**МојМајстор**

**Вовед**

Апликацијата **МојМајстор** е платформа која е дизајнирана да ви помогне лесно и брзо да пронајдете мајстори за различни услуги како автомеханичари, водоинсталатери, електричари, молери и многу други.

Во развојот на апликацијата се користат современи технологии за да се обезбеди сигурност, брзина и ефикасност во процесот на поврзување на корисниците со мајсторите:

* **PostgreSQL**: релациона база на податоци која се користи за зачувување и управување со сите информации за корисниците и мајсторите. PostgreSQL овозможува стабилно и сигурно управување со големи количини на податоци.
* **Spring Framework**: основа на серверската логика, Spring овозможува ефикасна и сигурна комуникација помеѓу фронтенд и базата на податоци. Се користи Spring Boot за брзо создавање на RESTful веб-сервиси кои ја поддржуваат апликацијата.
* **Angular**: фронтенд фрејмворк кој овозможува динамично, модерно и интерактивно корисничко искуство. Angular го управува корисничкиот интерфејс и комуникацијата со серверот преку API повици, овозможувајќи лесно пребарување и прикажување на мајстори.

Целта на апликацијата е да го поедностави процесот на пронаоѓање и вработување на мајстори за различни услуги, како и да им помогне на мајсторите да го зголемат својот бизнис преку платформата.

**Модели и Структура на Базата на Податоци**

Апликацијата **МојМајстор** користи релациона база на податоци со PostgreSQL за управување со информации за корисници, мајстори и рецензии. Во продолжение е преглед на основните модели што ја дефинираат структурата на базата на податоци:

**1. Worker (Мајстор)**

Табелата Worker ги претставува сите мајстори регистрирани на платформата. Оваа табела содржи информации за мајсторите, вклучувајќи опис на нивните услуги, адреса, општина, телефонски број, категорија и просечна оценка.

* id (Long) - Примарен клуч кој автоматски се генерира.
* description (String) - Опис на услугите кои ги нуди мајсторот.
* address (String) - Адреса на мајсторот.
* municipality (String) - Општина во која се наоѓа мајсторот.
* phoneNumber (String) - Телефонски број на мајсторот.
* category (Enum) - Категорија на мајсторот, која може да биде: **AVTOMEHANICHAR**, **VODOINSTALATER**, **ELEKTRICHAR**, или **MOLER**.
* averageRating (Long) - Просечна оценка што ја добил мајсторот од корисниците.

**2. User (Корисник)**

Табелата User ги претставува корисниците на платформата, вклучувајќи и нормални корисници и мајстори. Покрај основните информации како е-пошта и име, оваа табела содржи и референца кон Worker за оние корисници кои се мајстори.

* id (Long) - Примарен клуч кој автоматски се генерира.
* email (String) - Уникатна е-пошта на корисникот.
* fullName (String) - Полно име на корисникот.
* password (String) - Лозинка за пристап.
* role (Enum) - Улогата на корисникот: **NORMAL** за обични корисници или **MAJSTOR** за мајстори.
* worker (Worker) - Врска до објектот Worker, доколку корисникот е мајстор.

**3. Review (Рецензија)**

Табелата Review ги зачувува рецензиите што корисниците ги даваат за мајсторите. Секоја рецензија содржи информации за тоа кој ја оставил рецензијата, кој мајстор е оценет, коментар и дадената оцена.

* id (Long) - Примарен клуч кој автоматски се генерира.
* fromId (Long) - Идентификатор на корисникот кој ја оставил рецензијата.
* fromName (String) - Име на корисникот кој ја оставил рецензијата.
* toId (Long) - Идентификатор на мајсторот кој е оценет.
* comment (String) - Коментар од корисникот за мајсторот.
* rating (int) - Оценка што ја добил мајсторот (од 1 до 5).

**4. Rating (Оцена)**

Табелата Rating е посебна табела која зачувува индивидуални оценки што корисниците ги даваат на мајсторите. Оваа табела е поврзана со табелите Worker и User преку надворешни клучеви.

* id (Long) - Примарен клуч кој автоматски се генерира.
* rating (int) - Оцена (од 1 до 5) што ја дава корисникот.
* worker (Worker) - Врска до мајсторот што е оценет.
* user (User) - Врска до корисникот што ја дал оценката.

Овие модели заедно ја дефинираат структурата на базата на податоци на **МојМајстор**, овозможувајќи ефикасно управување со податоците за корисниците, мајсторите и рецензиите на платформата.

**Достапни API Ендпоинти**

Апликацијата МојМајстор користи RESTful API ендпоинти за комуникација помеѓу фронтендот и бекендот, овозможувајќи различни операции за управување со профили, рецензии, оценки, и автентикација на корисници. Во продолжение се опишани основните API ендпоинти:

**1. Профил Контролер (ProfileController)**

Овој контролер ги управува корисничките профили и нуди функционалности за преглед, пребарување и ажурирање на профилите.

**- GET /api/profile**

- Опис: Го враќа профилот на корисникот. Ако е дадено `id`, враќа профил на корисникот со тој ID. Ако не е, враќа профил на тековно најавениот корисник.

- Параметри:

- `id` (опционален) - ID на корисникот чии податоци треба да се вратат.

- `principal` - Тековно најавен корисник.

- Резултат: JSON одговор со профилот на корисникот.

**- GET /api/profile/search**

- Опис: Пребарува корисници врз основа на нивното име и враќа листа на мајстори.

- Параметри:

- `search` - Клучен збор за пребарување.

- `principal` (опционален) - Тековно најавен корисник.

- Резултат: Листа на корисници што се мајстори и кои одговараат на пребарувањето.

**- GET /api/profile/searchPageable**

- Опис: Пребарува корисници со поддршка за пагинација.

- Параметри:

- `page` (стандардно: 0) - Број на страница.

- `size` (стандардно: 10) - Големина на страница.

- `principal` (опционален) - Тековно најавен корисник.

- Резултат: Пагинирана листа на корисници.

**- POST /api/profile**

- Опис: Ажурирање на профилот на најавениот корисник.

- Параметри:

- `principal` - Тековно најавен корисник.

- `profile` - Податоци за ажурирање.

- Резултат: HTTP 200 ако профилот е успешно ажуриран.

**2. Рејтинг Контролер (RatingController)**

Овој контролер е одговорен за додавање на оцени за мајстори.

**- POST /api/ratings/rate**

- Опис: Додавање оцена за мајстор и ажурирање на просечната оцена на мајсторот.

- Параметри:

- `userId` - ID на корисникот што ја дава оценката.

- `workerId` - ID на мајсторот што се оценува.

- `rating` - Дадена оцена (од 1 до 5).

- Резултат: Порака за успех ако оцената е успешно додадена.

**3. Рецензија Контролер (ReviewController)**

Овој контролер е одговорен за додавање и прегледување рецензии за мајстори.

**- POST /api/review**

- Опис: Додавање рецензија за мајстор.

- Параметри:

- `principal` - Тековно најавен корисник.

- `review` - Податоци за рецензијата (ид на мајсторот, коментар, оцена).

- Резултат: Рецензијата е успешно додадена и враќа податоци за рецензентот.

**- GET /api/review**

- Опис: Преглед на сите рецензии за даден мајстор.

- Параметри:

- `id` - ID на мајсторот чии рецензии се пребаруваат.

- Резултат: Листа на рецензии за мајсторот.

**4. Автентикација (Authentication)**

Овие ендпоинти се користат за регистрација и најава на корисници.

**- POST /api/register**

- Опис: Регистрирање нов корисник.

- Параметри:

- `payload` - Информации за корисникот (е-пошта, име, лозинка).

- Резултат: Успешна регистрација и токен за автентикација, или порака за грешка ако корисникот веќе постои.

**- POST /api/login**

- Опис: Најавување на корисник.

- Параметри:

- `payload` - Е-пошта и лозинка за најава.

- Резултат: Успешна најава и токен за автентикација, или порака за грешка ако податоците се невалидни.

Овие API ендпоинти овозможуваат целосна функционалност за управување со кориснички профили, мајстори, оценки, рецензии и автентикација на корисници на МојМајстор.

**Безбедност и автентикација**

Процесот на автентикација во платформата „МојМајстор“ користи JWT (JSON Web Token) за безбедно автентикациско решение. Ова овозможува безбедна и ефикасна проверка на корисниците при пристап до API-то. Подолу е деталното објаснување на процесот:

**1. Регистрација на корисник**

Регистрацијата се обработува преку `AuthController` на апито`/api/auth/register`.

- Апи: `/api/auth/register`

- HTTP метод: `POST`

- Тело на барање: Содржи детали за корисникот (е-мејл, лозинка итн.) претставени преку објектот `UserRequest`.

Процес:

- Прво, системот проверува дали веќе постои корисник со истиот е-мејл преку `userService.findByEmailOrNull()`.

- Ако е-мејлот е уникатен, новиот корисник се зачувува во базата на податоци.

- Потоа, на корисникот му се генерира JWT токен преку `tokenService.createToken(user)` и се враќа како одговор на успешната регистрација.

Одговор при успешна регистрација: JWT токен.

Одговор при неуспешна регистрација: Порака дека корисник со истиот е-мејл веќе постои.

**2. Најава на корисник**

Најавата се врши преку `/api/auth/login`, каде што корисникот ги внесува своите креденцијали (е-мејл и лозинка).

- Апи: `/api/auth/login`

- HTTP метод: `POST`

- Тело на барање: Е-мејл и лозинка претставени преку објектот `UserRequest`.

Процес:

- Се проверува дали постои корисник со внесениот е-мејл.

- Лозинката на корисникот се проверува преку `hashService.checkBcrypt()`, која го споредува внесениот хеш со зачуваната лозинка.

- Ако креденцијалите се точни, се генерира JWT токен преку `tokenService.createToken(user)`.

- Корисникот го добива токенот како одговор.

Одговор при успешна најава: JWT токен.

Одговор при неуспешна најава: Порака „Невалидни креденцијали“.

**3. JWT Генерирање и валидација**

JWT се користи за да се потврди идентитетот на корисникот за време на комуникацијата со серверот.

- Генерирање на токен: Токенот се генерира во `TokenService` користејќи ја `JwtEncoder` која е конфигурирана во `JwtEncodingConfig`.

- Валидација на токен: Кога корисникот прави барање со JWT, токенот се верификува во позадина преку `JwtAuthenticationFilter`. Оваа класа:

- Го чита токенот од `Authorization` хедерот.

- Го валидира токенот преку `TokenService`.

- Ако токенот е валиден, информациите за корисникот се зачувуваат во `SecurityContextHolder` и корисникот продолжува со барањето.

- Ако токенот е невалиден, се враќа одговор „Unauthorized“.

**4. JWT Филтер за барања**

Во `JwtAuthenticationFilter`, барањата што доаѓаат до серверот се филтрираат на следниов начин:

- Читање на JWT: Се извлекува JWT токенот од барањето.

- Валидација: Ако токенот е валиден, барањето се дозволува; ако не е валиден, се одбива со статус 401 (Unauthorized).

- Филтрирање на барања: Овој филтер не се применува на патеките за најава, регистрација и на некои други јавни API патеки.

**5. Заштита на API патеки**

Конфигурацијата на безбедноста во апликацијата се изведува преку `SecurityConfig`. Оваа класа дефинира кои патеки треба да бидат дозволени и кои треба да бидат заштитени:

- Дозволени патеки: Патеките за најава, регистрација и јавните API патеки се дозволени без автентикација

**6. Angular Клиент - Управување со автентикација**

Angular клиентот го користи `AuthService` за да управува со автентикацијата. Овој сервис:

- Се грижи за зачувување и добивање на JWT токен од `localStorage`.

- Методи за најава и проверка: Преку `login()` методот се испраќа барање до серверот, а преку `isLoggedIn()` се проверува дали корисникот е автентициран.

Intercepting барања: `AuthInterceptor` автоматски го прикачува JWT токенот на сите барања до серверот, освен за дозволените јавни патеки.

**Ангулар компоненти**

Во апликацијата се наоѓаат следниве компоненти:

1. **about**: Оваа компонента го прикажува текстот за платформата, нуди информации за нејзината намена, како и за тоа како функционира.
2. **become-worker**: Оваа компонента се користи за корисниците кои сакаат да станат мајстори (работници) на платформата, со информации и форма за регистрација како мајстор.
3. **header**: Компонентата за хедер (насловна лента), која содржи навигациското мени и основните информации за апликацијата како лого или наслов.
4. **home**: Почетната компонента (home page), каде што веројатно се најважните информации.
5. **login**: Компонентата за најава на корисниците. Овде корисниците ги внесуваат своите е-мејл адреси и лозинки за да се логираат во апликацијата.
6. **profile**: Оваа компонента е за профилот на корисникот, каде што се прикажуваат и уредуваат личните податоци како е-мејл, адреса и слично.
7. **register**: Компонентата за регистрација на нови корисници. содржи форма за внесување на основни податоци за креирање нова сметка.
8. **search**: Компонента за пребарување, поврзана со пребарување на мајстори по локација или катергорија.
9. **search-user**: Специфична компонента за пребарување на корисници по име.
10. **servicer**: Компонента за прикажување на детали за мајстор (работник), каде што се наведени информации како име, контакт, категорија и рецензии.
11. **worker**: Оваа компонента прикажува детали за мајсторите (работниците), вклучувајќи нивните услуги, локации, и контакти.

Оваа структура на компоненти сугерира дека апликацијата е организирана да обезбеди функционалност како за корисници кои бараат услуги, така и за работници кои нудат услуги, со систем за најава, пребарување, и профили. Понатаму за поголемите и поважните компоненти следи и подетален опис.  
  
**ServicerComponent**

Е фокусирана на прикажување на деталите за еден мајстор (работник), со цел корисниците да добијат повеќе информации и да остават рецензии за него. На неа, се прикажува целосното име на мајсторот, неговата емаил адреса, мобилниот телефон, категоријата во која работи, општината во која е активен, краток опис на неговите услуги, како и просечната оцена од претходните клиенти.

Во долниот дел од компонентата, доколку корисникот е најавен, има можност да остави коментар и оцена за мајсторот. Постојат полиња каде што може да се избере рејтинг од 1 до 5 и да се внесе текстуален коментар. По поднесувањето на формата, новиот коментар се прикажува веднаш.

Покрај тоа, на истата компонента, се прикажува листа од сите претходни коментари и оценки дадени од други корисници. Секој коментар содржи име на корисникот што го оставил, оцената што ја дал, и неговиот текстуален коментар. Ова овозможува транспарентност и создавање доверба кај потенцијалните клиенти што би го ангажирале мајсторот.

Севкупно, компонентата има двојна функција: да информира за мајсторот и да овозможи интеракција преку рецензии од страна на корисниците.

**SearchUserComponent**

Оваа компонента им овозможува на корисниците да пребаруваат мајстори преку внесување текст во полето за пребарување. Како што корисникот пишува, пребарувањето се ажурира во реално време благодарение на имплементацијата на функциите за забавување и препознавање на промените (debounce и distinctUntilChanged). Секогаш кога некој ќе започне со пребарување, веднаш се прикажува листа на корисници кои одговараат на критериумот, без потреба од дополнителни кликнувања.

Корисникот може да кликне на име на мајсторот за да го отвори неговиот профил. Ова е особено корисно кога се бара специфичен мајстор по име, бидејќи апликацијата брзо враќа резултати и овозможува лесен пристап до профилот.

**SearchComponent**

Оваа компонента нуди понапредно пребарување и филтрирање на мајсторите. Корисниците можат да филтрираат мајстори според категоријата на услуги (на пример: автомеханичар, водоинсталатер) и општината во која работат (на пример: Центар, Ѓорче Петров). Филтрите овозможуваат корисникот да ги ограничи резултатите само на оние мајстори што одговараат на неговите потреби.

Под филтрите се прикажува листа од мајстори кои ги исполнуваат критериумите. За секој мајстор се прикажуваат основни податоци: име, категорија, општина, и телефонски број. Корисникот има опција да притисне на копчето „Повеќе информации“ за да го отвори детален преглед на профилот на мајсторот.

Дополнително, постои функција за страничење (pagination), што значи дека ако има повеќе мајстори, листата е поделена на страници и корисникот може да се движи напред-назад низ нив за да ги разгледа сите достапни мајстори. Ова овозможува поефикасно прелистување кога има многу резултати.

**ProfileComponent**

Оваа компонента е одговорна за прикажување и управување со профилот на корисникот. На корисникот му се прикажуваат основните податоци како име и презиме, и емаил, а ако корисникот е мајстор, се додаваат и дополнителни информации како адреса, општина, телефонски број и краток опис. Овие информации се внесуваат и уредуваат преку форми, користејќи Angular Material компоненти, и промените се рефлектираат директно во моделот на податоци благодарение на двонасочната врска на податоци.

Функционалностите на компонентата вклучуваат можноста за зачувување на промените во профилот. По зачувувањето, корисникот се пренасочува на почетната страница. Ако профилот не покажува дека корисникот е мајстор, достапна е опцијата за преминување на страницата за регистрација како мајстор. Оваа компонента ја користи ProfileService за ажурирање на профилот и управување со податоците.

**RegisterComponent**

Оваа компонента се користи за регистрација на нови корисници. Таа ги собира основните информации од корисникот, вклучувајќи име и презиме, емаил, лозинка и потврда на лозинка. Формата за регистрација е изградена со Angular's FormBuilder, што овозможува валидација на внесените податоци. Корисникот мора да ги пополни сите полиња и да ги усогласи лозинките пред да може да ја заврши регистрацијата.

По успешна регистрација, корисникот се пренасочува на почетната страница. Ако лозинките не се совпаѓаат, се прикажува порака за грешка. Регистрацијата се врши преку RegisterService, кој комуницира со серверот за да ја заврши постапката. Компонентата обезбедува и копче за враќање на почетната страница ако корисникот одлучи да не продолжи со регистрацијата.

**Сервиси**

**WorkerService**

WorkerService е сервис кој управува со податоците на мајсторите. Неговата главна функција е да овозможи креирање нови мајстори преку create методот, кој испраќа POST барање на серверот со податоците за мајсторот. Овие податоци се проследуваат како параметар до API-то за мајстори, дефинирано во apiUrl.

**ReviewService**

ReviewService се користи за управување со рецензиите. Овој сервис има два основни метода:

1. review - Овој метод испраќа POST барање на серверот со податоците за оцената или рецензијата. Податоците се проследуваат како параметар до API-то за рецензии.
2. fetchReviews - Овој метод го добива списокот на рецензии за конкретен објект, идентификуван преку ID. Тој испраќа GET барање на серверот и враќа Observable со рецензиите.

**RegisterService**

RegisterService е сервис кој управува со регистрацијата на нови корисници. Методот signup испраќа POST барање на серверот со податоците за регистрација на нов корисник, кои се проследуваат како параметар. API-то за регистрација е дефинирано со apiUrl.

**ProfileService**

ProfileService управува со податоците на профилите на корисниците.

1. getProfile - Овој метод добива профил на корисник по ID преку GET барање и врши мапирање на податоците, така што категоријата и општината се конвертираат во нивните соодветни енумски вредности.
2. searchUsers - Овој метод извршува пребарување на корисници на основа на термин за пребарување и враќа список на профили кои се конвертираат во соодветните енумски вредности.
3. searchUsersPageable - Овој метод обезбедува пребарување на корисници со поддршка за пагинација, враќајќи страница од резултати за дадената страница и големина. Податоците се мапираат за да се конвертираат во соодветните енумски вредности.
4. updateProfile - Овој метод испраќа POST барање за ажурирање на профилот на корисникот со новите податоци.

**AuthService**

AuthService управува со автентикацијата на корисниците.

1. isLoggedIn - Овој метод проверува дали корисникот е логиран со проверка на JWT токенот во локалнота меморија.
2. getAuthToken - Овој метод го добива JWT токенот од локалната меморија.
3. login - Овој метод испраќа POST барање на серверот со податоци за логирање, вклучувајќи емаил и лозинка, и враќа Observable со резултатите од логирањето.
4. getLoginStatus - Овој метод враќа Observable кој го следи статусот на логирање.
5. updateLoginStatus - Овој метод го ажурира статусот на логирање и известува сите претплатници преку loginStatusSubject.

**Use Case Сценарија**

**1. Регистрација на нов корисник**

1. **Корисникот** ја посетува веб-страницата на апликацијата „МојМајстор“.
2. **Корисникот** кликнува на **"Регистрирај се"**.
3. **Корисникот** пополнува форма со лични податоци: име, презиме, емаил адреса, лозинка и потврда на лозинка.
4. **Корисникот** кликнува на **"Регистрирај се"**.
5. Апликацијата ги валидира податоците (на пр. дали лозинките се совпаѓаат).
6. Ако податоците се валидни, корисникот добива потврда за успешна регистрација и е пренасочен на почетната страница.
7. Ако има грешки, корисникот добива пораки за грешки и треба повторно да ги внесе податоците.

**2. Најава на корисник**

1. **Корисникот** посетува веб-страницата на апликацијата „МојМајстор“.
2. **Корисникот** кликнува на **"Најава"**.
3. **Корисникот** внесува емаил адреса и лозинка.
4. **Корисникот** кликнува на **"Најава"**.
5. Апликацијата валида податоците и ако се точни, корисникот се најавува и е пренасочен на почетната страница.
6. Ако податоците се неточни, корисникот добива порака за грешка.

**3. Пребарување на мајстори**

1. **Корисникот** посетува страница за пребарување мајстори.
2. **Корисникот** користи филтери (категорија, општина) за да ги ограничи резултатите.
3. **Корисникот** гледа список на мајстори кои ги исполнуваат критериумите.
4. **Корисникот** кликнува на **"Повеќе информации"** за да го види деталниот профил на мајсторот.

**4. Пребарување на мајстори по име**

1. **Корисникот** користи функција за пребарување по име.
2. **Корисникот** внесува текст во полето за пребарување.
3. Резултатите се ажурираат во реално време.
4. **Корисникот** кликнува на име на мајстор за да го види неговиот профил.

**5. Поглед на профил на мајстор**

1. **Корисникот** ја отвара страницата за деталите на мајстор.
2. На страницата се прикажуваат информации како име, контакт, категорија, општина, краток опис и просечна оцена.
3. **Корисникот** може да остави нов коментар и оценка (ако е логиран).
4. Сите претходни коментари и оценки се прикажуваат на страницата.

**6. Управување со кориснички профил**

1. **Корисник** ја отвара страницата за профилот.
2. **Корисник** може да гледа и уредува свои лични податоци.
3. По внесувањето на промените, корисникот кликнува на **"Зачувај"**.
4. Промените се чуваат и корисникот се пренасочува на почетната страница или страницата за регистрација на мајстор (ако е применливо).

**7. Приказ на информациите за платформата**

1. **Корисникот** кликнува на **"За нас"** во хедерот или мениото.
2. **Корисникот** чита информации кои ја опишуваат платформата, нејзината намена и како функционира.