

HOTEL ANIPIA

Webová aplikace slouží pro uživatele (zákazníky), které chtějí vytvořit rezervaci pokoje pro svého domácího mazlíčka. Zatím tato aplikace jen umožňuje vyplnit kontaktní formulář na stránce, který se následně:

- uloží do databáze (H2)
- odešle jako e-mail prostřednictvím Mailtrap

Celkově projekt je postaven pomocí:

Frontend

- HTML
- CSS
- JavaScript
- Skript pro AJAX odesílání dat

Backend

- Java 17
- Spring Boot
- Spring Web, Spring Data JPA, Spring Mail
- H2 databáze
- Mailtrap (pro simulaci e-mailového serveru)
- Maven (pro správu závislostí)

Databáze H2

Byla zvolena především kvůli její jednoduchosti, rychlosti a snadné integraci s frameworkem Spring Boot. H2 je ideální pro testování, protože umožňuje běžet jako in-memory databáze a její velkou výhodou je také webová konzole, která usnadňuje správu a testování SQL dotazů. Všechny zprávy se automaticky ukládají do data/anipia.mv.db. Aby se dostat do konzole H2, musí se v prohlížeči zadat `http://localhost:8080/h2-console`. Odeslání e-mailu běží asynchronně, aby neblokovalo uživatele.

Mailtrap

Pro testování odesílání e-mailů v aplikaci bylo zvoleno použít Mailtrap, protože umožňuje bezpečně simulovat a sledovat e-mailovou komunikaci. Také poskytuje přehledné uživatelské rozhraní, ve kterém lze kontrolovat obsah i formát zpráv. Navíc se snadno integruje se Spring Bootem (což je ideální pro tenhle projekt) a dalšími vývojářskými nástroji pomocí jednoduchého SMTP nastavení.

Struktura projektu

```
ANIPIA/
├── data/
│   ├── anipia.mv.db
│   └── anipia.trace.db
├── Documentation/
│   ├── Diagrams/
│   │   └── UI/
│   ├── Business story.pdf
│   └── Product page.png
├── Spring/anipia/
│   └── src/
│       ├── main/java/com/anipia/
│       │   ├── controller/
│       │   ├── model/
│       │   ├── repository/
│       │   ├── service/
│       │   └── AnipiaApplication.java
│       ├── resources/
│       │   ├── static/
│       │   │   ├── CSS/
│       │   │   ├── Photos/
│       │   │   └── *.html, *.js
│       │   ├── templates/
│       │   └── application.properties
└── README.md
```

Odeslání formuláře

Pro tenhle proces byl pomocí Visual Paradigm vytvořen "Business Process Model and Notation" a "Use Case Diagram", které ho snadně vizuálně popisují. Pro spuštění a nastavení projektu jsou následující kroky:

1. Uživatelský vstup (Frontend)

- Uživatel otevře webovou stránku (např. index.html), kde najde kontaktní formulář.
- Vyplní pole: jméno, e-mail a zpráva.
- Po stisknutí tlačítka Submit JS (script.js) zachytí událost odeslání formuláře.
- Místo klasického odeslání stránky se spustí AJAX volání pomocí fetch(), které vytvoří HTTP POST požadavek na backend API (<http://localhost:8080/contact/submit>).
- Jestli uživatel něco nezadá, tak mu pošle oznámení, že mu něco chybí.
- Data jsou odeslána ve formátu JSON (name, email, message).

2. Přijetí požadavku a zpracování (Backend - Controller)

- Backendová metoda v `ContactFormController` přijme JSON tělo požadavku (`@RequestBody ContactForm contactForm`).
- Spring automaticky namapuje JSON na objekt `ContactForm`.
- Zavolá se služba `contactFormService.saveContactForm(contactForm)`.

3. Uložení do databáze (Backend - Service a Repository)

- V `ContactFormService` se pomocí `contactFormRepository.save(contactForm)` uloží nová zpráva do databáze H2.
- Databáze automaticky přiřadí unikátní ID každé zprávě.

4. Odeslání e-mailu (Backend - Service)

- Po úspěšném uložení služby pokračuje odesláním e-mailu.
- Metoda `sendEmail(ContactForm)` vytvoří e-mailovou zprávu pomocí `SimpleMailMessage`.
- E-mail je odeslán na e-mail adresu hotelu přes SMTP server Mailtrap.

5. Odpověď backendu (Controller na Frontend)

- Pokud vše proběhne v pořádku, backend vrátí „Form submitted successfully“.
- Pokud nastane nějaká chyba, vrátí so terminálu stav 500 INTERNAL SERVER ERROR s odpovídající chybovou zprávou.

6. Zobrazení výsledku uživateli (Frontend)

- JS přijme odpověď z backendu.
- V případě úspěchu zobrazí uživateli zprávu „Message sent successfully!“.
- V případě chyby zobrazí „Error sending message!“.

API endpointy

1. POST /contact/submit

Umožňuje odeslat kontaktní formulář (z webu nebo přes klienta).

2. GET /contact/ping

Kontrola správnosti serveru, zda běží.

3. GET /contact/by-email

Vrátí všechny zprávy zadaného uživatele podle jeho e-mailu.

4. DELETE /contact/deleteByEmail

Smaže všechny zprávy z databáze podle e-mailu.

5. GET /export/pdf

Export všech zpráv do PDF souboru.

Testování API (přes curl - cmd)

Příkaz pro přijetí dat formuláře a jejich zpracování:

```
curl -X POST http://localhost:8080/contact/submit ^ -H "Content-Type: application/json" ^ -d '{"name": "Uzivatel", "email": "uzivatel@test", "message": "Testovací zprava z formulare."}'
```

Příkaz pro testování, že backend běží správně:

```
curl "http://localhost:8080/contact/ping"
```

Příkaz pro výpis všech zpráv od určitého uživatele podle e-mailu:

```
curl "http://localhost:8080/contact/by-email?email=uzivatel@test"
```

Příkaz pro odstranění všech zpráv z databáze podle e-mailu:

```
curl -X DELETE "http://localhost:8080/contact/deleteByEmail/uzivatel@test"
```

Příkaz pro export všech zpráv z databáze do formátu PDF:

```
curl -o message_forms.pdf http://localhost:8080/export/pdf
```

Nebo v prohlížeči: <http://localhost:8080/export/pdf>