Final Project for Python Programming and Data Visualisation course

Jakub Jackowski 299248
Mateusz Milewski 299272

January 10, 2022

Contents

1 Introduction

The goal of this project was to make an interactive visualisation of data provided by device which measure pressure of feet on the ground. The data for this project have been made available courtesy of Mr Paweł Zawadzki, who is registering them for his PhD thesis research.

2 Architecture

Simplified architecture of the whole system is shown on the components diagram (Figure 1). There are four main modules and external measurements server integrated with the system via API, which. All components are separate applications running on distinct Docker containers, so they can work asynchronously.

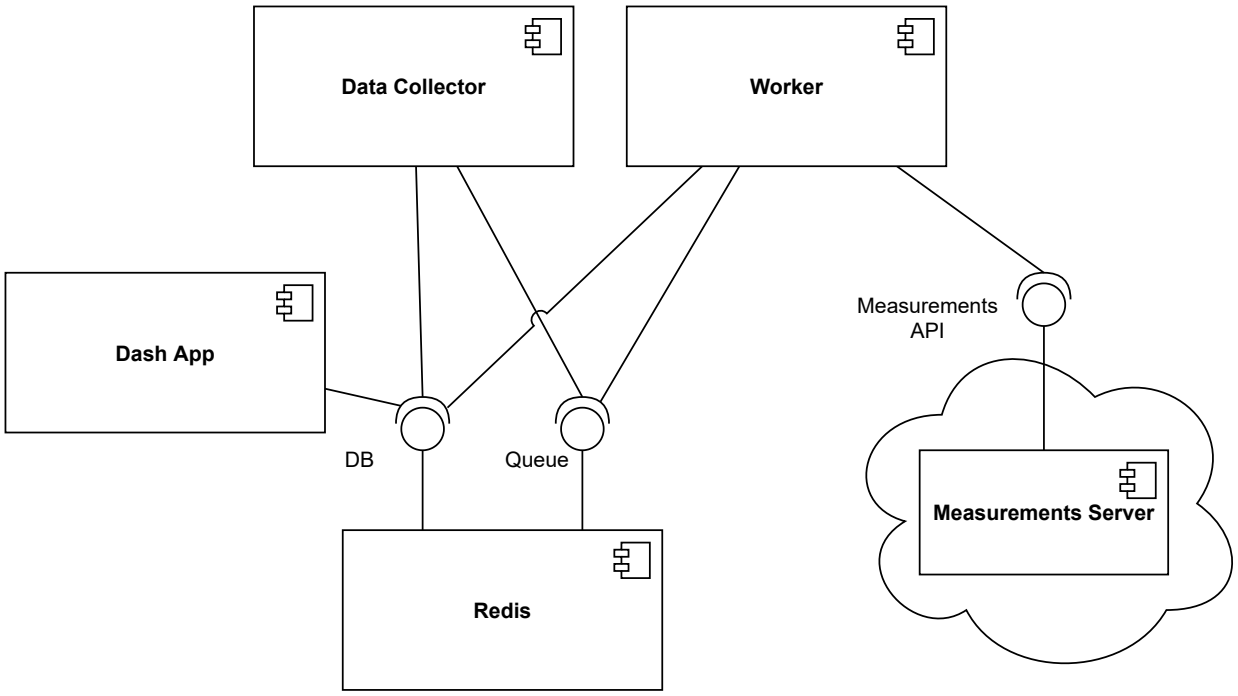

Figure 1: Architecture diagram

2.1 Dash app

This module is responsible for interactive visualization which was fully created using Plotly Dash library. We divided our front-end application in 3 sections:

- Global inputs section
- Current data section
- Advanced analyze section

2.1.1 Global inputs section

In this section, you can find two inputs that have impact on the whole app. Play/pause button which enables and disables collecting data and the second one is person selector which gives option to switch between patients measurements that are currently displayed.

2.1.2 Current data section

This section is responsible for displaying current measurements. Here you can find table with 6 rows, where each one stands for one sensor and also as suggested in functional requirements we added a foot visualisation by creating a custom dash component. Thanks to which you can easily analyze current measurements.

2.1.3 Advanced analyze section

Last section stands for advanced analysis. This panel contains a big line graph which shows previous measurements up to 10 minutes back. On top of diagram there is multiple dropdown selection where you can choose which traces will be displayed on graph. There is also binary button which gives option to switch between displaying normal measurements or anomalies that were detected earlier by data collector.

2.1.4 Callbacks

As part of the creation of the application, we made 3 custom dash components: Foot visualization, patient selector and play/pause button. Thanks to them, we were able to handle more complicated interactions with React help. Even so, we still used dash callbacks to manage interaction between components. We used one global interval for data updating on each section. The interval can be pause or resumed by user in global inputs section.

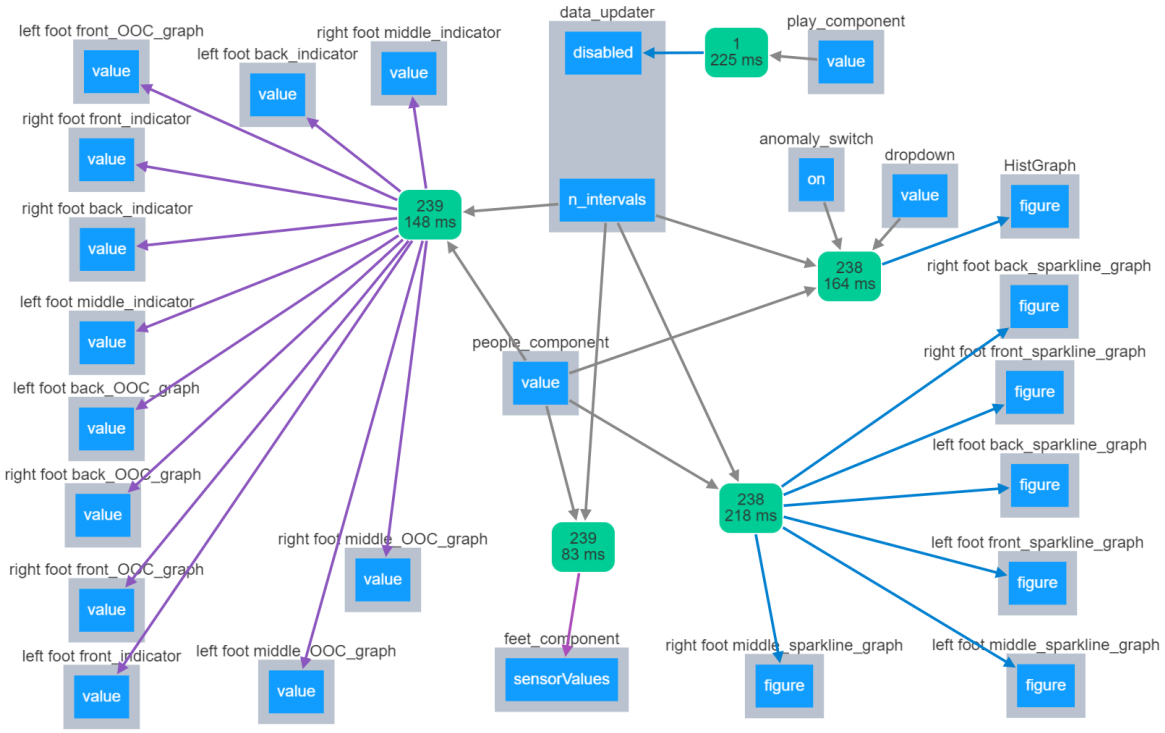

Figure 2: Plotly Dash callbacks diagram

2.2 Data Collector

Data collector application logic is mainly located inside DataCollector class. There is a method `run()` which starts a process in endless loop. In each iteration, Data Collector

creates jobs for retrieving data from the API and enqueues them in Redis. Moreover, it removes data from database which is older than 10 minutes.

2.3 Redis

In our system Redis is playing double role. Firstly it is our storage for the data retrived from the API. Secondly we use it as our tasks queue. All components use database in different ways. Workers are saving the data, Data Collector is removing old measurements and Dash App is reading the data stored. Queue is used only by Data Collector and Worker for enqueuing and getting tasks to work with.

2.4 Worker

Workers are processes which are managed by Supervisord process control system. In configuration file we specified number of worker processes and how they should behave. Supervisor will restart particular instances if they fail or timeout. Worker is responsible for sending request to API, retrieving data from it and save it in Redis database. Then it will check for anomaly and if it occurs, it is additionally saved under the different key.

3 How to run

To start, project download the archive file and extract it. Inside the extracted folder, you can find **README.md** file where you can read how to run project. In case you don't want to browse that file, here is a short version of that:

- Before starting project make sure you are connect to faculty vpn.
- Install Docker because the whole project is containerized.
- To start app, run **docker-compose up** inside main directory.
- After couple seconds all containers should be running and collecting data in real time.
- You can find interactive visualisation in your browser on http://localhost:8050/

Troubleshooting: Sometimes running python workers inside containers can cause problems with blocking each other. In that case don't worry if app does not show up in browser at first place. You have to wait a couple seconds and the app should start running.

4 Application design

Application interface is designed to be clear and simple for the ease of use. We decided to keep everything in dark theme as it is the most preferable color scheme for general computer users.

Figure 3: Application interface

Application design shown on Figure 3 is consistent with what we planned in the first milestone. The only change is an extra panel with start/stop button and new feet component design which represents more realistic appearance.