Задача от TelebidPro

Кандидатстване за работа

Лоренцо Антонио Екели

24 December 2023

Технологии

За Frontend частта на задачата съм използвал HTML, CSS и JavaScript. Избрах да използвам тях, защото целият уеб работи с тези три технологии. С техните доработки(framework) можеш да пишеш всякакви апликации.

За Backend съм използвал Node.js. Да можеш да пишеш на JS и за Frontend и Backend е много голямо предимство. Това е добре за по-бързо разработване и лесно поддържане на софтуера. Понеже той е управляван от събития и е с неблокиращ I/O модел, това му позволява голям брой едновременни връзки.

За бази данни съм ползвал MySql. Това е една от на-популярните бази данни. Понеже е релационна база данни това позволява високи скорости на заявките и сигурността им.

Функционалности

От функционалностите зададени ми по условие съм изпълнил повечето функции от бакенда.

Login -Той приема обектите на заявката и отговора като вход. Когато се направи POST заявка към този маршрут, тя ще анализира тялото на заявката, за да получи полетата за имейл и парола.

След това използва bcrypt, за да хешира изпратената парола и прави заявка в базата данни, за да намери потребител с изпратения имейл.

Ако бъде намерен потребител, той сравнява хешираната изпратена парола със съхранената хеш парола на потребителя. Ако те съвпадат, той изпраща 200 отговор за успех, показващ, че влизането е било успешно. Ако те не съвпадат, той изпраща 401 неоторизиран отговор.

Ако не бъде намерен потребител с изпратения имейл, той също изпраща 401 неоторизиран отговор.

Целта е да се обработват валидирането на идентификационните данни за влизане на потребителя в бекенда чрез проверка на паролата спрямо хешираната парола, съхранена за този потребител в базата данни. Той използва bcrypt за сигурно хеширане и солиране на пароли преди сравнение, за да защити срещу атаки с груба сила.

Register- Той приема POST заявка като вход, съдържащ имейла, паролата, собственото и фамилното име на потребителя в тялото на заявката.

Той ще върне отговор 200, ако регистрацията е успешна, или отговор 400 със съобщение за грешка, ако има проблем с регистрацията.Първо проверява дали имейлът вече съществува, преди да създаде нов потребител, за да избегне дублиране на имейли.

Паролата се хешира с помощта на bcrypt, преди да се запази за сигурност

Той връща подходящи отговори за грешка, ако съществува имейл или запис на грешка, за да се справи с грешките

Кодът използва обещания за обработка на асинхронни действия като хеширане и записване в DB

Delete-Връща отговор 200 при успех или грешка 405, ако заявката не е DELETE. Когато се получи заявка за ИЗТРИВАНЕ, кодът анализира тялото на заявката, за да получи потребителския идентификатор. След това извиква метода User.deleteById, предавайки идентификатора, за да изтрие този потребител от базата данни.

Update-Връща отговор 200 при успех или 405, ако заявката не успее. Проверява дали има такъв потребител, ако има пренаписва старите данни, ако няма връща грешка.

Понеже не съм ползвал sequlize, а съм ползвал mysql2, защото sequlize е framework. Първо създадох pool за базата данни. След това създаваш клас за моделите(който е еднакъв с таблицата в базата) и използваш Sql функции. Там създадох и статични методи, което доста улеснява операциите, защото няма нужда от допълнително създаване на обекти.

Изпращането на мейл съм го осъществил с nodemailer. Там идеята за проверяване дали мейлът е истински е с генериране на случайно число и да го праща. Има нова таблица в базата данни, която запомня числото и на кой потребител е, и след известно време го трие(не е имплементирано). Проверява се дали потребителят е получил съобщението чрез числото.

Файловете

В папка utils има два файла database.js е за интеграцията с базата данни, а userUtil.js е за логиката за пращане на мейли.

В папка unit съм сложил файла userTest.js той е за тестване на функционалностите.

В папка routes е логиката за CRUD операциите(това трябва да е controller в express.js).

В папка models са моделите.