

# Timber

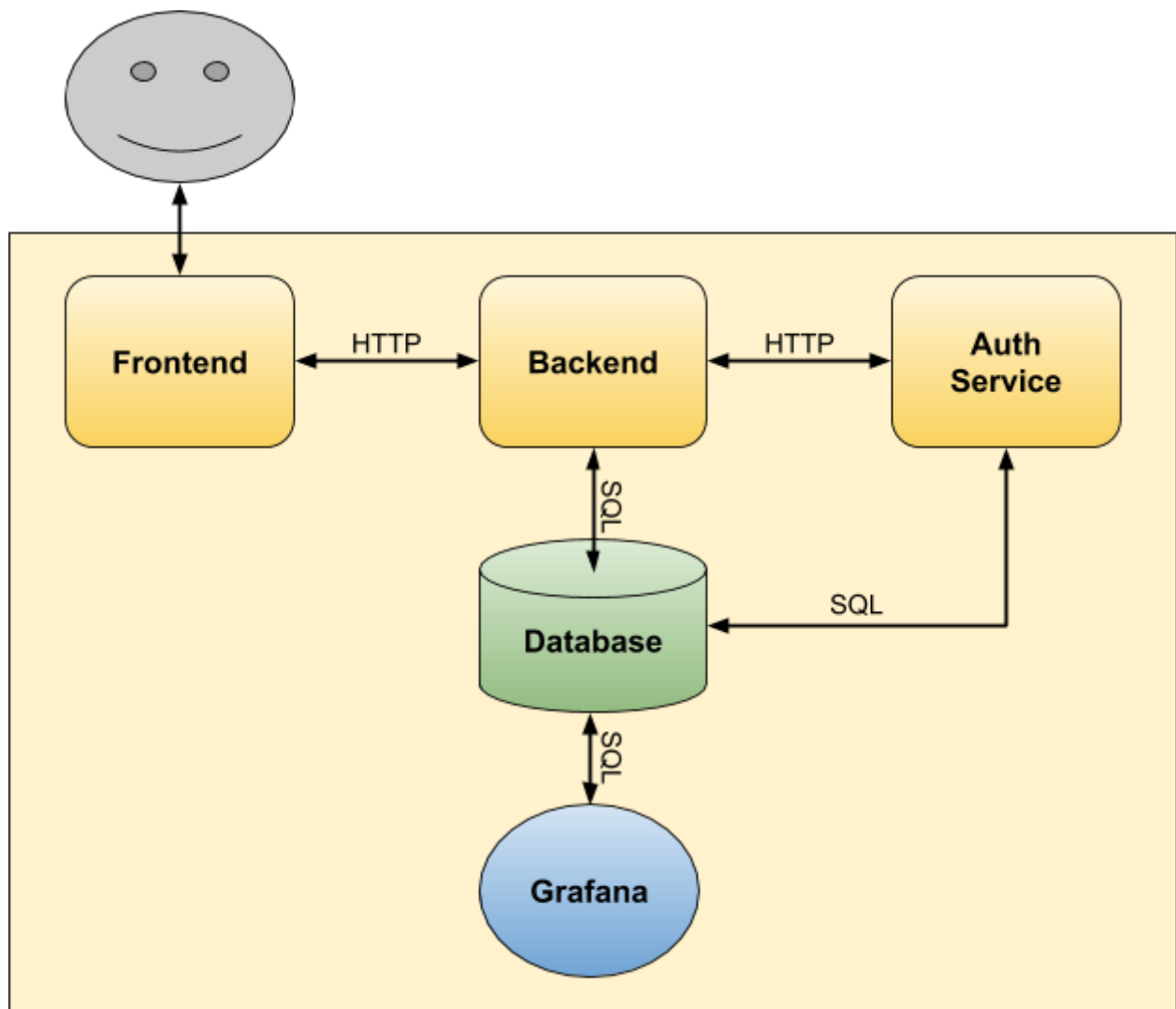
**Timber** service is a matrimonial service that can be used to create personal profiles and retrieve best matches for possible future partners. It's like Tinder, but crappier.

**GitHub:** <https://github.com/teodorpopescu/timber>

## 1. Overview

The service is composed of 5 microservices, all deployed as dockers:

1. **Backend** - which offers a RESTful API which can be used to request data from the service (like people that you might be interested in) and add/modify personal data
2. **Authentication service** - which offers a RESTful API which can be used to manage user's authentication credentials and retrieve/verify temporary access tokens to the server
3. **Frontend** - which can either be a website interface or a CLI interface that can be used to communicate with the server via its RESTful API
4. **SQL database** - which is used to store user's personal data and login credentials
5. **Grafana** - monitoring service which will be responsible of showing to the admins that number of users and other significant information



## 2. Microservices description

### a. Backend

The purpose of this service is two fold:

- Manage the user information
- Find suitable matches for each user

This service has the following RESTful API:

I. `/user` endpoint:

- GET (headers: username and token)  
Get the data stored for the given username, in JSON format  
=> Returns 200 [json] or 403 User missing  
or 401 Bad username or token
- POST (headers: username and token; body is a form with firstname, lastname, age and sex)  
Create a new user that will use the given password  
=> Returns 201 User created or 403 User already exists
- PUT (headers: username and token; body is a form that can contain one or more of firstname, lastname, age and sex)  
Update the data of the given user  
=> Returns 200 User update or 403 User missing  
or 401 Bad username or token or 400 Invalid [attribute]
- DELETE (headers: username and token)  
Delete the given user  
=> Returns 200 User deleted or 403 User missing  
or 401 Bad username or token

II. `/findMatch` endpoint:

- GET (headers: username and token; body is a form that can contain one or more of olderThan and youngerThan)  
Find all the matches for the current user that corresponds to their preferences  
=> Returns 200 [json] or 401 Bad username or token

III. `/everything` endpoint (debug purpose - should be disabled)

- GET (no headers)  
Return the content of the authentication table from the database in JSON format  
=> Returns 200 [json]

Note: All the operations can also return 400 Invalid username, 400 Invalid token or 500 Internal server error.

DockerHub: <https://hub.docker.com/r/tpopescu0710/timber-backend>

## b. Authenticator (the authentication service)

The purpose of this service is to:

- Manage the credentials by storing them in encrypted salted format (using AES) that will add a layer of security to our application
- Provide token management, that can be used to authenticate operations made by a user without them needing to send the password in each request. This tokens also expires, which will be a built-in log out mechanism that also provides an extra layer of security

This service has the following RESTful API interface:

`/user` endpoint:

- **POST** (headers: `username` and `password`)  
Create a new user that will use the given password  
=> Returns `201 User created` or `403 User already exists`
- **PUT** (headers: `username`, `password` and `new_password`)  
Update the password of the given user  
=> Returns `200 Password update` or `400 New password missing`  
or `403 User missing` or `401 Bad username or password`
- **DELETE** (headers: `username` and `password`)  
Delete the given user  
=> Returns `200 User deleted` or `401 Bad username or password`  
or `403 User missing`

`/getToken` endpoint:

- **GET** (headers: `username` and `password`)  
Get an authentication token that can be used as a parameter in order to authorize the operations that will be made by the given user. This token will expire after some time (~30 minutes)  
=> Returns `200 [token]` or `401 Bad username or password`

`/checkToken` endpoint:

- **GET** (headers: `username` and `token`)  
Verify the validity of the given token  
=> Returns `200 Valid token` or `401 Invalid token`

`/everything` endpoint (debug purpose - should be disabled)

- **GET** (no headers)  
Return the content of the authentication table from the database in JSON format  
=> Returns `200 [json]`

Note: All the operations can also return `400 Invalid username`, `400 Invalid password` or `500 Internal server error`.

DockerHub: <https://hub.docker.com/r/tpopescu0710/timber-authenticator>

c. Frontend

d. SQL Database

The database has the following tables:

- I. `Authentication` table, which stores (`Username`, `Password`, `Salt`), where password is encrypted
- II. `Users` table, which stores (`Username`, `FirstName`, `LastName`, `Age`, `Sex`)

e. Grafana (the monitoring service)