**Документация проект №26**

1. **ТЕМА:** „taxiZilla”
2. **АВТОРИ:**

Делян Любомиров Бойчев; 0550071400; ул. „3-ти март“ 12 , град Лясковец, 0889380037, ПМГ „Васил Друмев“, 9в клас

Мартин Георгиев Маринов; 0244151402; ул. „България“ 17 , град Велико Търново, 0882740231, ПМГ „Васил Друмев“, 12в клас

1. **РЪКОВОДИТЕЛ:**

Георги Петров Игнатов, 0889255850, [g\_ignatov@mail.bg](mailto:g_ignatov@mail.bg), Старши учител по информатика и информационни технологии

1. **РЕЗЮМЕ:**

**4.1. Цели:**

TaxiZilla има за цел да улесни работата на таксиметровите шофьори, на техните фирми и на потребителите, които ползват услугите им. Уеб приложението е гъвкаво и дава възможности за извършване на операции като поръчка на такси, поръчка за пазаруване и доставка до дома с няколко клика. Това може да намали разходите на фирмата за диспечери и да улесни потребителите при поръчване на такси. Това приложение е единно за територията на цялата страна и не е нужно търсенето на телефонни номера на таксиметрови шофьори или фирми при пътуване в друг град.

**4.2. Основни етапи в реализирането на проекта:**

1. Проучване на технологии и избор на подходящите

2. Реализация на Back End чрез използване на NestJS, MariaDB, Node.js, TypeORM и Passport

3. Реализация на Front End чрез използване на HTML, CSS, JavaScript и Bootstrap

4. Поправка на грешки

5. Създаване на мобилно приложение за шофьори чрез използване на Flutter

6. Поправяне на грешки

7. Тестване на продукта

**4.3. Ниво на сложност на проекта:**

**4.4. Логическо и функционално описание на решението:**

**Frontend**

Модул js

adminPanel.js – В този модул се намират всички функции, които извършват операции в Admin панела.

changePages.js – В този модул се намират всички функции, които извършват смяната на страниците и правят приложението SPA.

design.js – В този модул се намират всички функции, които извършват промени в дизайна без redirect.

keyGenerator.js – В този модул се намира функция, която генерира ключ за верификация дали потребителят е преминал през сайта или е използвал някое приложение за заявка.

main.js – В този модул се намира кода, който се изпълнява при отваряне на приложението.

postAndGet.js – В този модул се намират функциите за custom POST и GET заявки.

userOperations.js – В този модул се намират функциите, които извършват всички операции с потребителите.

Модул pages

404.html – Страница при грешка 404.

adminPanel.html – Страница за Admin панел.

homePage.html – Начална страница.

index.html – Страница, която се зарежда при стартиране на приложението.

loginFirmPage.html – Страница за влизане като фирма.

loginPage.html – Страница за влизане като потребител.

makeOrderPage.html – Страница за правене на поръчка.

profileFirmPage.html – Страница за профил на фирма.

profilePage.html – Страница за профил на потребител.

registerFirmPage.html – Страница за регистриране като фирма.

registerPageUser.html – Страница за регистриране като потребител.

verifiedFalse.html – Страница за даване на информация, че потребителят вече е потвърдил профила си.

verifiedTrue.html – Страница за даване на информация, че потребителят е потвърдил профила си успешно.

Модул adminPanelTabs

addDriverTab.html – Страница за добавяне на таксиметрови шофьори от Admin панел.

firmRemoveTab.html – Страница за премахване на фирми от Admin панел.

moderationVerifyFirmTab.html – Страница за одобряване на фирми от Admin панел.

userActivateTab.html – Страница за потвърждаване профила на потребители от Admin панел.

userChangeRoleTab.html – Страница за смяна роля на потребители от Admin панел.

userEditTab.html – Страница за редактиране на профили на потребители от Admin панел.

userRemoveTab.html – Страница за премахване на потребители от Admin панел.

**Backend**

Backend-а на приложението се дели на 3 нива: Модули, които комуникират с микроуслугите и другите модули, Микроуслуги които извършват операции и комуникират с репозиторитата и репозиторита, които извършват операции с базата данни.

Съставни части на Backend-а:

AppModule – Служи за инициализация на другите модули

AuthModule – Модул за автентикация, оторизация и операции за потребители и шофьори

AuthController – Контролер, който слуша за заявки за операции с потребители и шофьори. При получаване на заявка предава данните на AuthService.

AuthService – Една от микроуслугите на приложението, която изпълнява операции с потребители и шофьори. Комуникира с UserRepository.

UserRepository – Използва се за извършване на операциите за потребители и шофьори в базата данни.

UserEntity – Съдържа информация за структурата на таблицата User в базата данни, което оказва на приложението как да я създаде

JWTStrategy – Описва валидацията на JWT Token

JWTPayload – Описва обекта, който се съхранява в JWT Token

TaxiDriver Class – клас в който се извършва взимане на позициите на шофьори, изчисляване на разстояние до клиента и изпращане на заявки до шофьорите за поръчка.

RegisterUserDTO – Data transfer object за валидация на данните при регистрация на потребител и посредник между контролера и микроуслогата за въведените от потребителя данни.

UserRoles Enum – Описва възможните роли на потребителите

UserStatuses Enum – Описва възможните статуси на потребителите

FirmModule - Модул за автентикация, оторизация и операции за фирми

FirmController – Контролер, който слуша за заявки за операции с фирми и предава информацията на FirmService

FirmService – Микроуслуга за извършване на операции с фирми. Комуникира с FirmRepository и SupportedCityRepository.

FirmRepository – Използва се за извършване на операциите с фирми в базата данни

SupportedCityRepository – Използва се за извършване на операции с поддържаните от фирмите градове в базата данни.

RegisterFirmDTO – Data transfer object за валидация на данните при регистрация на фирма и посредник между контролера и микроуслогата за въведените от потребителя данни.

JWTStrategy – Описва валидацията на JWT Token

JWTPayload – Описва обекта, който се съхранява в JWT Token

SuppotedCity Entity - Съдържа информация за структурата на таблицата SupportedCity в базата данни, което оказва на приложението как да я създаде

Firm Entity - Съдържа информация за структурата на таблицата Firm в базата данни, което оказва на приложението как да я създаде

OrderModule – Модул за операции с поръчки

OrderController – Контролер, който слуша за заявки за операции с поръчки. При получаване на информация предава информацията на OrderService.

OrderService - Микроуслуга за извършване на операции с поръчки. Комуникира с OrderRepository.

OrderRepository – Използва се за извършване на операции с поръчки в базата данни.

OrderStatus Enum – Описва възможните статуси на поръчка

Order Entity - Съдържа информация за структурата на таблицата Order в базата данни, което оказва на приложението как да я създаде

AppModule – Обединяващ модул

TypeOrmModule – Модул за операции с база данни

SessionModule – Модул за операции със сесии

JWTModule – Модул за JWT Token-и

PassportModule – Модул за оторизация

Модулите, микроуслугите и репозиторитата си комуникират чрез Dependency injection, което е механизъм на платформата NestJS.

Освен тези неща има няколко помощни файла.

coordsAndStatus.array.ts – Декларация на масиви, които се използват в приложенитео

email.transport.ts – Съдържа данни за SMTP сървъра, чрез който се изпращат имейли за потвърждение на акаунта.

http-exception.filter.ts – Филтрира изключенията. Вместо да изпрати на браузъра изключение, което ще излезе в конзолата, то изпраща просто response string “401” което е HTTP кодът съответстващ на Unauthorized.

**4.5. Реализация:**

Използвани технологии:

TypeORM

NestJs

Passport

MariaDB

Node.js

TypeScript

Flutter

HTML

CSS

Bootstrap

JavaScript

Използван софтуер:

Gimp

LibreOffice

VS Code

GitHub

Git

Windows 10

Arch Linux

Node.js

MariaDB

Flutter

Използвани ресурси:

Снимки с лиценз Creative Common и npm пакети.