Quick Report - A web application that allows users to report events

Petre-Bogdan Serban

ABSTRACT

Bureaucracy is starting to be a bigger problem in this era and people have a need to make these processes quicker. Quick Report is a web application that allows citizens to report new events from their community. Users have to authenticate in the platform and then, they can report new events for state authorities. After the event is concluded, state authorities' representatives can change the status of the event and users can see what happened to that event.

Introduction

In Romania is a constant need for upgrading and creating web platforms to ease the communication between the citizens and the state authorities. The work that has been done in the last years showed that this will help with the bureaucracy and can lead to smaller queues and to a better communication. An automated platform where people can upload their new discovered events can lead to quicker fixes of those and it is a great tool for state authorities to find out their priorities.

Tech Stack

For creating this project, the MERN tech stack will be used in the developing phase:

- ReactJs - front-end
- NodeJs and Express - back-end
- MongoDb – database
- Google Analytics
- SeleniumAPI – testing
- FireBase - deployment
- GitLab – code base and CI/CD

These technologies offer great tools to develop web application based on http requests.

ReactJS is a JavaScript toolkit for creating reusable user interface components that is declarative, fast, and versatile. It's an open-source, component-based front-end library that's just responsible for the application's view layer. Taking security into consideration, the application will be using JWT tokens for the authentication process. Also, the ReactJs application will be based on axios requests to follow ajax development guidelines.

Node.js is a JavaScript run-time environment for servers. Node.js' standard library includes a set of asynchronous I/O primitives that prevent JavaScript code from blocking, and libraries in Node.js are often created following non-blocking paradigms, making blocking behavior the exception rather than the rule. It makes a great pairing for our React Frontend.

MongoDB is a NoSQL database management system that is free and open source. MongoDB is a database that allows you to organize, store, and retrieve document-oriented data. The documents resemble JavaScript Object Notation, although they employ a binary JSON variation (BSON). The choice for MongoDB stands in the ease of integration within our application (MERN Stack) and the fact that it could be easily used in a cloud deployment context by employing the help of Atlas, a fully-managed cloud database that handles all the complexity of deploying, managing, and healing your deployments.

Firebase is a set of hosting services for any type of application. It offers NoSQL and real-time hosting of databases and other content, and it suits well this project since we can make use of some free-to-use features for deploying our application.

Selenium is an umbrella project encapsulating a variety of tools and libraries enabling web browser automation, and we will be using the python framework of Selenium for creating a suite of automated test to cover the uses of the application.

Once a web application is developed and tested the next phase in its lifecycle is monitoring and reporting for keeping track of what is working and what not. For this purpose, we chose to integrate **Google Analytics** into the project for an easy but feature packed solution.

All code base will be versioned using **GitLab**. Also, this choice targets the robust CI/CD pipeline integrations that are provided by GitLab. This CI/CD will contain building the project, testing automatically and deployment.

Design and Preparation

This project started with creating the design for the pages of the application and creating a plan for development. The application targets citizens of a country and employees of different state authorities that can change the status of an event. For this, two different types of users are used throughout the application. The design of the application consists of 3 pages (first page, dashboard page, and insert event page). First page needs to have two buttons, one for login and one for register, and each one will redirect the user to the specific action. Dashboard page needs to show the user events inserted by him, if the user is a citizen one, and the events corresponding to the state authority if the user is a state authority. Also, for the citizen user, a button is needed to redirect to the form that is used to create a new event. Insert event page consists of a minimalist form in which a title, a description and a category of the event is required.

Implementation

Before the development of the application, the design should be considered. As for this reason, as a first step, an user flow will be determined so it could be followed into the development phase. Also, in terms of UI design, Proto.io will be used for the mock-up of the constituting pages.

In terms of architecture and implementation decisions, the project followed some key points of reference:

- AJAX approach - with use of axios library
- Minimal and Simple design with user focus
- Test driven development (TDD) - with use of SeleniumAPI
- Data Security - authentication based on JWT tokens
- App monitoring through Google Analytics
- CI/CD pipeline integration
- Firebase deployment

Authentication

The Authentication process is straight-forward as the industry dictates and takes into consideration the security of the user. The password is encrypted using bcrypt and the whole process is based on the standard JWT that defines a compact and self-contained way for securely transmitting information between parties, the user receives a unique token for the current session, everything being incorporated into the Mongo database.

Insert Event Form

For the insert event form, it was used a form from React Bootstrap that creates a simple and reliable form for the users to report an event. The data is sent to backend on a post request from the backend that will create a new entry in the table Events. A relation with a user is required for an event.

Dashboard

The dashboard page looks different based on the type of the user logged into the application. If a user is a citizen, it will display the events created by him, with details about the event and the status. Frontend will make a HTTP request to the backend and this one will provide the frontend with the events. Also, a button is present that redirects the user to the insert event form. For a state authority user, the dashboard shows the events assigned to that authority. In database, each event has a category(salubritate, rutier, ordine, administrativ). To check for authority, the authority from the user's email is evaluated(e.g. stefan@administrativ.ro -> administrativ is the category). JSON objects are used for communication between frontend and backend.