GraffLib Technical Specification

Mažvila Mantas, Namajūnas Joris, Švedas Gintautas,
Voroncov Ivan, and Supervisor: dr. Linas Bukauskas

Vilnius University Faculty of Mathematics and Informatics

October, 2021

Contents

1 Introduction

1.1 Overview

This is the technical specification for the project GraffLib. GraffLib is a software for sharing graffiti and seeing its changes. Our project uses the "model-view-controller" software design pattern. The user uses controllers, that manipulate models, which update the view that the user sees. This is done to separate what the user sees, and internal representation of information.

1.2 Purpose

This is the document describing the technical specifications of GraffLib. The purpose of this software is to allow users to upload, analyze and compare images of graffiti.

2 Overview of the whole system

2.1 System context

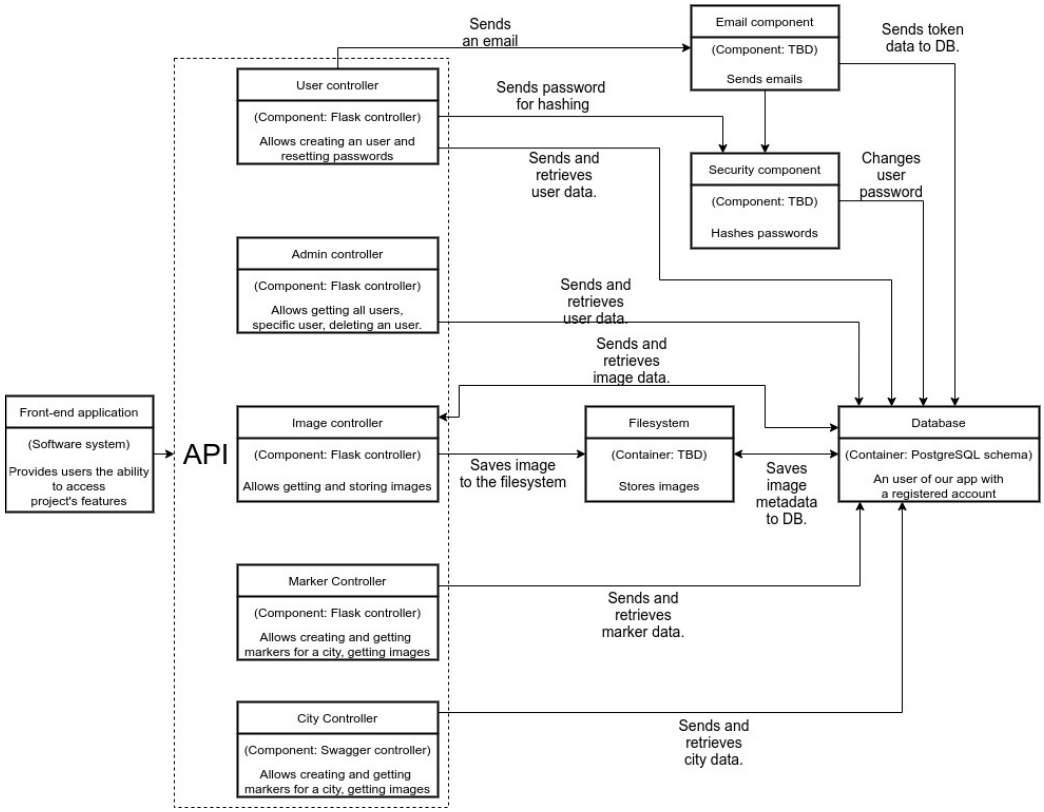

Figure 1: Partial context diagram

Our system can be accessed through an API, that will be hosted on a web server. The requests will be routed to this API. In figure 1, we see our context diagram, where the communication between our features is explained. As of writing this specification, the image uploading (Figure 2) is very basic (feature-wise), and it will stay simple for now. The images get stored in our file system, and the metadata gets sent to the database. Eventually we might add image compression to save space, scan the uploaded image for viruses. The users controller goes through a security component where passwords get hashed using bcrypt. There is possibility to add authentication for the Users controller. The other controllers (Admin, Marker, City) are used only for sending and receiving data to and from the database.

2.2 System artifacts

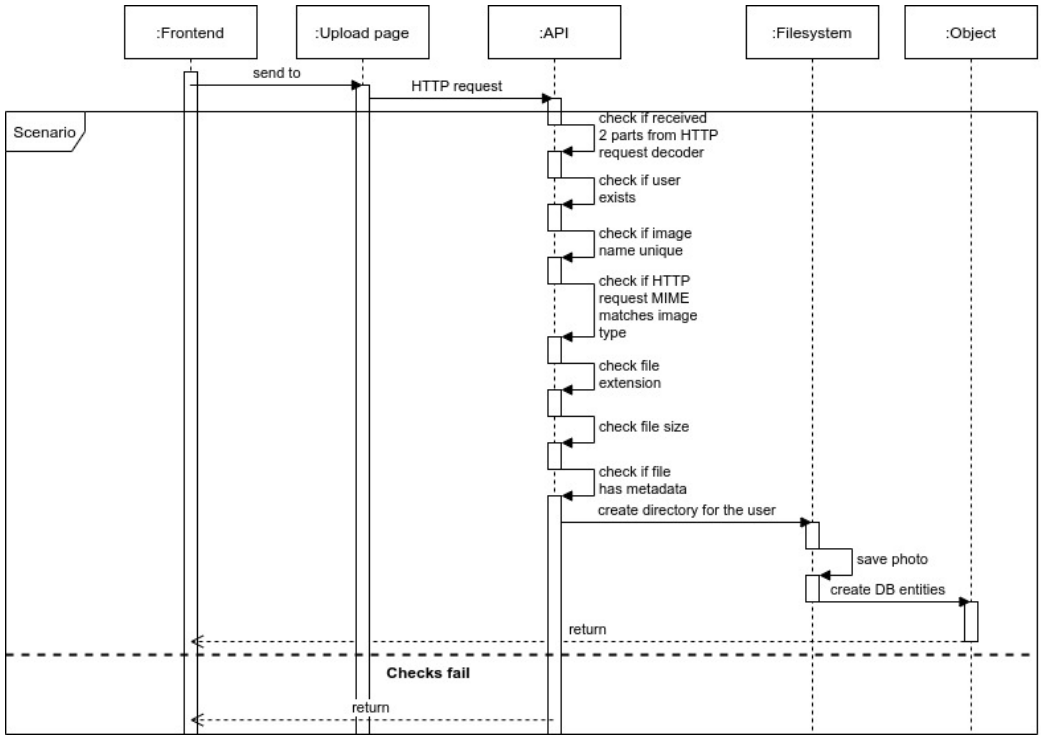

Figure 2: Image upload sequence diagram

The process of uploading an image is pictured in Figure 2. In the first scenario, it assumes that a picture gets through all the checks. The checks limit the size of the file, verify that the file is an image, has metadata and that it's sent by an user. In that case, the image is saved in the user's directory in our filesystem. The data gets sent to the database, and the user is sent back to the front-end.
If the checks fail, the user is sent back to the front-end, where it will show an error message.

3 High-level overview of the system internals

3.1 Structural aspects

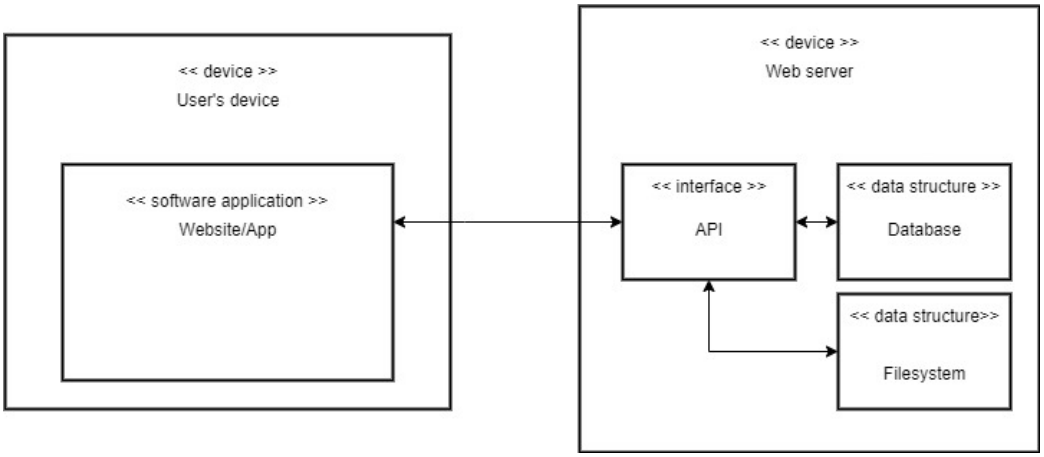

Figure 3: Structural aspects

The API, the database and the file system will most likely stay in a single virtual machine, running on Vilnius University Faculty of Mathematics and Informatics infrastructure.

4 Tools and technologies

4.1 Back-end

The following software tools are the most important to the GraffLib project's Back-end.

- Flask
- Python3
- Swagger
- PostgreSQL

We are using quite a few python libraries (42), but the most important ones are displayed below.

- Flask
- bcrypt
- GeoAlchemy2
- Jinja2
- marshmallow
- marhsmallow-geojson
- marshmallow-sqlalchemy
- psycopg2
- Shapely
- SQLAlchemy
- Werkzeug

4.2 Storage

These are the systems that we use for storage.

- PostgreSQL
- A file system

The database will store user accounts, image metadata, and other content. We chose PostgreSQL as our RDBMS because it allows for easy interaction with GIS.
We are also using a file system for image uploads. We think it will make image storage significantly easier when compared to uploading pictures to PostgreSQL or using some NoSQL database.