Software Architecture Report

Date: 29.01.2024

*(last commit ID on main:* ***5945512a9bbc109810cb06de31f1409e96c2ccef****)*

1. Purpose of the project:
2. [Description](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/tree/main?tab=readme-ov-file#description)
3. [Feature mindmap](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/eduards/github-action/livrabil-1/diagrams-screenshots/feature_mindmap.jpg) - we wanted to implement the features marked with \* and so we did, along with fixing bugs along the development process
4. Guides:
5. [Cloning the project and working on tasks](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/tree/eduards/github-action?tab=readme-ov-file#work-policy)

Running locally – [Backend setup](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/tree/eduards/github-action/api#budget-tracker-api) and [Frontend setup](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/eduards/github-action/ui/README.md#budget_tracker)

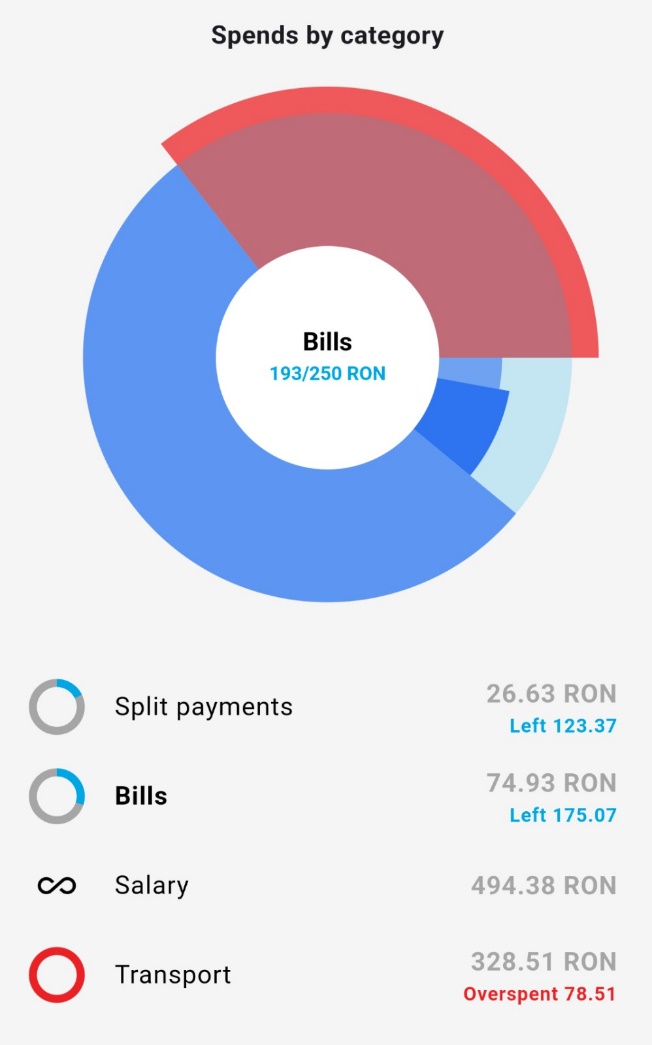
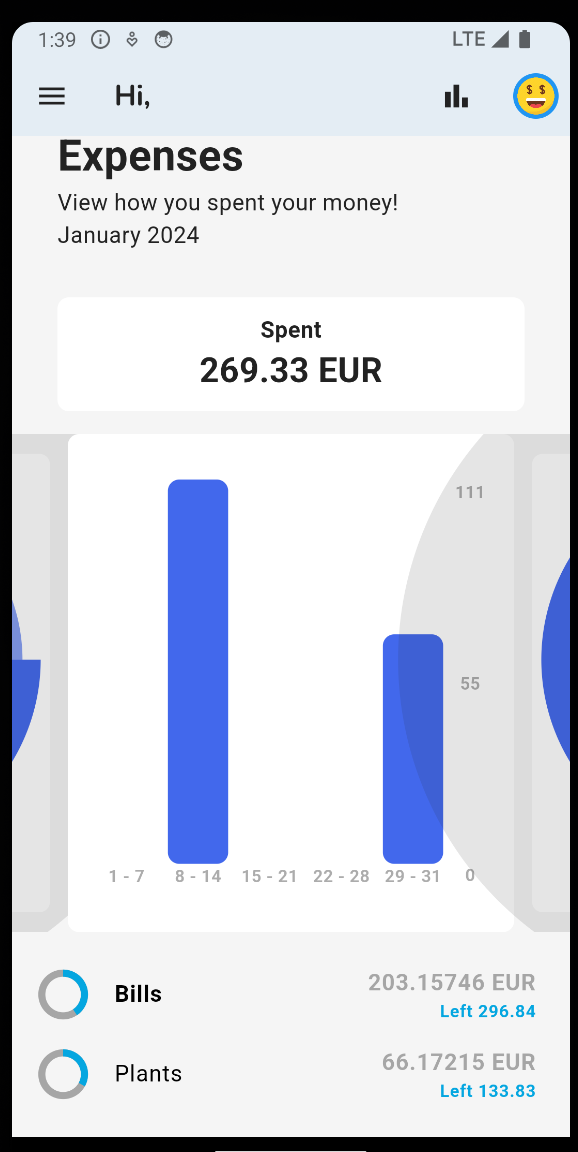
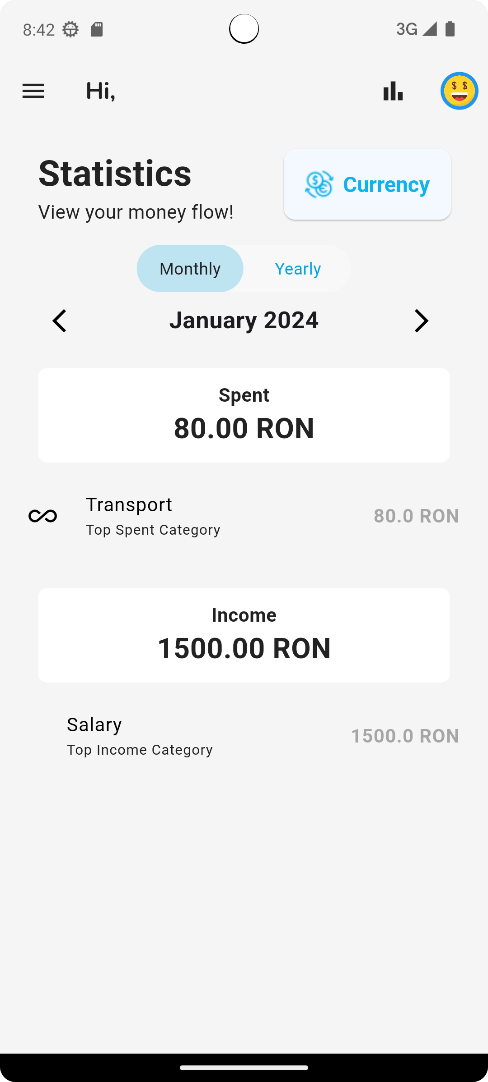
1. Build the project

We created a [dockerfile](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/api/Dockerfile) in which we specified how the project’s backend should be built. We integrated the build step into our CI pipeline. The result of this step is a docker image that you use for deploy. To build the frontend you should run the project locally and build it using flutter’s CLI that will result in a mobile application that you can run on emulators or physical device.

1. Deploy the project

To deploy the project you need to use the docker image from the build step by updating the docker-compose image version. We are hosting our application on a private server with a public domain name.

1. Application entry points:
2. **Data sources** - We are using as main data source a MySQL database and a public currency API.
3. **Data inputs** - We are providing a mobile user interface that accepts a wide range of inputs such as text fields, pickers etc. We also provide an API for integration with other projects.
4. **Configuration files** - We have a [configuration for the backend](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/api/src/main/resources/config/env.sample.properties) that sets the database, mail server and java security properties and credentials. We also have a [front-end configuration file](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/ui/lib/config.dart) that sets the API entry point.
5. High level diagrams of the architecture
6. [User journey map](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/eduards/github-action/livrabil-1/diagrams-screenshots/user_journey_map.png) and [Integration points](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/eduards/github-action/livrabil-1/diagrams-screenshots/integration_points.png)
7. **Our most valuable feature** is the Income & Expense Statistics dashboard.

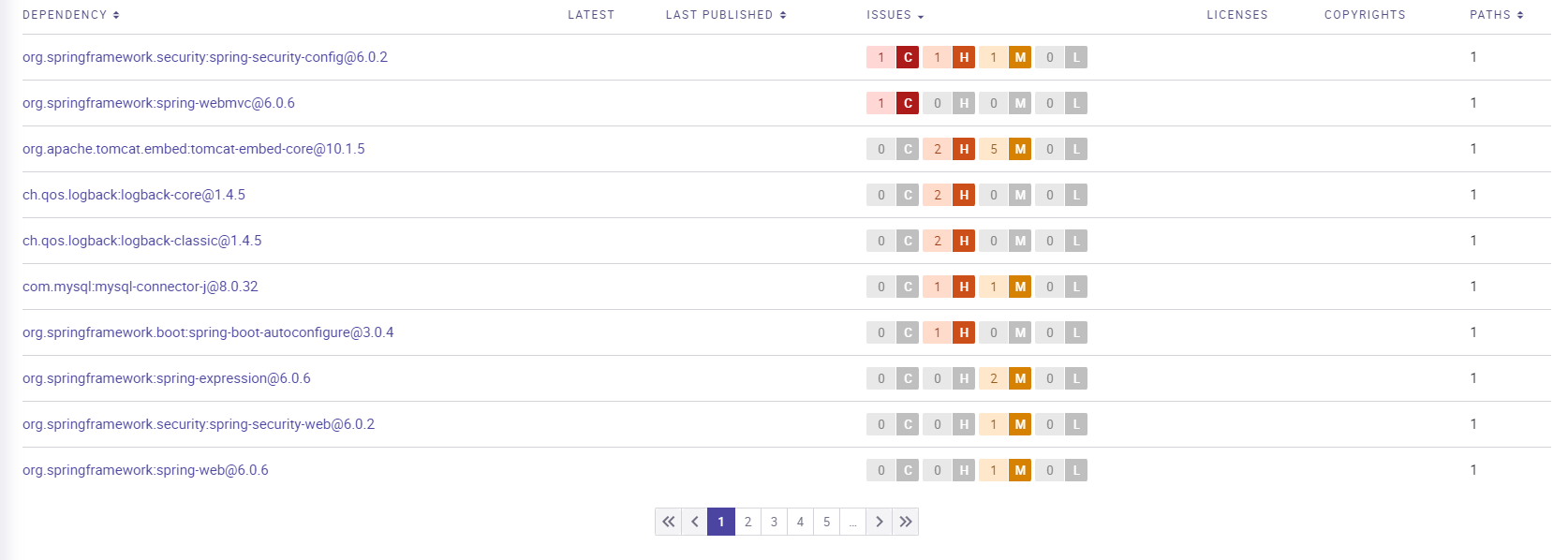


1. Deployment plan:
2. The application is deployed on a private server on-premise.
3. On every push on any branch the [test pipeline](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/.github/workflows/maven.yml) is triggered that builds and runs the tests on the current branch. On every push on main the [build pipeline](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/.github/workflows/docker_build.yml) is triggered that dockerizes the maven artifact and results in a docker image that will be used for deploy.
4. Description of the QA process:

* [Unit tests](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/api/src/test/java/com/budgettracker/api/budgeting/unit/services/StatisticsServiceTest.java) that validate isolated functions.
  + Backend unit testing: using **JUnit** with Mockito
  + Frontend unit testing: using Mocktail
* [Integration tests](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/blob/main/api/src/test/java/com/budgettracker/api/feature/StatisticsControllerTest.java) that validate the connection between services, repositories and other more complex components.
  + H2 in-memory database
  + SpringBootTest
* [Performance tests](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/tree/main/performance-tests-results) that help detect issues regarding complexity and scalability of the app.
* **Manual testing** along the development process and also in dedicated tasks to report bugs

1. **External dependencies** included in the project:

In order to manage project dependencies, for the backend part we used a package manager, Maven - and a list with all dependencies managed by Maven can be found in [mvn-dependencies.txt](https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-budget-tracker/tree/main/livrabil-2/mvn-dependencies.txt).

1. [ExchangeRate API](https://www.exchangerate-api.com/docs/supported-currencies) – we used it to get the exchange rates in order to convert the expenses and incomes to the desired/default currency in the statistics dashboard.
2. We used **Snyk** to get this overview of **vulnerabilities** in dependencies.