Документация на уеб приложението „We-tried“

Съдържание

I. Документация към клиента, доставчика и администратора

1. Обща информация за приложението

We-tried е модерно уеб приложение, което ти дава възможност само чрез няколко клика да поръчаш храна онлайн от любимият си ресторант. Не изисква никакво инсталиране на приложение, като неговото достъпване става директно чрез браузъра на вашето устройство, независимо дали ще е настолен компютър, лаптоп, таблет или мобилен телефон. Подходящо за всякакви офис поръчки, групови поръчки и спонтанни хапвания. Разполагаме с:

- Лесен преглед на менютата на ресторантите, с реални снимки, ясни описания и възможност за персонализация на поръчката;
- Сигурност при поръчване;
- Леснодостъпен процес от избора до плащането, без излишни стъпки;
- Проследяване статуса на поръчката в реално време.

2. Стартиране

За да бъде стартирано приложението, трябва да се отвори браузър и да бъде написано или името на нашият сайт, или нашият домейн. При отваряне на уеб приложението се зарежа страница с вход или регистрация.

3. Регистрация и вход в приложението

В страница с вход или регистрация, трябва да влезете в вашият профил, ако нямате такъм си създавате чрез регистрация. Ако сте нов доставчик, администратора допълнително ще Ви даде роля като такъв.

II. Документация за разработчици

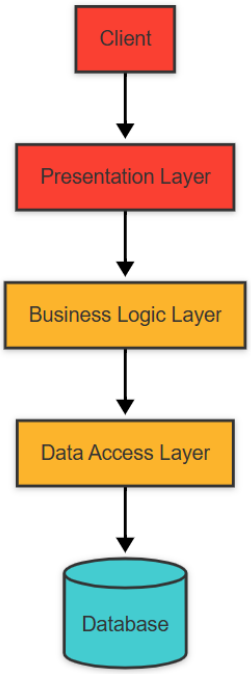
1. Бизнес план

Бизнес планът е документ, характеризиращ пазара, конкуренцията, маркетинга, идеята, организацията, оперативната дейност и финансите на един бизнес. Той изпълнява функцията на пътна карта за стартиране и управление на даден бизнес.

- **Резюме -** онагледява сбито проекта „We-tried“ за уеб платформа за доставка на храна;
- **Анализ на средата -** информира за вътрешните и външните фактори, които окозват влияние върху пазара, като инфлация, социални навици, силна конкуренция от други платформи като Glovo и Foodpanda и др.;
- **Анализ на пазара –** огражда целевата аудитория, в случая хора между 18-45 г., основно в големите градове. Както и показва предимствата на приложението като удобство, спестено време и разнообразна храна;

- **Анализ на конкуренцията –** съпоставя „We-tried“ с утвърдените конкуренти;
- **Маркетингов план –** маркетингът ще се състои от реклама, флаери, социални мрежи и инфлуенсърски партньорства;
- **План за управление –** компанията ще бъде ООД с двама основатели е екип от директори, както и допълнително външна помощ за рекламите, правни и HR услуги;
- **Оперативен план –** дейността включва разработка на уеб приложението, интеграция с ресторанти и логистика чрез собствени куриери;
- **Финансов план –** началната инвестиция е 70 000 лв. като се очаква печалба още през първата година.

2. Описание на монолитната архитектурна диаграма



Диаграмата представя класически клиент-сървър модел с трислоен монолитен архитектурен дизайн, типичен за традиционни уеб приложения. Той се състои от:

- **Клиент (Client)** – потребителският интерфейс (в случая браузър), състоящ се от Presentation Layer (PL), които отговаря за комуникацията с клиента;
- **Сървърна част (Server)** – включва два основни слоя, а именно Business Logic Layer (BLL), служещ за обработка на бизнес правилата и логиката, и Data Access Layer (DAL), който управлява взаимодействието с базата данни;
- **База данни (Database)** – централно хранилище за данни.

За създаването на архитектурната диаграма сме използвали mermaidchart.com

3. База данни – базата се състои от 8 таблици:

- ***food_order*** - централна таблица, представяща направена поръчка;
- ***order_item*** - представлява продукт в поръчка;
- ***dish*** - съдържа ястията на ресторантите;
- ***cart*** - съдържа продукти на потребител преди поръчка;
- ***user*** - потребители на системата;
- ***deliverer*** - доставчици;
- ***restaurant*** - ресторантите;
- ***admin*** - администратори на системата.

4. Front-end

Приложението предлага разнообразни функционалности за различни роли (администратори, собственици на ресторанти, доставчици и потребители).

- ***add-dish*** **-** реализира формата за добавяне на ново ястие към ресторант в приложението, достъпна за администратори или собственици на ресторанти, които имат право да управляват менюто;
- ***add-restaurant*** - форма за добавяне на нов ресторант в приложението, достъпна за администратори и собственици, които имат право да създават ресторанти в системата;
- ***all-ordersl*** – страница за всички налични поръчки, достъпна само за доставчиците, оттук доставчиците преглеждат свободни поръчки и да си ги добавят в списък с доставки;
- ***deliverer-orders*** – интерфейс за доставчици в приложението, дава им възможността да преглеждат и избират поръчки, които да доставят;
- ***home*** – основна страница след влизане в профил, дава функционалности като търсене на ресторанти, визуализиране на ресторанти по категории, както и административни опции за добавяне, редактиране и изтриване;
- ***index*** – начална страница на приложението, предназначена да приветства потребителите и да им предостави възможността да влязат в системата или да се регистрират. Улеснява потребителите още с влизането в приложението с визуалната си организация;
- ***internal-server-error*** - използва шаблон за показване на съобщение за грешка, като информира потребителя, че е възникнал проблем и че екипът от разработчици работи по отстраняването му;
- ***login*** - шаблон представлява страницата за вход ***/Log In/*** в уеб приложението. Потребителите въвеждат своите идентификационни данни, за да получат достъп до системата.;
- ***not-found*** - представлява страницата за грешка ***404 - Not Found***, която се показва, когато потребителят се опита да достъпи несъществуваща страница в уеб приложението.;
- ***profile-delivery*** – същината на страницата е да покаже профила на доставчика в приложението. Той съдържа както личната информация на доставчика, така и списък с вече доставени поръчки;

- ***profile-user*** - този шаблон ни дава профила на потребителя в приложението. Потребителят може да достъпи своите основни данни, като потребителско име, имейл и парола (в скрит вид);
- ***register*** - форма за регистрация на нов потребител в приложението като трябва да въведе потребителско име, имейл, парола и да потвърди паролата си;
- ***restaurant*** **-** използва се за представяне на информация за конкретен ресторант, включително неговите ястия, с възможности за добавяне, актуализиране и изтриване на ястия (само за администратори). Също така включва търсене, линкове за профила на потребителя и достъп до кошницата.;
- ***revenue*** - показва статистика за приходите на платформата. Съдържа функционалности за филтриране на приходите по ден, седмица, месец или година, като показва информация за приходите в избрания период;
- ***shopping-cart*** - представя страница за пазарската кошница на потребителя, където той може да види избраните ястия, да променя количествата и да завърши поръчката, както и показва форма за доставка и избор на метод на плащане;
- ***update-dish*** - форма за обновяване на информацията за ястие в ресторантската система, включително име, описание, цена, тип и изображение. Достъпна за администратори и ресторантски оператори;
- ***update-restaurant*** - шаблон предоставящ форма за актуализация на информацията за ресторанта, като име, адрес, телефонен номер, работно време, тип и изображение. Достъпна за администратори и ресторантски оператори.

5. Back-end

Структурата на проекта съдържа няколко модула и функционалности:

а. Admin - служи за управление на администратори. Негова функционалност е възможността да трие, добавя и променя ресторанти и меню, като при операциите add и edit се достъпва страница съдържаща информация за променената такава;

б. Cart - изпълнява функцията за управление на количката на потребителя. Съдържа функционалности като преглед на количката, добавяне на продукт, начин на плащане и промяна на количеството;

в. Config - съдържа конфигурационни класове за Spring;

- Файл ***BeanConfiguration*** предоставя важна функционалност за сигурност, а именно хеширане на потребителските пароли;

- ***WebMvcConfiguration*** ни дава функционалност създаваща място, където се дефинират всички настройки, свързани със сигурност и обработка на HTTP заявки и responses.

г. Deliverer - отговаря за логиката свързана с доставчиците – тяхното създаване, управление и заявки към тях. Предоставя функционалности като подробен преглед на профил, извеждане на списък с всички поръчки в системата и такива свързани с конкретен доставчик;

- В ***DelivererService*** се добавя още една функционалност свързана с бонуси. Тя гласи, че при достигане на определена доставена сума, му се добавят 300 лв. бонус.

д. Dish - състои се от функционалности за създаване, редактиране и категоризация на ястия;

- Чрез ***CreateDishRequest*** се създават нови ястия;
- ***DishController*** е контролер, който обработва операциите, свързани със създаване и управление на ястия в ресторантите;
- ***UpdateDishRequest*** служи за обновяване на информацията за ястията;
- ***DishService*** отговаря за операции свързани с ястията като създаване, обновяване, изтриване и качване на изображиния.

е. Index - съдържа началния контролер на приложението, който обработва заявките към началната страница на приложението;

ж. Order - отговаря за създаване, управление и проследяване на поръчки;

- ***OrderController*** е контролер, който ни дава основа за управление на поръчки;
- ***OrderService*** предоставя функционалност за работа с поръчки в приложението.

з. Register /регистрация/;

- Файлът ***RegisterRequest*** представлява модел на заявка за регистрация на потребител. Той се използва за събиране и валидиране на входните данни от потребителя при създаване на нов акаунт. Класът включва полетата потребителско име, имейл адрес, парола и потвърждение на паролата;
- Контролерът ***RegisterController*** обработва HTTP заявки, свързани с регистрацията на нови потребители. Той работи с *RegisterRequest* обекти, валидира ги и при успех използва UserService, за да създаде нов потребител в системата.

и. Log in /вход/;

- Файлът ***LoginRequest*** представлява модел на заявка за вход (логване) в системата. Използва се за валидиране на потребителските данни, въведени във формата за вход – потребителско име и парола;
- Контролерът ***LoginController*** отговаря за обработката на заявките за вход в системата. Приема и валидира потребителски входни данни чрез *LoginRequest*, удостоверява потребителя чрез UserService и създава сесия.

й. Restaurant - изпълнява функцията за управление на ресторантите. Съдържа функционалности като създаване и актуализиране на информацията за ресторанти, както и за обработка на HTTP заявки**;**

к. Revenue - служи като функционалност за обработка на логиката за изчисляване на приходите на ресторантите. До тази функционалност има достъп само администраторът**;**

л. Security - предоставя необходимата информация за процеса на удостоверяване. Чрез Security се задават ролите на потребителите (admin/deliver/user);

м. User - отговаря за управлението на операциите, свързани с потребителския профил като подробен преглед на информацията за текущият логнат потребител и гарантира, че само логнатите потребителите имат достъп**;**

н. Application - това е стартовата точка на приложението и е като „мотор“, който стартира цялата система. Без този елемент приложението няма да работи, докато всички други функционалности надграждат основата.

6. Тестове

В проекта "We-tried" се отделя специално внимание на тестването на основната бизнес логика, за да сме сигурни, че приложението работи надеждно при различни условия. За тази цел използваме JUnit 5 за създаване на тестовете и Mockito за мокване на зависимостите, като така можем да фокусираме тестовете единствено върху функционалността на услугите (Service слоевете), без да се налага да използваме реална база данни.

а. Покрити модули

Понастоящем имаме разработени тестове основно за модула **DelivererService**. В него проверяваме различни аспекти на поведението на услугата, като обхващаме както очаквани (позитивни) сценарии, така и гранични и негативни случаи.

б. **Тестове за DelivererServic -** в DelivererService са създадени следните тестове:

- **Търсене на доставчик по ID** - проверяваме дали доставчикът се връща коректно, ако съществува в базата данни. При липса на такъв доставчик очакваме системата да хвърли правилно изключение (***IllegalArgumentException***);
- **Извличане на завършени поръчки за доставчик** - тестваме дали методът ***getCompletedOrders*** връща точно списъка с поръчки със статус "COMPLETED", свързани с конкретния доставчик;
- **Изчисляване на месечни приходи**;
  През ***calculateMonthlyRevenue*** проверяваме дали системата правилно изчислява общата стойност на всички завършени поръчки на даден доставчик за последния месец.
  Също така тестваме граничен случай, в който няма направени поръчки, и се очаква да се върне стойност 0.
- **Проверка за право на бонус**;
  Методът ***isEligibleForBonus*** определя дали даден доставчик има право на бонус. Тестовете проверяват случаи, когато доставчикът е над минималната сума за бонус (и трябва да получи бонус), и когато е под нея (и не трябва да получи).
- **Изчисляване на общата стойност на всички поръчки на доставчик**;
  Чрез ***calculateTotalOrdersValue*** валидираме дали приложението правилно събира стойностите на всички поръчки, които даден доставчик е изпълнил.
- **Извличане на всички поръчки в системата**;
  Методът ***getAllOrders*** се тества за да сме сигурни, че се връща коректен списък с всички направени поръчки, независимо от техния статус.
- **Извличане на всички поръчки на конкретен доставчик**.
  Чрез метода ***getDelivererOrders*** проверяваме дали се зареждат правилно всички поръчки, свързани с конкретен доставчик, без значение от техния статус.

в. **Използвани техники и инструменти -** за постигане на изолираност на тестовете и избягване на зависимости от външни системи:

- Използваме **Mockito** за мокване на DelivererRepository и OrderRepository.
- В тестовете с финансови операции се използват **BigDecimal** стойности за избягване на грешки при сумиране и закръгляне.
- За времевите зависимости, като периода за месечните приходи, използваме контролирано задаване на времената чрез LocalDateTime.

Всеки тест е структуриран така, че да покрива един конкретен случай, за да можем при грешка лесно да открием проблема.