AKADEMIA GÓRNICZO-HUTNICZA

Wydział Informatyki, Elektroniki i Telekomunikacji

Katedra Informatyki

Dokumentacja procesowa, specyfikacyjna

Zespół:

Izabella Szlósarczyk, Dawid Suder

1. OCENA RYZYKA

1.1 Tabela oceny ryzyka

Ryzyko związane z **funkcjonalnością** aplikacji:

lp.	user story	Konsekwencje błędów	Ocena ryzyka (0-10)	Uzasadnienie
1.	Powiadomienia Mailowe	użytkownik nie otrzyma notyfikacji	**2**	Wymaga jedynie podstawowej funkcjonalności
2.	Przeglądanie ofert	wylistowane oferty będą złe oferty	**7**	Jest to główna funkcjonalność aplikacji, dlatego ważne , by oferty były poprawnie filtrowane i wyświetlane, na ten fakt wpływa wiele naszych modułów
3.	Dodanie oferty	niepoprawności w dodawaniu oferty	**7**	Kolejną ważną funkcjonalnością naszej aplikacji jest możliwość dodania oferty, dlatego ważnym jest, aby poprawnie działała. Ponieważ zależy ona od wielu innych modułów, ryzyko wystąpienia błędów jest wysokie.
4.	Edycja/usunięcie oferty	oferta jest niepoprawnie edytowana/usuwana, dodana jest kolejna	**4**	Operacja na bazie danych przy błędnej implementacji może generować błędne wpisy, jednak jest to stosunkowo łatwy do uniknięcia problem.
5.	Edycja/usunięcie użytkowników	użytkownik jest niepoprawnie edytowany/usuwany, dodany jest kolejny	**4**	Operacja na bazie danych przy błędnej implementacji może generować błędne wpisy, jednak jest to stosunkowo łatwy do uniknięcia problem.
6.	Podgląd konta	użytkownik dostaje niepoprawne dane	**3**	Ryzyko jest związane z operacjami wykonywanymi na bazie danych, jak również przekazywanie danych pomiędzy *front-endem* a *back-endem*
7.	Odpowiedź na ofertę	Odpowiedź na ofertę nie zostaje poprawnie zarejestrowana w systemie	**2**	Ze względu na cel istnienia naszej aplikacji sprawa wybrania oferty jest kluczowa. Funkcjonalność ta jest zależna od implementacji pozostałych.

Ryzyko związane z **tworzeniem** i przyszłym **działaniem aplikacji**:

lp.	Rodzaj ryzyka	Ocena ryzyka (0-10)	Funkcjonalności w których ryzyko występuje
1.	**mała znajomość API facebookowego**	**8**	• logowanie
2.	**Małe doświadczenie zespołu z komunikacją na linii Java - MongoDB, możliwość pojawienia się niespójnych danych, problemy z (transakcyjnością) MongoDB**	**5**	• przeglądanie ofert, • przeglądanie ofert od ostatniego logowania • przeglądanie ofert(administrator) • tworzenie ofert, • edycja ofert, • edycja wszystkich ofert, • tworzenie konta, • usuwanie konta, • zmiana hasła, • edycja danych kontaktowych, • edycja konta
3.	**konieczność wykorzystania bibliotek/sterowników niepewnej jakości**	**2**	• cała aplikacja
4.	**Problem z integracją aplikacji na linii front-end - back-end**	**8**	• cała aplikacja

Zagrożenia **związane z produktem**:

Jak w każdej aplikacji webowej spotykamy się z takimi problemami jak:

- **możliwe koszty/problemy serwera** - w założeniu projekt będzie operować na serwerach darmowych, jednak może dojść do sytuacji, gdy będą one niewystarczające, dodatkowym możliwym problemem mogą być technologie pod jakimi pracują serwery

- **przechowywanie danych -** dane powinny być trzymane w taki sposób, by były nieprzekłamane, aplikacja praktycznie każdą operacje wykonuje na danych, jej dobre działanie zależy wobec tego od prawdziwości danych przechowywanych w bazie
- **bezpieczeństwo** - dane systemu muszą być chronione przed wglądem osób trzecich; ze względu na dane osobowe, które będą przechowywane w bazie, jest to szczególnie ważne
- **stabilność** - odpowiednio zaplanowana architektura powinna zabezpieczyć aplikację przed dużą częścią problemów, powinna również zapewnić obsługę błędów, sytuacji wyjątkowych

RYZYKO	Prawdopodobieństwo	Konsekwencje
bezpieczeństwo	20%	3
stabilność	50%	2
systemy zewnętrzne	15%	2
koszty serwera	5%	1

3. Moduły aplikacji

3.1 Bazy danych

3.1.1 Opis

Do przechowania listy kontaktów użytkownika wykorzystujemy nierelacyjną bazę danych, z uwagi na liczne zależności. W bazie przetrzymywane będą informacje na temat :

- Użytkownika
    - imię, nazwisko, e-mail, login, hasło, płeć, wiek
- Oferty (*Offer*)
    - typ oferty, właściciel oferty, preferowane terminy, płeć, wiek, miejsce
- Timeru - dokument ten pomoże w stworzeniu terminu spotkania
    - data spotkania, godzina rozpoczęcia i zakończenia

3.1.2 Wykorzystane technologie

Moduł został zrealizowany przy użyciu MongoDB, SpringBOOT. Model danych stworzony został w javie. Za pomocą adnotacji @DBRef zostały zaimplementowane zależności pomiędzy danymi.

3.2 Łączności z bazą danych

3.2.1 Opis

Moduł stworzony do bezpośredniej komunikacji pomiędzy bazą, a logiką aplikacji. Jego zadaniem jest wysłanie zapytań do bazy danych, pobranie ich z bazy.Pobrane dane wysyła dalej do moduły komunikacji z serwerem.
Moduł bazy danych został tak zaprojektowany (poprzez repozytoria) by w zapytania były proste, z użyciem Criteria, które pozwalają na automatyczne zaciąganie referencji.

3.2.2 Wykorzystane technologie

Do stworzenia połączenia pomiędzy logiką aplikacji, a bazą danych wykorzystane zostały repozytoria, Criteria Query (które pozwalają na zachowanie spójności pomiędzy danymi),

3.2.3 Interfejsy

```
interface ReadFromDatabase {
        List<User> getUsersData( );
        List<Term> getOffersData( );
        List<Term>getTimerData( );
}
```

3.3 Moduł komunikacji z serwerem

3.3.1 Opis

Moduł przyjmujący zlecenia od modułów logiki biznesowej, a następnie przekazujący je do modułu łączności z bazą danych. Wszystkie dane, które otrzyma od modułu komunikacji z bazą danych odpowiednio przerabia i przesyła do odpowiedniego modułu logiki biznesowej.

3.3.2 Interfejsy

```
interface AuthorizationHelper {
        String getPassword( User user );
}

interface AccountsManagerHelper {
        User getUserData( String login );
        void changeDataBaseData( User user, OperationType operationType );
}

interface OffersManagerHelper {
        List<Offer> getOffers( );
        void changeOfferDataBaseData( Offer offer, OperationType operationType );
        List<Term> getSchedule( Optional<User> user,
                ScheduleType scheduleType );
        void changeScheduleDataBaseData( List<Term> schedule,
                ScheduleType scheduleType, Optional<User> user );
}

interface MailNotificationsHelper {
        List<User> getUserInformation( Offer offer );
}
```

3.3.3 Jsony

Moduł kontrolera wystawia *URI* i przesyła następujące wiadomości:

URI:	zapytanie	odpowiedź:

/login
 	
(POST)
	{
	"email": "email",
	"password": "password",
	}
		{
		"userId": "userId",
		"token": "token",
		"indexNumber": "number",
		"firstName": "",
		"lastName": "",
		"email": "",
		“sex”: "",
		“age”: “”
		}

/logout (GET)		{"msg": ""}
/offers/all (GET)		[{ { "userId": "", "offerId": "", "userName": "", "userSurname": "", "userMail": "", "typeOfOffer": "", "when":{ "date": "", end": "", start": ""} , "when2":{ "date": "", end": "", start": ""} "place": "", “prefferedAge": "", “prefferedSex": ""} }}]
offers/choose (POST)	{ "userId": "", "offerId": "", "userName": "", "userSurname": "", "userMail": "", "typeOfOffer": "", "when":{ "date": "", end": "", start": ""} , "when2":{ "date": "", end": "", start": ""} "place": "", “prefferedAge": "", “prefferedSex": "" }	{"msg": ""}

offers/{offerId}/delete (GET)		{"msg": ""}

users/{id}/edit
 	
(POST)
	{
	"userId": "userId",
	"token": "token",
	"indexNumber": "number",
	"firstName": "",
	"lastName": "",
	"email": "",
	“sex”: "",
	“age”: “”
	}
		{"msg": ""}

users/new (POST)	{ "indexNumber": "", "name": "", "surname": "", "academicYear": "", "email": "", "password":}	{"msg": ""}
/users/{id}/offers (GET)		[{ { "offerId": "", "typeOfOffer": "", "when":{ "date": "", end": "", start": ""} , "when2":{ "date": "", end": "", start": ""} "place": "", “prefferedAge": "", “prefferedSex": ""} }}]
/users/{id}/offers (POST)	{ "offerId": "", "typeOfOffer": "", "when":{ "date": "", end": "", start": ""} , "when2":{ "date": "", end": "", start": ""} "place": "", “prefferedAge": "", “prefferedSex": ""}	{"msg": ""}

/users/{id}/offers/new (POST)	{ "offerId": "", "typeOfOffer": "", "when":{ "date": "", end": "", start": ""} , "when2":{ "date": "", end": "", start": ""} "place": "", “prefferedAge": "", “prefferedSex": ""}	{"msg": ""}

3.5 Moduł UI

3.5.1 Opis

Moduł interfejsu użytkownika, komunikujący się z kontrolerem za pomocą *REST* . Graficzny interfejs kliencki składa się z nieskomplikowanych widoków obsługiwanych poprzez odpowiednie kontrolery. Spięcie GUI z logiką zapewniają dwa serwisy - OfferService oraz UserService.

3.5.2 Wykorzystane technologie

Moduł został stworzony przy użyciu: html, bootstrap, css,npm i nodeJS. Za pomocą AngluarJS zrealizowana została logika kliencka. Wstępna wersja również zapewniała mock serwer (kiedy jeszcze nie było połączenia z backendem), dzięki, któremu obserwować można było wchodzące/wychodzące jsony.

3.5.3 Widoki

1. Podstawowy widok aplikacji zawierający informacje o aplikacji
2. Logowania
3. Rejestracji
4. Po zalogowaniu
    a. Edycja danych użytkownika
    b. Oferty
        i. Przeglądanie ofert
        ii. Przeglądanie ofert dodanych przez użytkownika

iii. Przeglądanie nowości - od czasu ostatniego logowania

3.6 Moduł zarządzania użytkownikami

3.6.1 Opis

Moduł odpowiedzialny za zarządzanie kontami użytkowników. Będzie on odpowiedzialny za::

- Edycję danych użytkownika
- Tworzenia/usuwanie konta
- Edycja konta (wymagane uprawnienia administratora)

3.6.2 Wykorzystane technologie

Moduł ten będzie realizowane przez odpowiednie kontrolery. Stworzony zostanie w javie.

3.6.3 Interfejsy

3.7 Moduł autoryzacji

3.1.1 Opis

Moduł odpowiedzialny za autoryzację użytkowników w systemie. Będzie realizował funkcjonalności logowania oraz wylogowywania. Dodatkowo zostanie połączony z facebookowym API do autoryzacji.

3.1.2 Wykorzystane technologie

3.1.3 Interfejsy

```
interface AuthorizationService {
        LoginResponse authorizeUser( LoginRequest loginRequest );
}
```

3.8 Moduł obsługi ofert

3.1.1 Opis

Moduł odpowiedzialny za obsługę ofert. Będzie realizował następujące funkcjonalności:

- Tworzenie oferty
- Edycja własnych ofert
- Usuwanie własnych ofert

4. DZIENNIK PROJEKTU

Daty	**Zadanie**	**problemy**	**Stan zadania**	**wykonawca**
25.04.2016	Wstępny szkielet GUI aplikacji **[UI]**	-	**Zakończone powodzeniem**	**Izabella Szlósarczyk**
25.04.2016	Wstępne podpięcie *Angular.js'a* do części odpowiedzialnej za logowanie/wylogowania **[UI]**	-	**Zakończone powodzeniem**	**Izabella Szlósarczyk**
31.04.2016	Stworzenie lokalnego [*mock*] serwera	-	**Zakończone powodzeniem**	**Izabella Szlósarczyk**
31.04.2016	Tworzenie *jsonów*, testowanie ruchu na serwerze **[UI]**	-	**Zakończone powodzeniem**	**Izabella Szlósarczyk**
05.05.2016	Stworzenie lokalnej bazy i generatora danych do testów **[Moduł bazy danych]**	Wygenerowanie bazy w taki sposób, by nie było pustych referencji w repozytoriach	**Zakończone powodzeniem,** zmiana podejścia do tworzenia i dostępu do bazy	**Izabella Szlósarczyk**
05.05.2016	Podłączenie bazy danych do *Springa*, zmiana metod do wyciągania danych z bazy, użycie *Criteria Query* **[Moduł bazy danych]**	Sprawdzenie, testowanie metod	**Zakończone powodzeniem**	**Izabella Szlósarczyk**
20.05.2016	Wstępna próba podłączenia autoryzacji przez	problem połączenia autoryazji z	**w trakcie**	**Izabella Szlósarczyk**

	facebooka **[moduł autoryzacji]**	front endem i back endem		
20.05.2016	Poszerzenie bazy danych o dokument Timera **[Moduł bazy danych]**	-	**zakończone powodzeniem**	**cały team**
20.05.2016	Update szkieletu projektu	-	**-**	**cały team**