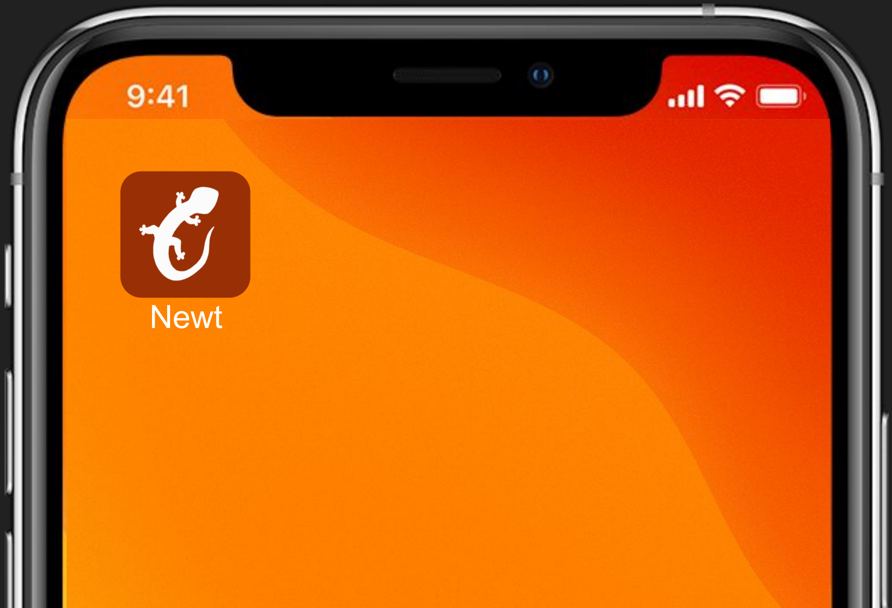
Newt

Optimistic Newts

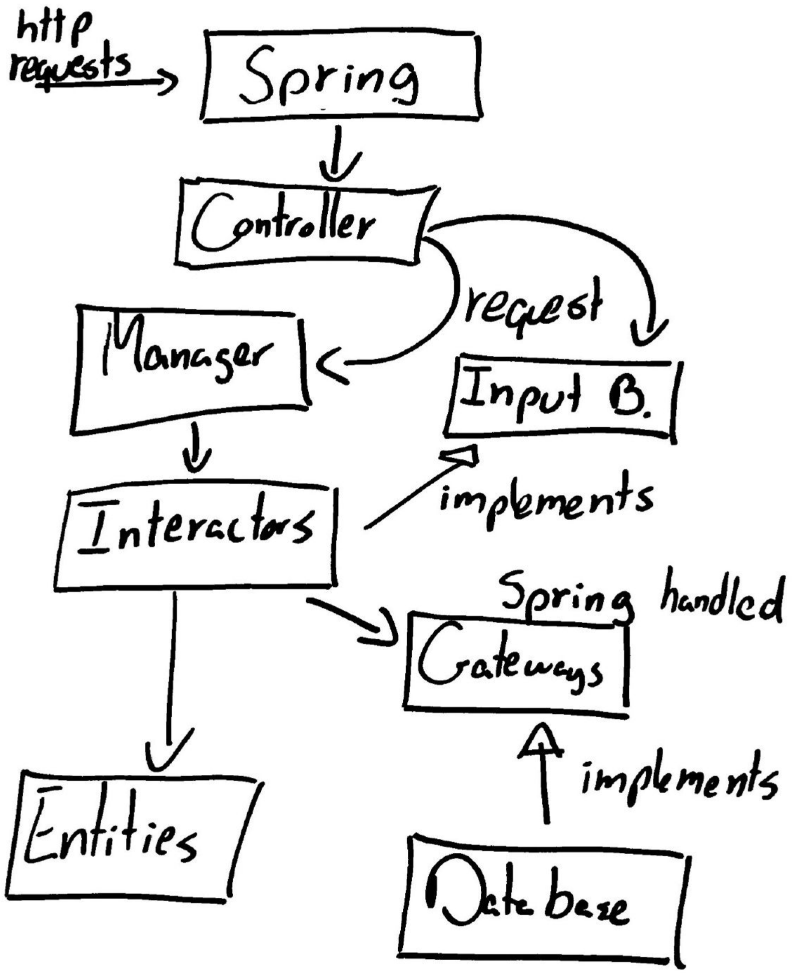
An App for Meeting People and Conversations

Overview

Our team is developing an application that facilitates meeting new people with similar interests through conversations. We want to give our users the opportunity to connect with other users without experiencing the social pressure induced by social media platforms



P.s. we did not make an ios app this just looks cool.


http requests
Spring
Controller
request
Manager
Input B.
Interactors
implements
Spring handled
Gateways
implements
Entities
Data base

Specification:

Since Phase 1, we worked towards completing our specification through our User Stories, and implementing a user-friendly front end for our web app.

Examples of what we changed in the remaining specification:

- Implementing Authentication
- Password Encryption using BCrypt and SpringBoot Security
- Interactors for Message, including conversation editing and deletion
- Conversation Controller
- Additional sorting algorithms for conversations, such as based on friends or your location

As a User, I want to be able to...

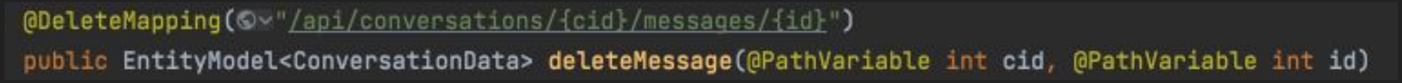
- Create, Log in and Log out of an account
- Safe and secure user and password protection
- User profile personalization and editing
- Browse through and communicate with other users
- Find, join, engage and contribute to interesting conversations and meet other users with similar interests
- Discover new topics which may interest me through recommendations, and filter-out topics that I am not interested in seeing
- Look for conversations both in my city and internationally to meet people with different perspectives
- Find, add, and remove people as friends

Web App

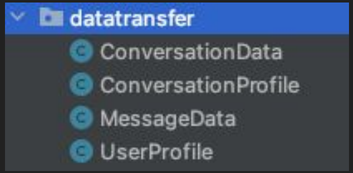
After phase 1, where we moved to using Spring Boot and PostgreSQL for our database and server, we turned our focus towards our Front End.

Things we worked through:

- Moving to and Configuring React
- Using React Routes to enable navigation among views of various components
- Familiarizing ourselves with JavaScript and CSS
- Connecting our front-end functionalities to our already existing back-end
- Data Transfer Objects for secure transfer of conversation, user, and messages



```java
@DeleteMapping("/api/conversations/{cid}/messages/{id}")
public EntityModel<ConversationData> deleteMessage(@PathVariable int cid, @PathVariable int id)
```



```
datatransfer
    ConversationData
    ConversationProfile
    MessageData
    UserProfile
```

Design Decisions

In phase 2, we continued to make some important design decisions.

Implementing our Front-End

- When implementing our front-end, we originally wanted to use Gatsby for implementing a fast and efficient front-end.
    - Gatsby: open-source framework that combines functionality from React for easy creation of **static** pages
    - React open-source front-end JavaScript library for building user interfaces based on UI components
- Static websites consist of a series of HTML files, each one representing a physical page of a website.
- Compared to static websites, which are purely informational, a dynamic website is more functional. It allows users to interact with the information that is listed on the page

Design Patterns

We have done our best to ensure our program uses design patterns where applicable to solve problems effectively and efficiently in our code. Here are some examples:

- **Facade Design Pattern:**

  The Facade design pattern provides a simple interface to a complex subsystem, containing many moving parts.

  Conversations and Users have a lot of small interactors, so to simplify our code, we created “manager” facade classes for Conversations and Users to delegate calls to specific interactors as needed, rather than having all of these small interactor methods together in one class.

- **Strategy Design Pattern:**

  The Strategy Design pattern facilitates defining family of algorithms in separate classes, while still making their objects interchangeable.

  An important feature of our app is recommending new conversations based on a User’s interests. So, to sort through which Conversations to recommend we have a few different algorithms in mind. We employed the Strategy Design Pattern for this, creating a ConversationSorter interface then creating implementing classes for each of our specific sorting algorithms. This way they are easily interchangeable.

Accessibility Report

Equitable use

- text-to-speech for those unable to type
- high-contrast mode for those with visual impairments
- Dark mode/light mode

Perceivable/Operable: Flexibility of Use, Simple and Intuitive, Low physical effort, Size and space for approach and use

- making content easily perceivable, regardless of which device they are using.
- making buttons, sliders, menus accessible regardless of the device they are using.
    - Making device specific apps

Understandable: Perceivable Information

- keeping content and the interface easy to navigate and understand

Open Questions

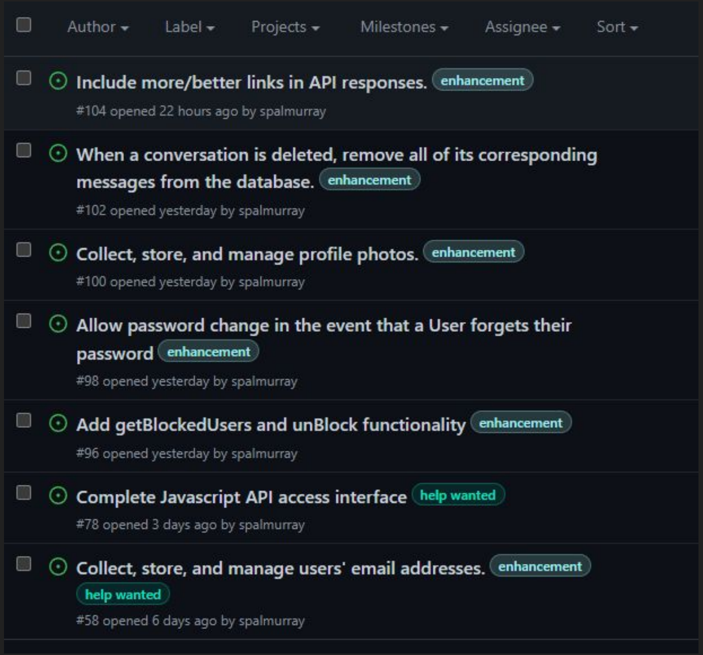
A few things we’re still considering :

- Based on our already existing front and back ends, what are the steps to finally bring an app to the public?
- What kind of database would we use in a real implementation of our app? What are the best database hosting platforms?
- How and where would we host the website itself?
- Are there any remaining security or functionality considerations we would need to make?

Looking Forward

Here are some ideas of the next steps in our development:

- Implementing more functions
  - Managing user emails
  - Password changing
- Implementing more features in the front end
  - Changing profile pictures
  - Implementing a few buttons that the user will interact with
- Making our Web App public



Include more/better links in API responses. enhancement
#104 opened 22 hours ago by spalmurray

When a conversation is deleted, remove all of its corresponding messages from the database. enhancement
#102 opened yesterday by spalmurray

Collect, store, and manage profile photos. enhancement
#100 opened yesterday by spalmurray

Allow password change in the event that a User forgets their password enhancement
#98 opened yesterday by spalmurray

Add getBlockedUsers and unBlock functionality enhancement
#96 opened yesterday by spalmurray

Complete Javascript API access interface help wanted
#78 opened 3 days ago by spalmurray

Collect, store, and manage users' email addresses. enhancement help wanted
#58 opened 6 days ago by spalmurray

DEMO

THANKS for listening :)