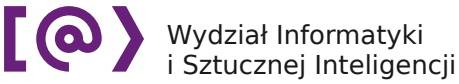
POLITECHNIKA CZĘSTOCHOWSKA
WYDZIAŁ INFORMATYKI I SZTUCZNEJ INTELIGENCJI



PRACA DYPLOMOWA INŻYNIERSKA

Aplikacja klient-serwer

client-server application

Dawid Ziora
Nr albumu: 136700
Kierunek: Informatyka
Studia: stacjonarne
Poziom studiów: I
Promotor pracy:
dr inż. Bartosz Kowalczyk

Praca przyjęta dnia:

Podpis promotora:

Częstochowa, 2024

Spis treści

Wstęp

Aplikacje webowe są bardzo powszechnie używanymi aplikacjami naszych czasów. Coraz częściej chcemy aby na naszych urządzeniach nie pojawiało się wiele różnego rodzaju aplikacji, tylko by wszystko mogło zostać upakowane w przeglądarke internetową z której prosto i szybko uzyskamy dostęp do programów i aplikacji których potrzebujemy. W tej pracy postaram skupić się na aplikacji typu fintech, czyli aplikacji związanej z usługami finansowymi.

cel pracy

Celem mojej pracy będzie uzyskanie działającej aplikacji bankowej w której klient będzie porozumiewał się z serwerem przy pomocy różnego rodzaju zapytań.

zakres pracy

Językiem z którego skorzystam podczas pisania pracy będzie Java oraz Spring (Do logiki serwerowej - Backendu), baza danych MongoDB do obsługi bazy danych oraz Angular i npm do obsługi frontendu.

opisówka

1. Technologie, z których korzysta aplikacjia

1.1. Spring Framework [1]

Jest to struktura, która dostarcza funkcjonalność dla aplikacji. Odpowiada ona między innymi za:

- wstrzykiwanie zależności (Dependency Injection)
- Obsługe zdarzeń (Events)
- Zarządzanie zasobami (Resources)
- Walidacje (Validation)
- Wiązanie danych (Data binding)
- Konwersje typów (Type conversion)
- SpEL
- AOP

Spring jest bardzo popularnym frameworkiem do budowania aplikacji webowych. Jego popularność zawdzięcza takim rzeczą jak

- łatwe wstawianie zależności (Dependency Injection)
- dobrze zintegrowany z innymi frameworkami javy takimi jak JPA/Hibernate
- Posiada framework MVC do budowania aplikacji webowych

1.1.1. Spring boot

- **Starter** pozwala skorzystać z wcześniej skonfigurowanych zależności
- przykłady starterów
  - spring-boot-starter-data-jpa
  - spring-boot-starter-web

- **Automatyczna konfiguracja** - spring boot jest wyposażony w narzędzie, które na bazie zależności **jar**, które dodaliśmy do projektu próbuje skonfigurować projekt

1.1.2. Spring MVC

Jest on zaawansowanym frameworkiem webowym. Składa się on z takich elementów jak:

- Dyspozytor Serwletu (DispatcherServlet)
- Mapowanie żądań (Request Mapping)
- Metody obsługi żądań (Handler methods)
- Obsługa wyjątków (Exepcitons handling)

1.2. angular

Ważne komendy:

- Tworzenie komponentów
  ng g c [nazwa]
  ng generate component [nazwa]
  ng generate component sciezka/[nazwa]
- sprawdzenie co wykona komenda
  –dry-run
- DataBinding - tworzenie zmiennych w tekst tak jakby
  zmienna - tradycyjny sposob
  funkcja () - signal
- Routing - bardzo ważna rzecz
  służy do tworzenia strony na jednej stronie
  Wszystko w app.routes
  W templatce podajemy <routing> i wtedy w zaleznosci od adresu przenosi nas na odpowiednią strone
  item Tworzenie klas
  ng g class [nazwa]

item Tworzenie serwisów

ng g s [nazwa]

- łączenie się z serverem - Tworzymy serwis o jakiejś nazwie następnie w środku serwisu umieczsamy adress http z którego skorzystamy oraz prywatny adres klienta

Listing 1.1. Łączenie z serwerem

```
@Injectable({
 providedIn: 'root'
})
export class ServiceName {
  //bazowy adres serwera
  private baseUrl = 'http://localhost:8080/list';

  constructor(private  httpClient: HttpClient) { }

  getUsersList(): Observable<User[]>{
    return
        this.httpClient.get<User[]>(`${this.baseUrl}`);
  }
}
```

- Wstrzykiwanie zależności jest możliwe poprzez polecenie @Injactible oraz narzędzia dostarczania usług (providers). Tworzymy usługe w niej właśnie znajduje się możliwość wstrzyknięcia tejże usługi. Następnie w jakimś komponencie może zostać wstrzyknięta usługa.

Listing 1.2. Wstrzyknięcie serwisu

```
export class NazwaKomponentu implements OnInit {
 constructor(private userService: UserService) {
  //Wstrzykiwanie uslugi
 }
 //Dalsza przykladowa funkcjonalnosc (uzycie uslugi)
 ngOnInit() {
  this.getUsers();
 }
 getUsers() {
```

```
10    this.userService.getUsersList().subscribe(data => {
11     this.users = data;
12    });
13  }
```

1.3. MongoDB

2. Wymagania i diagramy

2.1. wymagania funkcjonalne

2.2. wymagania niefunkcjonalne

2.3. Diagramy przypadków użycia

2.4. Diagramy sekwencji

3. Kod, funkcje, itd. CORE

4. podsumowanie oraz wnioski

Co osiągnąłem

Bibliografia

[1] H. Kopka and P. W. Daly. *A Guide to LaTeX*. Addison-Wesley, Reading, MA, 1999.

Spis rysunków

Spis tabel