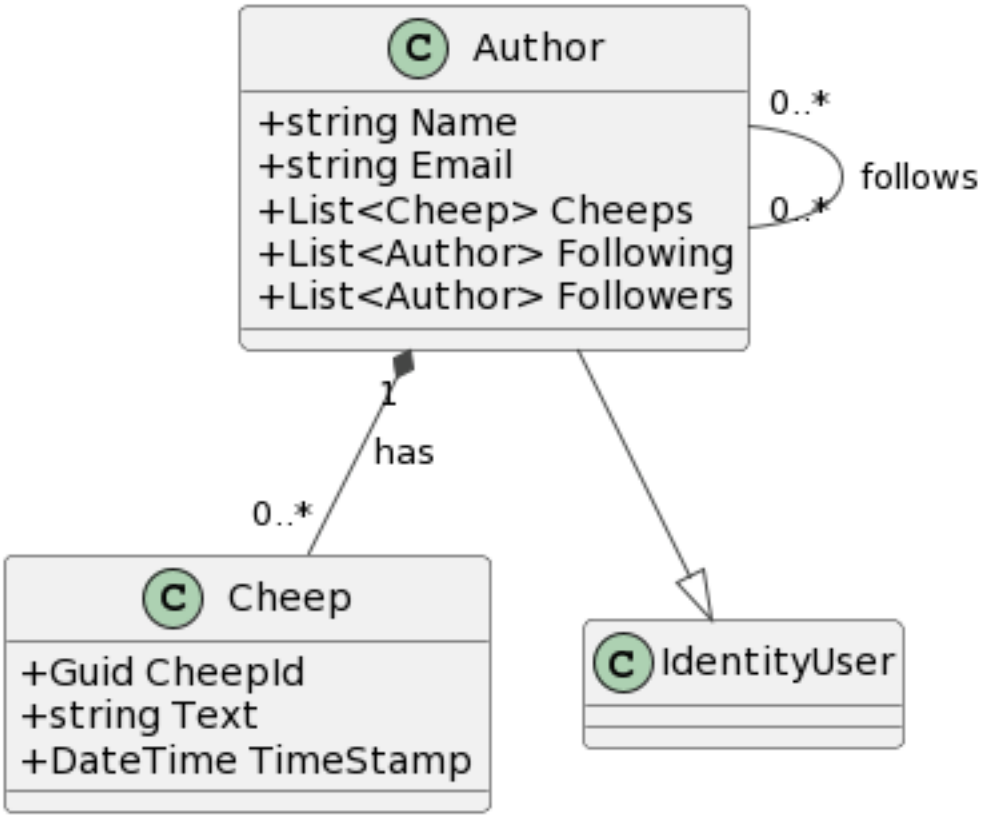

Figure 1: Domain Model

1.1.1 Author

The Author entity is an extension of the ASP.NET Identity's IdentityUser, inheriting features essential for authentication and authorization.

Each Author has a collection of Cheep entities, representing the messages, or posts, that the author creates within the application. This one-to-many relationship is depicted by a composition association, emphasizing that Cheeps are intrinsic to their Author and cannot exist independently.

In addition to creating cheeps, Authors can 'follow' other Authors. This is represented by a many-to-many self-referencing association, indicating that an Author can follow multiple other Authors and also be followed by multiple others. This relationship captures the essence of the application's social interaction capabilities.

1.1.2 Cheep

A Cheep is essentially a message, or a singular piece of communication, created by an Author. Each Cheep is uniquely identified by a Guid and contains the message text along with a timestamp of its creation. Cheeps is in a one-to-many composition with Authors, meaning that, Authors can have many cheeps, but cheeps must have exactly one author. Additionally, they have a strong life-cycle dependency.

1.1.3 Following

Hmm... lidt tbd, lad os lige snakke om implementationen, evt association class, ellers lister som nu

1.1.4 Reactions

TBD

is direct and exclusive, signifying that every Cheep is a direct output of an Author.

1.2 Architecture — In the small

Our application is separated into 3 main layers, that are common for the onion architecture

Layers: * core * infrastructure * web

Dependencies: * identity -> infrastructure * ef core -> infrastructure * core -> infrastructure * OAuth -> web * core -> web * infrastructure -> web *

1.3 Architecture of deployed application

1.4 User activities

1.5 Sequence of functionality/calls trough *Chirp!*

2 Process

2.1 Build, test, release, and deployment

2.2 Team work

2.3 How to make *Chirp!* work locally

2.4 How to run test suite locally

3 Ethics

3.1 License

3.2 LLMs, ChatGPT, CoPilot, and others

4 Perspektivering, eller overvejelser, eller noter, eller fri leg?

Den her sektion er ikke en del af templatet, men jeg tænker her kan vi skrive nogle overvejelser som vi har gjort og som vi måske/måske ikke vil have med i rapporten.

Overvejelse: Bør followers i class diagrammet være en association class?