The project “Pay and keep track of bills from one platform”

The project “Pay and keep track of bills from one platform” is developed by Marija Gjorgjievska under the mentorship of Boban Joksimoski. The source code of the project and all the information about starting the project can be found on this repository.

Abstract

The idea for the project “Pay and keep track of bills from one platform” is to pay the bills and keep track of the bill from one platform, the title say it all. The idea behind this project is simple but helps a great deal.

This is the first version of this application and because of that has some setbacks. One of them is that the application is only for Gmail users and the other one is that paying of the bills is not possible at the moment.

The application is interactive application with font end and back end. The technologies that we use for the back end are: Docker PostgreSQL Environment and GraphQL Server on the other hand for the front end we use: React with Relay management library which fetch and update data with GraphQL . The importance of these technologies for this project is described in the body of this project report.

The combination of technologies used in this application is very common in communications platforms and large-scale web application but that does not exclude it for small applications. There are benefits by using this technologies in small applications also, more detail information about the benefits can be found in Evaluation and Conclusion.

There is room for improvement and growth of the application. Together with users’ feedback and new coming ideas this platform can become essential using platform for every household.

Introduction

Today you can pay the bills from the comfort of your home, but you must go to different platforms to pay bills for different utility services, or to repeat the same action on one platform for two different bills. The web application “Pay and keep track of bills from one platform” gives you the comfort to log in on one platform and have the information about your bills and the possibility in the future to pay them all in one place.

There are two requirements for a user to be able to use the application. The application now works only with Gmail accounts, so the first requirement is to have a Gmail account. In the future we hope to extend it to other emails too. The second requirement for this application is to receive your bills by email. Today for every utility service provided for one household there is

an option to receive your bills by email, which is eco-friendly, so let's help the planet and enroll to receive the bills on email.

Owing to the support given by Google we have Gmail API which gives us the option to read and analyze the e-mail sent by companies who provide service utility for household. When you enroll for our application, you also give us permission to read the emails and select the necessary data for the application. Subsequent to enrollment for the application you will be able to see all of the bill that you will receive from that moment on. An additional feature that this application holds is the opportunity to see how much of your income is spent on bills and statistics for the bills amount through the years since you are using the application. Due to the requirements for the feature to pay the bills we were not able to implement that feature completely, there is only an illustration of how it would work.

Implementation

This application consists of back-end and front-end parts.

Back-End implementation

PostgreSQL is used as database for the application which is created with Docker. PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

For the back end we are using GraphQL server. A GraphQL server is the interface between the back end and the application front-end. It parses queries written in GraphQL and fetches the data from connected PostgreSQL databases.

Front-End implementation

The front-end is created in React. React is a free and open-source front-end JavaScript library for building user interfaces based on components. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js. Because React only concerned with the user interface and rendering components to the DOM, React applications often rely on libraries for routing and other client-side functionality.

Updating and fetching of the data is done with GraphQL. GraphQL is a language for querying and modifying data on servers. The unique thing about GraphQL is that rather than having a fixed set of API endpoints, the server provides a palette of options that we can use to request any combination of data that it may need. This allows us as front-end developers to move more quickly because there is no need to write and deploy new endpoints as data requirements change. It also means that when a new version is released, it can request just the data it needs, without extra fields leftover for compatibility with older versions.

We use Relay data management library for React that lets you fetch and update data with GraphQL. Relay brings the composability of React components to data fetching. Each

component declares its own data needs, and Relay combines them into efficient pre-loadable queries. Every aspect of its design is to make the natural way of writing components also the most performant.

Evaluation and Conclusion

The connection of the database to the front-end is done by GraphQl and with the help of data management library Relay. Relay is designed for high performance at any scale. Relay keeps management of data-fetching easy, whether your app has tens, hundreds, or thousands of components. And thanks to Relay's incremental compiler, it keeps your iteration speed fast even as your app grows.

Relay is a critical infrastructure in Facebook, there are tens of thousands of components using it. Relay was built in tandem with GraphQL. Relay and GraphQl are great for Facebook which means they are great for applications on a big scale. Relay requires a bit more up-front setup and tools, in favor of supporting an architecture of isolated components which can scale with your team and app complexity. This is the reason why some will not support the use of Relay and Graphql on small applications like this one, but once you learn the principles of relay, you will have more time working on business logic instead of pipelining data. "Pay and keep track of bills from one platform" is small application but because of relay the data is efficiently fetched in a single GraphQL request, has data consistency in every moment, ensure project-wide consistency and correctness against GraphQL schema, optimized runtime and moves error detection to dev-time. If you're the sort of team that believes in using Flow and having all the features above the Relay and GraphQl are great combination for you application, no matter the size of the application. With Relay as your application grows you won't accidentally break other components.

PostgreSQL is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications.

Running PostgreSQL in Docker containers is a popular choice for developers and administrators due to its flexibility, scalability, ease of deployment and switch between different versions of PostgreSQL. Only in three steps you can set up your Docker PostgreSQL Environment, the only requirement is installation of Docker on your computer. Docker PostgreSQL Environment was good choice for this application because of the easy set up and for possible growth of the application.

Future Work of the application

The technology chosen for the application is great for the current state of the application and for future changes, improvements and growth of the application. Hopefully the community for all of technology will grow and support them.

Right now, the application can be used only by Gmail users but we hope to expand to other secured email.

In the terms of business logic, the application can grow with feedback and request by users. While developing the application with Relay and GraphQl I learned a lot about these technologies. I have knowledge of the possibilities the application is able to grow due to the technologies used and it is endless. An idea for future feature implementation came to mind while realization of the project. Because of the wide use of Relay and GraphQl in communication platforms I got an idea that this application can be a communication platform also. Users who will approve to be part of the combination platform will share their information about the amount they spent on household utilities based on the quality and quantity of the service they are getting. The users in communication on the platform based on the data available can share tips and tricks on saving money.

Bibliography

- https://relay.dev/
- https://graphql.org/
- https://react.dev/
- https://developers.google.com/gmail/api/quickstart/js
- https://blog.logrocket.com/guide-adding-google-login-react-app/#installing-required-package-add-google-login-react-app
- https://github.com/tobias-tengler/create-relay-app