Design of the report management system

By:

Dembegiotis, Callista	s5822637
Nieuwenhuis, Demian	s5163609
Giummarra, Alessio	s5793181
Mura, Maria	s4778898
Willmann, Leo	s5253144

Faculty of Science and Engineering, University of Groningen
WBCS017-10: Software engineering

June 2, 2024

Table of Content

Introduction

This document is one of three parts that will discuss the report management system developed for Gomibo. This document will discuss the design and technologies of the application.

The web application has a report page, where employees can submit complaints easily by answering a questionnaire and can optionally upload an image. On the maintenance team end, there will be a login page, so that only authorized personnel can access the information. They will be able to see a run-down list of submitted reports and their corresponding details summary. They will see a more detailed overview of the report when navigating to it. The maintenance personnel will be able to edit the priority and status of the reports according to their judgment and can also change the questionnaire at any time, using the designated page for this.

Technology

To program and implement the web application, we will be using the following technologies.

Laravel: a framework for the PHP language which follows the Model View Controller (MVC) design architecture. PHP is an open source general purpose server side scripting language used mainly in web development to create dynamic websites and applications.[1] Laravel comes with many in-built features. PHP has been chosen by Gomibo for the backend.

Gomibo had us choose between react and python for the frontend, and we chose react as it is aimed for web development. React is an open source front-end Javascript (JS) library for building user interfaces based on components. It can be used in conjunction with other front-end frameworks because React is only managing the user interface and rendering components to the Document Object Model.[2] We use JS to run dynamic client side web pages.

Our mysql database is an SQL relational database to quickly store and access information.

Phpmyadmin is a database management interface which lets you view and edit the database through a browser. Phpmyadmin also allows you to export the database tables, and many other features we did not use extensively.

Docker is used to containerize the application. This separates the application from the machine infrastructure and makes sure that no matter on what machine you run it, the application runs the same. The use of docker has been endorsed by Gomibo.

---

[1] https://www.geeksforgeeks.org/what-is-php-and-why-we-use-it/
[2] https://en.wikipedia.org/wiki/React_(software)

Architecture

Conceptual view

Conceptual view diagram

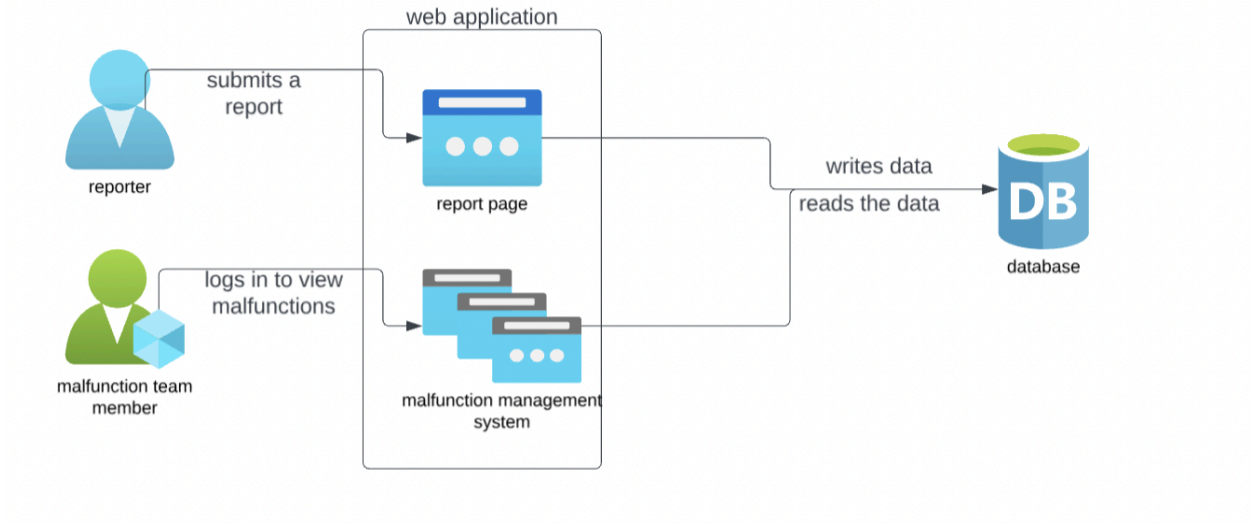

Figure 1. Conceptual view of the application

In this diagram, we can see how our web application was designed. There are two sides to it. One is for the reporter, for submitting a complaint. The other side, created for the malfunction team, requires login with credentials. The malfunction management system is made up of more pages with different functionality. Besides the login, there is a page for adding questions, one for viewing single reports and one for changing priority and status. All of this is possible with the use of the database that is connected to both sides of the web application and stores and sends the data when necessary.

Database

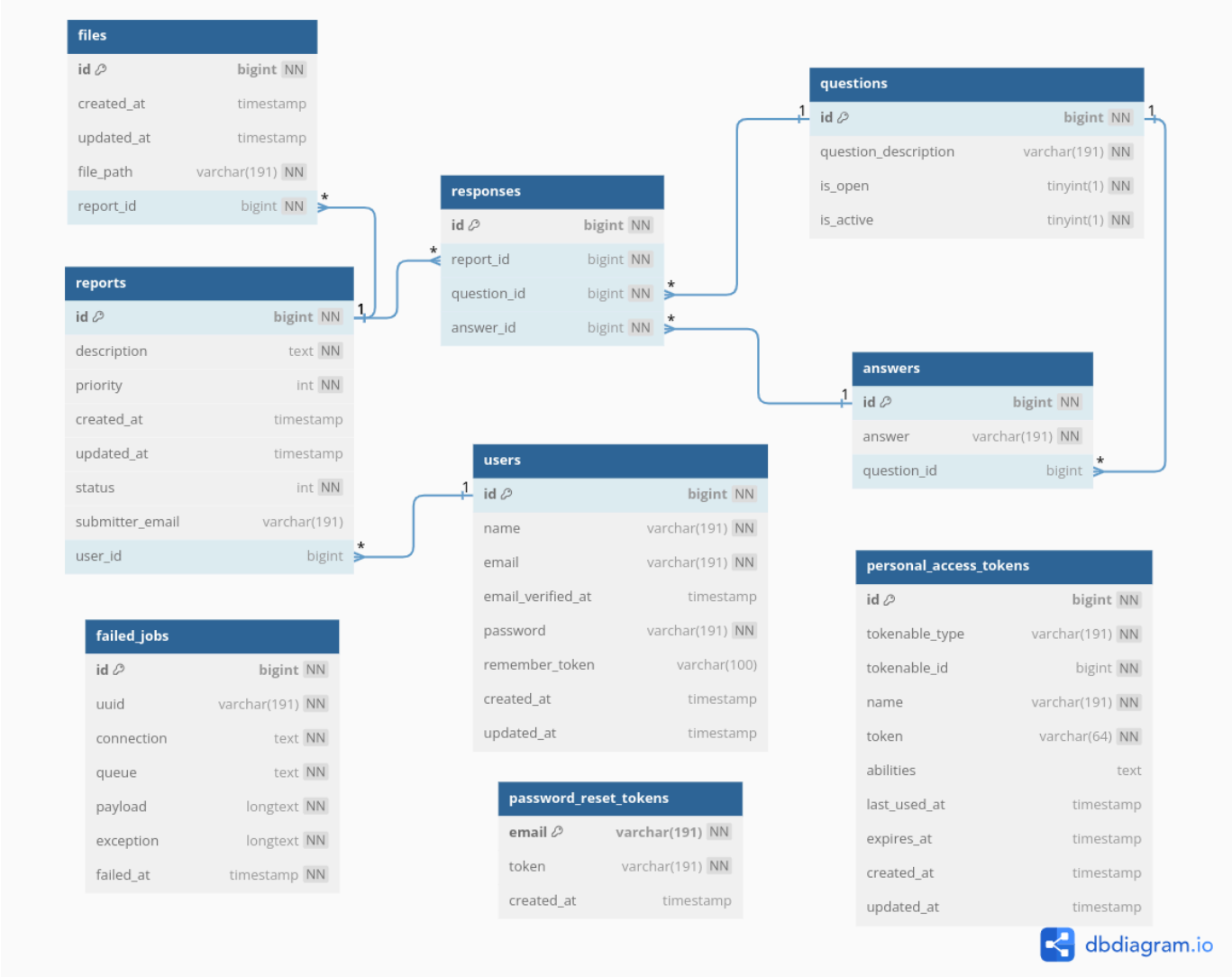

Figure 2. Database diagram created by dbdiagram.io

The database has multiple tables that are responsible for different things. This diagram is a visual representation of the actual database due to using phpmyadmin to export the current database configuration into dbdiagram.io for a nice visual. The reports table is the one that ties all other tables together as it stores all report information. The users table stores the credentials of the maintainers, and has a many to one relationship with the reports table. This allows for the function that one maintainer is assigned to many reports. The response, question, and answer tables ensure a database safe environment in which the questions and answers can be changed. A report has many responses which consist of a question and an answer. To enable the functionality of file uploading, the files table has been created with metadata and the file path to where the file is stored in local storage.

API

The Application Programming Interface (API) is responsible for the communication between the frontend and the backend. The frontend will send requests to the backend that conform to the interface defined by the API. The following section will quickly describe the endpoints that have been created. For more detailed information, visit the GitHub repository where we have created an OpenAPI 3.0 specification file, which when opened in the swaggerUI interface will show a graphical representation of the various API schema's and endpoints.

- /api/login
  Available HTTP methods:
  - POST [request body: email, password], lets the server know that a user with the specified credentials wants to login. The server will return some user information and store an access token to their browser's cookies. This endpoint is used by maintenance personnel, US1-M1.

- /api/logout
  Available HTTP methods:
  - POST [request body: empty], lets the server know that the user wants to logout. The server will know which user it is as they send their cookie token with the request and then their browser's cookies will delete the now invalid access token. This endpoint is used by maintenance personnel, US1-M1.

- /api/user
  Available HTTP methods:
  - GET [request body: empty], Gets the user information (excluding password) linked with the cookie token in the request. This endpoint is used by maintenance personnel, US1-M1.

- /api/reports
  Available HTTP methods:

    - GET [optional parameter: filters], retrieves report entries from the report table in the database. This endpoint is used by maintenance personnel, US3-M3.
    - POST [request body: report], creates a report entry into the database. The ID and creation time are automatically generated. This endpoint is used by reporter persona, US2-M2 and US2-S5.

- /api/reports/{id}
  Available HTTP methods:
    - GET [required parameter: id], retrieves the report entry from the database of the specified id with additional responses and files linked to it. This endpoint is used by maintenance personnel, US3-M2 and US3-S6.
    - PATCH [request body: report], updates the report entry to match the request body. This endpoint is used by maintenance personnel, US4-M1.

- /api/questions
  Available HTTP methods:
    - GET [optional parameter: filters], retrieves question entries from the question table in the database. This endpoint is used by maintenance personnel and report persona, US2-M1.
    - POST [request body: question], creates a question entry into the database. The id is automatically generated. This endpoint is used by maintenance personnel, US5-M1.

- /api/questions/{id}
  Available HTTP methods:
    - GET [required parameter: id], retrieves the question entry from the database of the specified id. This endpoint is used by maintenance personnel.
    - PATCH [required parameter: id] [required body: is_active], updates the is_active boolean in the questions table. This endpoint is used by maintenance personnel, US5-M2.

- /api/answers
  Available HTTP methods:
    - POST [request body: answer], creates an answers entry into the database. The ID is automatically generated. This endpoint is used by Report persona, US2-M4.

- /api/files/upload
  Available HTTP methods:
    - POST [request body: file], creates a file entry into the database with file information and the file itself is stored locally. The id is automatically generated. The generated_name is 20 random characters. This endpoint is used by the reporter, US2-S6.

- /api/files/{file}/download

Available HTTP methods:

- GET [required parameter: fileID], gets a file from the database based on its ID. This endpoint is used by the maintenance personnel, US3-S6.

Security

The application has security features to only allow the maintenance team access to the submitted reports. As seen in the API section, the api has endpoints for logging in and out. The authentication and authorization of these requests is handled by Laravel's Sanctum middleware. The Sanctum middleware also covers the following endpoints to ensure only logged in users can access them:

- GET /api/reports
- GET /api/reports/{id}
- PATCH /api/reports/{id}
- POST /api/questions
- PATCH /api/questions/{id}
- GET /api/files/{file}/download

When an unauthorized user tries to access them they get a `401 unauthorized` response message.

Sitemap

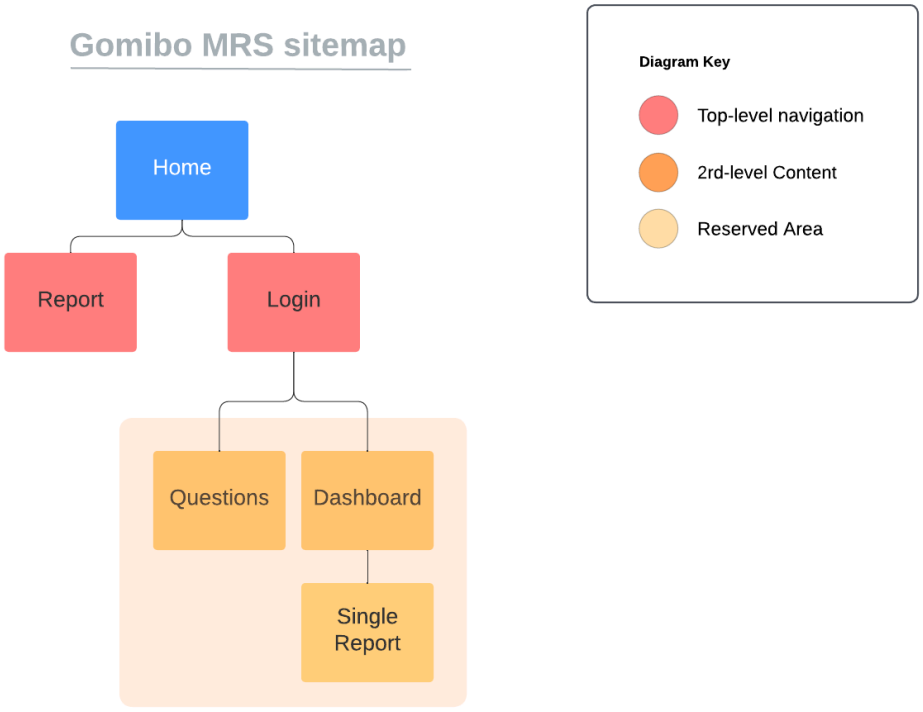

Figure 3. A graphical representation of the pages the frontend can serve. The reserved area requires authentication.

Figure 3 shows a view of the pages the frontend will serve. From the Homepage the Report page that can be used to report something in, then we have the Login page that maintenance personnel can use to access the Reserved Area, there they can find the Questions page, where they can make new questions and the Dashboard where they can see a summary of all the reports and edit their status. The single report page is used to view a single report with all of its question responses.

Module view

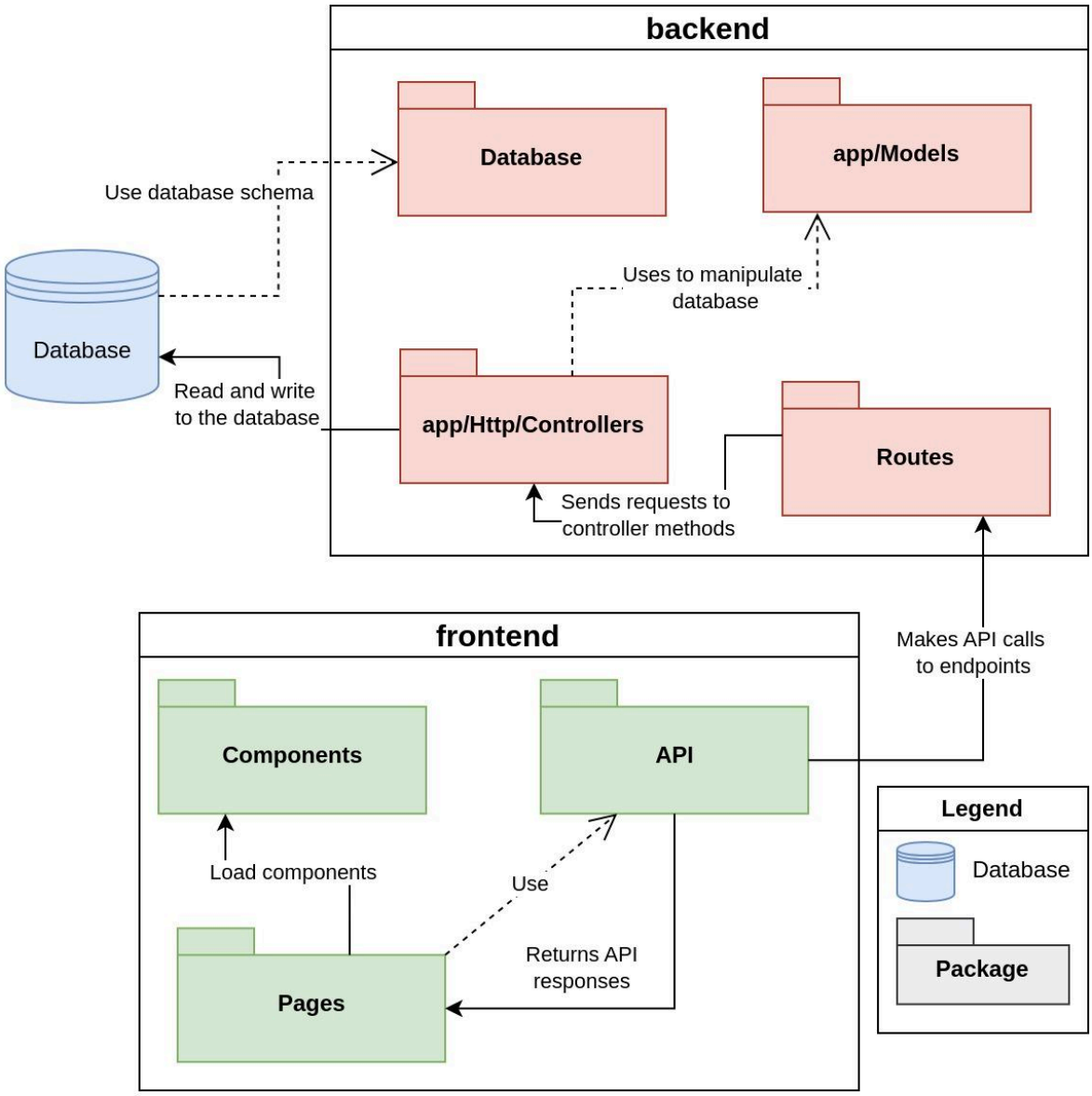

Figure 4. Application module view that shows the frontend, backend and database with their important packages and their interactions with each other.

Figure 4 shows the most important packages for the application and how they connect and communicate with each other. You can see that the backend and the frontend are their own systems that are only coupled by the API endpoints. The laravel backend follows the Model View Controller (MVC) pattern, where the model is the database with the schema defined in the database package, the view is the app/Models package, and the controller is the app/Http/Controllers package. The frontend has pages with their components and through user input, can navigate through the pages. Some pages will use the API to load the initial page with

information (i.e. the questionnaire or the dashboard view), and some will use the API only after user input to fill in certain fields (i.e. report submission).

Laravel Packages

Laravel is a framework for the php language and on a project creation generates a lot of packages. More packages than that were shown in the module view, and as we only use the Laravel backend for API and database handling, a lot of these packages go unchanged. Figure 5 shows the packages color coded by how changed they are from the initial laravel framework.

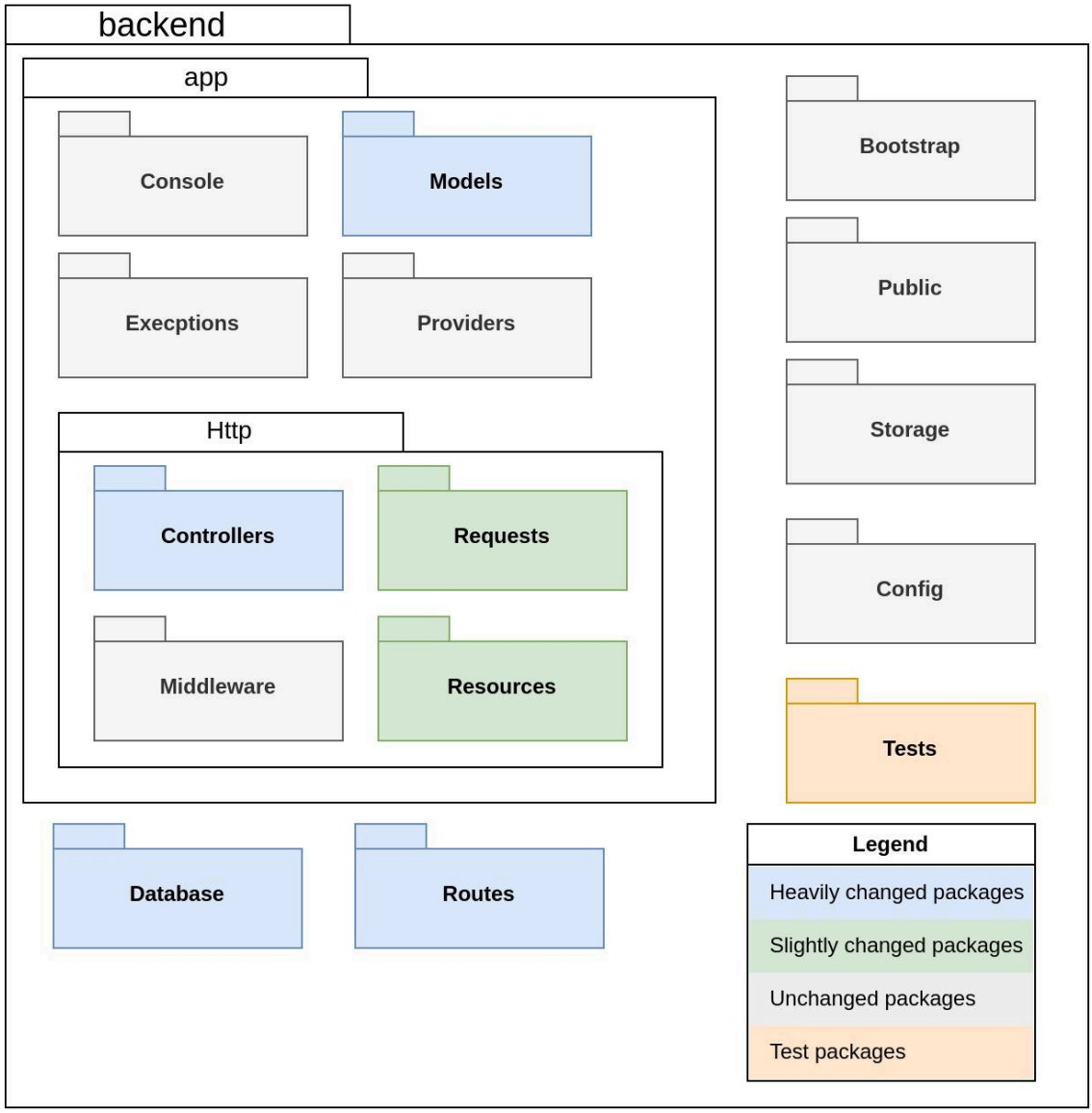

Figure 5. Packages of laravel backend color coded by how changed they are from the initial laravel framework.

Execution view

Reporter submit report Interaction Diagram

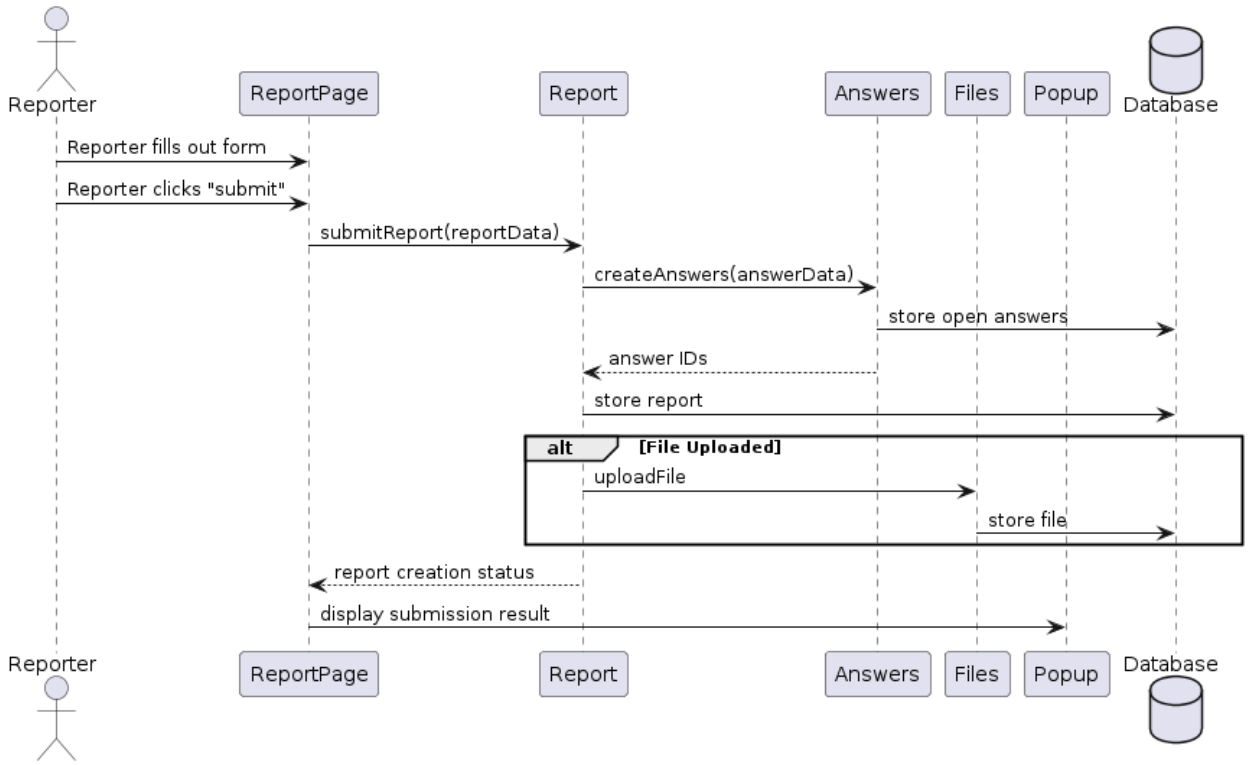

Figure 6. Interaction diagram for a reporter submitting a report

The diagram above is a sequence diagram of a reporter creating a new report (reporting a malfunction). The reporter fills out a new report and submits it by clicking the submit button. The reportData parameter for the submitReport function corresponds to malfunctionDescription, email, notify_submitter, questionAnswers, questions, showOtherTextInput, uploadedFile. Upon creation of the report, the answers to open questions get created and stored in the database in a separate table. The answerData parameter to the createAnswers function corresponds to questions, questionAnswers, showOtherTextInput. If the reporter uploaded files, they also get created and stored in the database in a separate table. The report returns the creation status: successful or unsuccessful which gets displayed to the reporter as a popup on the report page UI.

Maintenance Report Editing Interaction Diagram

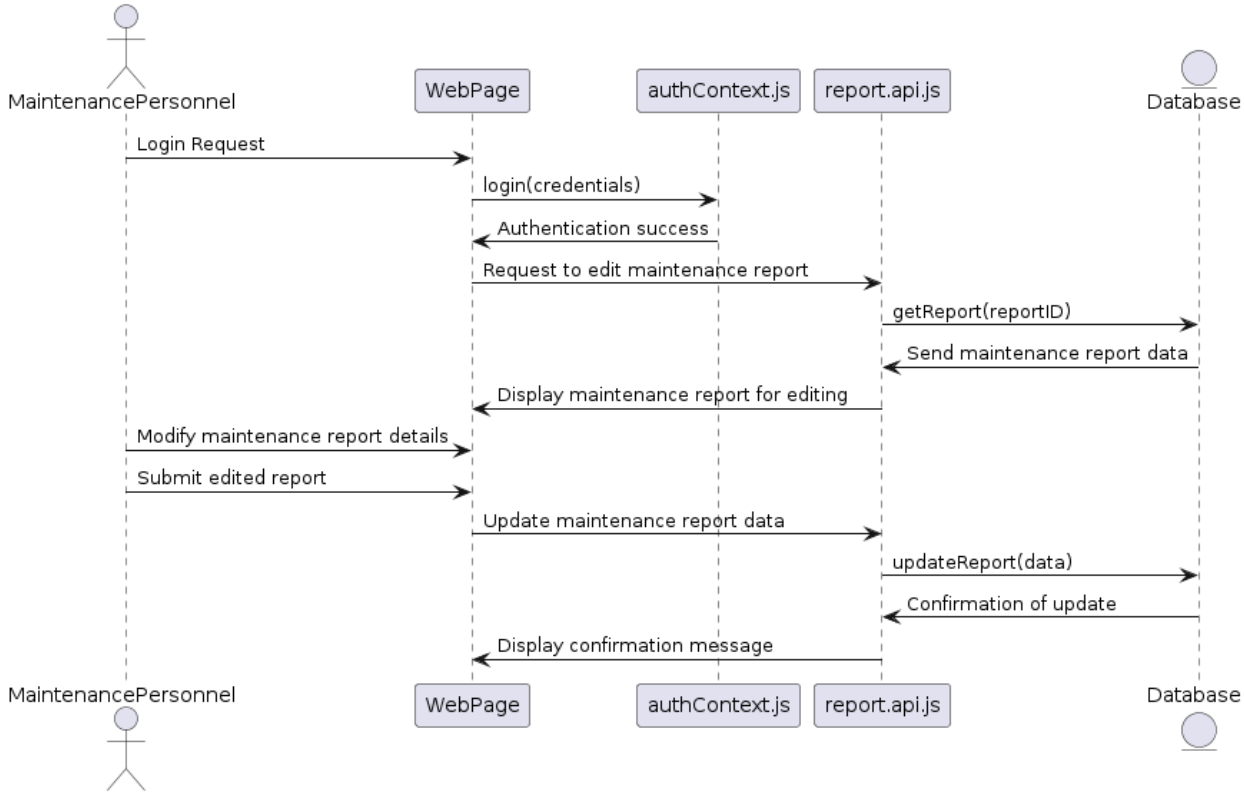

Figure 7. Interaction diagram for a maintenance personnel editing a report.

The diagram above is a sequence diagram of a member of the maintenance personnel editing a report. The maintenance staff member needs to log in via the web page, immediately afterwards they can access the reports, selecting one they will get all the data for the report through the function getReport(reportID) then they can edit the status or the priority and afterwards submit the edit through the function updateReport(data) . Once the status of a report has been changed, a confirmation message will be displayed.

Progress

Who	When (2024)	Where	What
Demian	26 feb	Database	Database Design (subject to change)
Demian	4 march	Design document	Updated database design diagram
Leo	10 March	Technology	Languages and frameworks
Callista	11 March	Execution View	Malfunction report system sequence diagram and description
Alessio	11 March	Execution View	Maintenance report editing sequence diagram
Maria	11 March	Architecture Introduction	Conceptual view diagram Wrote paragraph
Leo	13 March	API	API endpoints and schema
Leo	14 March	Module view	Module view
Demian	14 March	Database	Updated database design
Maria	14 March	Introduction	new introduction
Maria	14 March	Architecture	text for the conceptual view diagram
Callista	15 March	Execution View	Updated Malfunction report system sequence diagram
Leo	15 March	API	Api redesign
Alessio	25 March	Maintenance report editing sequence diagram	Fixed the text for Maintenance report editing sequence diagram
Callista	25 March	Malfunction report system sequence diagram	Edited text according to feedback. Changed style of diagram so it matches the *maintenance report editing* sequence diagram.
Demian	27 March	Database	Updated diagram and added description.
Leo	27 March	API	API descriptions

Maria	27 March	Architecture	Changed the conceptual view diagram to match the feedback. Also made changes to the text accompanying it.
Alessio	28 March	Sitemap	Added sitemap
Maria	29 March	Introduction	Changed the text
Leo	29 March	Module view	Text and diagram update
Leo	12 May	Database API	Updated database design Added API text
Leo	24 May	API	Added API requirements and personas
Leo	28 May	Security	Added security section
Leo	1 June	Api Security Module view Laravel packages	Added the login, logout and user endpoints in API; Updated security text; Updated module view and laravel packages section
Callista	1 June	Execution view - Malfunction Report System Interaction Diagram	Updated diagram and text
Callista	1 June	API	Added /files endpoints and descriptions
Alessio	1 June	Sitemap	Updated sitemap
Alessio	2 June	Execution view Maintenance Report Editing Interaction Diagram	Updated diagram
Maria	2 June	Conceptual view diagram	Updated diagram
Leo	2 June	All	Finishing touches