WebTicket

System do zarządzania zgłoszeniami w przedsiębiorstwie

Dokumentacja projektu

Autor: Tomasz Bajorek

Kraków, 28.08.2017r.

1. Cele projektu

Celem niniejszego projektu było stworzenie przy użyciu technologii Java EE aplikacji internetowej. Zgodnie z założeniami musiała być ona oparta o architekturę klient – serwer z wykorzystaniem wzorca RESTful. Dodatkowym wymaganiem było wykorzystanie w projekcie serwera bazy danych.

Tematem projektu jest aplikacja webowa, dedykowana firmom. Umożliwia ona zgłaszanie różnych problemów wewnątrz przedsiębiorstwa, które następnie mogą być rozwiązywane przez innych pracowników. Korzystanie z systemu jest możliwe dzięki stworzonemu klientowi, napisanemu w języku JavaScript.

Niniejsza dokumentacja opisuje wymagania postawione aplikacji oraz jej funkcjonalności. Zawiera diagramy oraz graficzne schematy. Znajduje się tutaj także informacja wdrożeniowa oraz podstawowy podręcznik użytkownika.

2. Wymagania techniczne

Przed projektem zostały postawione pewne wymagania techniczne. Podstawowym z nich było, iż aplikacja musi zostać wykonana w technologii Java EE z Java API for RESTful Web Services JAX-RS oraz będzie uruchomiona w ramach chmury obliczeniowej Bluemix. Baza danych również musi znajdować się w rozwiązaniu chmurowym. Dostęp do niej powinien być zrealizowany w oparciu o odwzorowanie obiektowo – relacyjne JPA. Dodatkowo w ramach serwera należy wykorzystać programowanie aspektowe (język AspectJ). W przypadku klienta jest dozwolone korzystanie z odpowiednich frameworków.

3. Wymagania funkcjonalne

Aplikacja powinna udostępniać swoją funkcjonalność po zalogowaniu. Rejestracja nie może być dostępna dla każdego, ale tylko po zaproszeniu danej osoby przez kogoś do tego upoważnionego.

Firma może być podzielona na działy lub departamenty, co powinno być możliwe do odwzorowania w aplikacji. Między nimi powinna być możliwa interakcja.

Konta użytkowników, podobnie jak pracownicy w firmie, mogą mieć różne uprawnienia. Administrator ma prawo zarządzać całą funkcjonalnością, kierownicy – swoimi departamentami, a pracownicy – aktywnościami w ramach przydzielonych im obowiązków. Konto gościa winno umożliwiać mu tylko odwiedzenie strony głównej oraz podstrony: rejestracji i logowania. Wszystkie próby skorzystania z funkcjonalności aplikacji muszą się skończyć wyświetleniem odpowiedniego komunikatu, a jeśli chodzi o część serwerową – brakiem jakiegokolwiek skutku ich wywołania.

Pracownik nie powinien móc przyjąć zgłoszenia, które sam zgłosił. Może je natomiast wysłać zarówno do własnego departamentu, jak też do innego. W ramach zgłoszenia zarówno autor jak i osoba je wykonująca powinni móc wymieniać wiadomości. Po jego zamknięciu autor ma możliwość oceny wykonania przez drugiego pracownika. Nie będzie on jednak o tym wiedział, gdyż informacja o jakości realizowanych zgłoszeń jest dostępna tylko dla kierowników i administratora.

4. Wykorzystane technologie

4.1. Serwer

Aplikacja została stworzona w języku Java[1], w technologii Java EE 7[2]. Program został wdrożony w chmurze Bluemix[3] na serwerze WAS Liberty[4]. Funkcjonalności przydatnej w budowaniu serwisów RESTful[5] dostarczyła technologia JAX-RS[6]. Do mapowania obiektowo – relacyjnego użyto JPA[7]. Jako bazę danych wykorzystano istniejącą również w ramach chmury Bluemix dashDB[8]. Wdrożenie aplikacji następuje przy pomocy narzędzia Cloud Foundry CLI[9]. Testy restowego API serwera zostały przeprowadzone przy pomocy dodatku do przeglądarki Chrome[10] o nazwie Postman[11].

4.2. Klient

Klient jest napisany w języku JavaScript, zgodnym ze standardem ECMAScript 6[12]. Aby było możliwe użycie go w przeglądarce internetowej, kod jest „kompilowany" do wersji obsługiwanej przez większość przeglądarek przy pomocy narzędzia Babel[13]. Proces ten następuje w ramach narzędzie WebPack[14], pozwalającą łączyć w jedną aplikację nie tylko kod JS, ale także HTML, CSS, pliki czcionek oraz obrazy.

Główną biblioteką użytą do napisania klienta jest ReactJS[15]. Pozwala on łatwo tworzyć aplikacje z komponentów. Jego zastosowanie zdeterminowało także użycie następnie dodatkowych bibliotek, takich jak React Router[16] do tworzenia wirtualnych "podstron aplikacji, RefluxJS[17] implementujący wzorzec Flux[18] do zarządzania przepływem danych w ramach programu czy React Quill[19] – do stworzenia prostego edytora WYSIWYG. React ani Reflux nie posiadają wbudowanego wsparcia dla tworzenia żądań AJAX, stąd też do tych celów została zastosowana biblioteka Axios[20]. Elementy interfejsu aplikacji stworzono dzięki użyciu zestawu gotowych stylów CSS, dostępnego w ramach biblioteki – Bootstrap[21].

---

[1] http://www.java.com
[2] http://www.oracle.com/technetwork/java/javaee/overview/index.html
[3] https://www.ibm.com/cloud-computing/bluemix/
[4] https://developer.ibm.com/wasdev/websphere-liberty/
[5] http://www.restapitutorial.com/
[6] https://docs.oracle.com/javaee/7/tutorial/jaxrs.htm
[7] https://docs.oracle.com/javaee/7/tutorial/partpersist.htm
[8] https://www.ibm.com/ms-en/marketplace/cloud-data-warehouse
[9] https://docs.cloudfoundry.org/cf-cli/
[10] https://www.google.pl/chrome/browser/desktop/index.html
[11] https://www.getpostman.com/
[12] http://es6-features.org/
[13] https://babeljs.io/
[14] https://webpack.js.org/
[15] https://facebook.github.io/react/
[16] https://reacttraining.com/react-router/
[17] https://github.com/reflux/refluxjs
[18] https://facebook.github.io/react/blog/2014/05/06/flux.html
[19] https://github.com/zenoamaro/react-quill
[20] https://www.npmjs.com/package/axios
[21] http://getbootstrap.com/

Testy klienta zostały przeprowadzone w środowisku Mocha[22]. Do renderowania komponentów użyto narzędzia Enzyme[23], a do tworzenia asercji – biblioteki Chai[24]. Zostały one zintegrowane przy pomocy biblioteki Chai-Enzyme[25].

5. Szczegółowy opis funkcjonalności

Poniższy opis jest w prawie wszystkich punktach wspólny dla części serwerowej i klienta. Generalnie są one ze sobą koherentne. Walidacja danych oraz sprawdzanie uprawnień są wprowadzone zarówno po stronie serwera, jak i klienta. Pierwszy przypadek gwarantuje bezpieczeństwo, nawet w sytuacji prób ominięcia webowego interfejsu klienta, zaś drugi umożliwia wygodne korzystanie z aplikacji użytkownikowi, bez wyświetlania mu nieprzydatnych funkcjonalności, niedostępnych dla jego typu konta. Różnice z funkcjonalności między klientem i serwerem zostały zaznaczone w konkretnych punktach.

5.1. Rodzaje użytkowników

W systemie mogą istnieć 4 rodzaje kont: gość, pracownik, kierownik oraz administrator. Trzy ostatnie umożliwiają dostęp do funkcjonalności aplikacji.

Użytkownicy mogą zmieniać dane swoich profili, a także tworzyć zgłoszenia oraz wykonywać zgłoszenia innych. W ramach tego mają możliwość pisania, edytowania oraz usuwania własnych wiadomości. Mogą również przeglądać listy zgłoszeń oraz profile innych osób.

Kierownicy widzą oceny przyznawane zgłoszeniom, a także mogą zapraszać nowych użytkowników do swoich departamentów. Mogą również zmieniać stanowiska użytkowników już istniejących.

Administratorzy mają pełne uprawnienia do korzystania ze wszystkich funkcjonalności. Mogą zmieniać zarówno departament jak i stanowisko każdego użytkownika, a także zapraszać nowych pracowników do dowolnego departamentu. Mogą również dodawać, edytować oraz usuwać departamenty. Widzą oceny wszystkich zgłoszeń z każdego departamentu. Jako jedyny rodzaj kont mogą usuwać dowolne zgłoszenia, także niezamknięte.

5.2. Logowanie

Każdy niezalogowany użytkownik może widzieć stronę logowania. Aby potwierdzić swoją tożsamość wymagane jest podanie adresu e-mail użytkownika aplikacji oraz powiązanego z nim hasła. W przypadku wprowadzenia złych danych logowanie się nie powiedzie, zostanie wyświetlony komunikat o błędzie.

Jeżeli dane są poprawne, serwer zwróci dane użytkownika oraz token. Jest on wymagany do potwierdzania każdego żądania zalogowanego pracownika. Przekazanie go następuje poprzez nagłówek X-Token. Token ten jest zapamiętany po stronie klienta w pamięci localStorage przeglądarki do czasu, aż użytkownik wyloguje się z aplikacji.

5.3. Lista zgłoszeń

Każdy zalogowany użytkownik ma możliwość przeglądania listy zgłoszeń stworzonych przez samego siebie, a także tych, które aktualnie wykonuje.

---

[22] https://mochajs.org/
[23] http://airbnb.io/enzyme/index.html
[24] http://chaijs.com/
[25] https://github.com/producthunt/chai-enzyme

Lista zawiera tytuł zgłoszenia, departament, do którego zostało ono skierowane, wykonawcę – jeśli jest w trakcie realizacji, a także daty utworzenia oraz ostatniej modyfikacji. Zamknięte zgłoszenia są oznaczone w specjalny sposób, podobnie jak ocenione. Użytkownik może z tego widoku przejść zarówno do podglądu zgłoszenia, jak również jego autora oraz wykonawcy.

5.4. Zarządzanie zgłoszeniami

Każdy użytkownik może dodać nowe zgłoszenie do dowolnego departamentu włącznie z własnym. Musi podać tytuł zgłoszenia. Kiedy zostanie ono utworzone, autor ma możliwość dodawania do niego wiadomości. Nie ma narzuconego limitu ich ilości. W przypadku pisania wiadomości użytkownik ma do dyspozycji prosty edytor WYSIWYG, dzięki czemu może używać takich elementów jak nagłówek, pogrubienie, podkreślenie, pochylenie, hiperłącze oraz lista punktowa i numerowana. Do wstawienia zgłoszenia do bazy danych a także jego przyjęcia przez inną osobę nie jest wymagana żadna wiadomość od autora. Podobnie na autora nie nałożono żadnych restrykcji, dotyczących tego, kiedy może on zamknąć zgłoszenie. Ma możliwość to uczynić nawet, jeśli nikt nie podjął się realizacji zgłoszenia.

Przyjmować zgłoszenie może każdy pracownik departamentu, do którego zostało ono skierowane. Jedynym wyjątkiem jest sam autor, który nie może przyjąć stworzonego przez siebie zlecenia.

Po zamknięciu zgłoszenia może zostać ono ocenione, lecz tylko wówczas, jeśli jest do niego przypisany wykonawca. Nie jest to jednak czynność obowiązkowa. Ocena nie może być zmieniona po jej wystawieniu. Jest ona widoczna zarówno dla autora, jak też dla kierownika i administratora. Nie może jej natomiast zobaczyć wykonawca.

Jedynie administrator ma możliwość usuwania zgłoszenia. Może to zrobić w dowolnym momencie, niezależnie od stanu jego realizacji.

5.5. Departamenty

Firma może być podzielona na departamenty. Każdy użytkownik musi być przypisany do któregoś z nich. Takie przypisanie otrzymują również wszystkie zgłoszenia. Na tej podstawie pracownik może je przyjąć do realizacji. Departament jest przydzielany użytkownikowi w momencie zaproszenia go do systemu. Może zostać zmieniony jedynie przez administratora.

Każdy użytkownik może przeglądać listę pracowników swojego departamentu oraz zgłoszone tam zgłoszenia. Kierownik widzi dodatkowo listę zaproszeń do swojego departamentu. Administrator natomiast ma prawo przeglądać wszystkie powyższe listy dla każdego z departamentów. Może ona także zmieniać ich nazwę.

5.6. Profile użytkowników

Każdy użytkownik posiada swój profil, widoczny dla innych pracowników, składający się z imienia, nazwiska, adresu e-mail, stanowiska oraz departamentu. Może on edytować imię, nazwisko oraz adres e-mail. Dodatkowo istnieje możliwość zmiany hasła. Kierownik ma możliwość edycji stanowisk wszystkich pracowników swojego departamentu. Administrator może przenosić użytkowników między departamentami.

Kierownik oraz administrator widzą dodatkową informację na profilu użytkowników – ocenę wykonywania przez nich zleceń. W przypadku kierownika dotyczy to tylko pracowników jego departamentu, a administratora – wszystkich użytkowników.

5.7. Dodawanie nowych kont

Program nie daje możliwości rejestracji konta. Dodanie nowego użytkownika następuje wyłącznie po zaproszeniu go przez kierownika lub administratora. Kierownik może zapraszać tylko do swojego

departamentu, a administrator – do dowolnego departamentu w firmie. Aby zaprosić nowego użytkownika należy podać jego adres e-mail oraz przyszłe stanowisko.

Po dodaniu nowego zaproszenia kierownik lub administrator mogą skopiować link, niezbędny do rejestracji nowego konta i wysłać go do użytkownika. Wysyłanie to odbywa się poza aplikacją.

Nowy użytkownik po otrzymaniu linku zapraszającego i otwarciu go musi podać swoje imię, nazwisko oraz hasło do logowania. Po zatwierdzeniu tych danych zostanie utworzone nowe konto, a pracownik może się już zalogować.

5.8. Kamienie milowe

Oprócz opisanej powyżej funkcjonalności wspólnej dla klienta i serwera, ten drugi posiada jeszcze możliwość dodawania do każdego zgłoszenia tzw. kamieni milowych. Dodawać je może zarówno autor jak też wykonawca zgłoszenia. Obydwaj ci pracownicy mają możliwość późniejszego oznaczania kroków milowych jako wykonane, a także ich usuwania lub zmiany nazwy. Funkcjonalność ta nie została zaimplementowana po stronie klienta.

6. Architektura REST

Program został stworzony w oparciu o architekturę RESTful z podziałem na część serwerową i klienta. Serwer oraz baza danych, z której on korzysta, znajdują się w chmurze obliczeniowej Bluemix. Architektura ta została pokazana na diagramie 1.

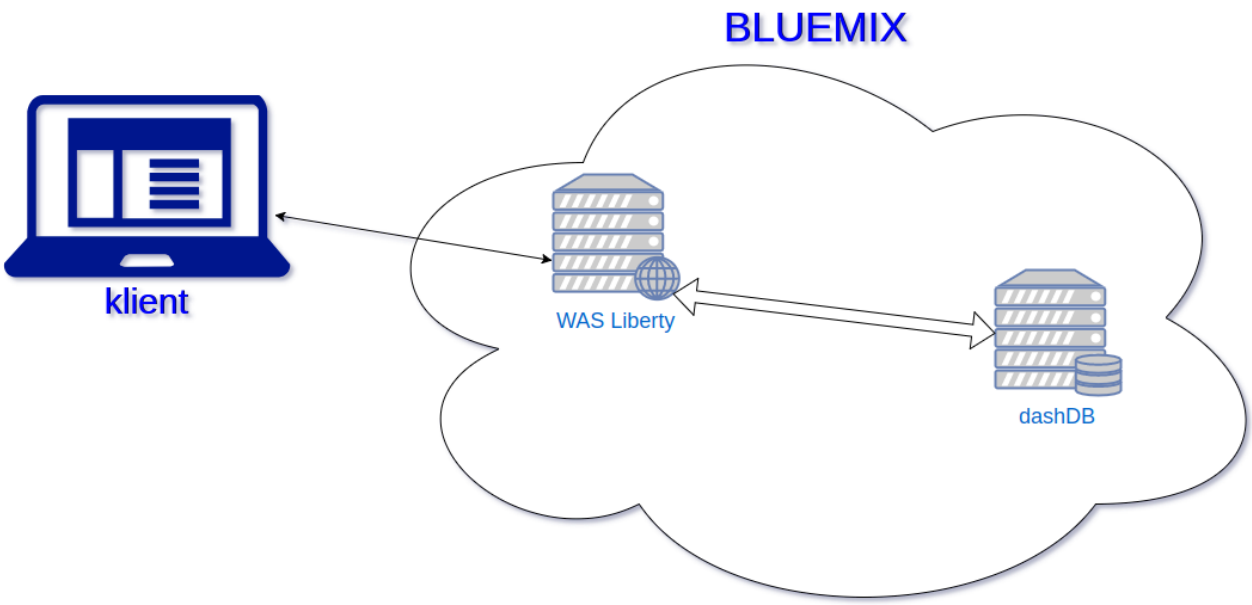


Diagram 1 Ogólny schemat architektury aplikacji WebTicket

6.2. Zasoby

Serwer umożliwia dostęp do swoich zasobów przy pomocy protokołu HTTP oraz czterech podstawowych jego metod: GET, POST, PUT oraz DELETE. Reprezentują one cztery podstawowe operacje CRUD (create, read, update, delete). W poniższej tabeli 1 zostały przedstawione wszystkie zasoby serwera, wraz ze ścieżkami, krótkim opisem, a także informacją, czy skorzystanie z nich wymaga uwierzytelnienia oraz jaka minimalna rola jest potrzebna.

Lista wszystkich zasobów wraz z ich opisem znajduje się w załączniku nr 1.

7. Struktura aplikacji

7.1. Baza danych

Program korzysta z bazy danych dashDB, znajdującej się w chmurze obliczeniowej Bluemix. Zarówno do mapowania obiektowo – relacyjnego, jak też do generacji schematu bazy danych użyto JPA. Diagram 2 przedstawia schemat ERD modelu danych, reprezentowanego w bazie.

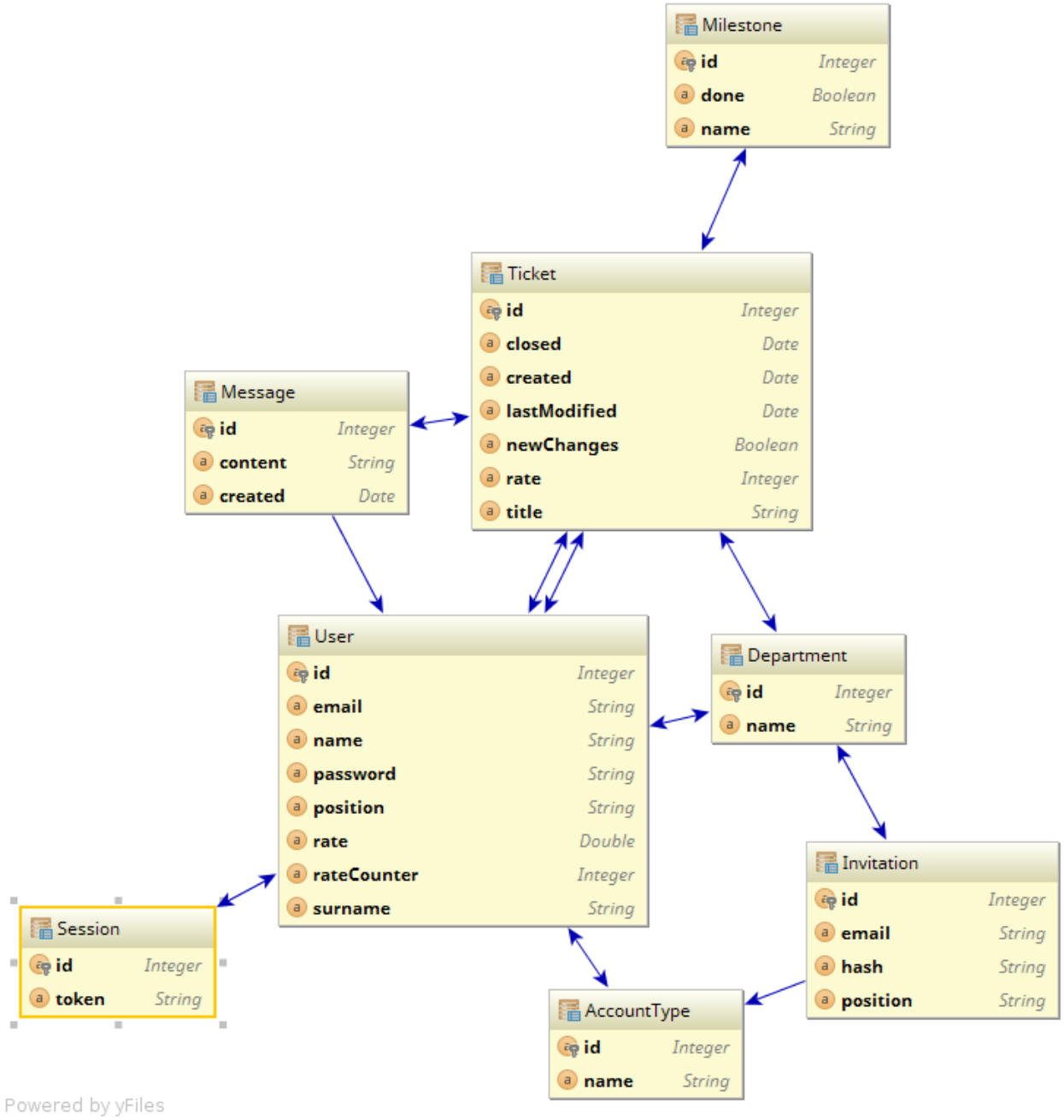


Diagram 2 Diagram ERD bazy danych aplikacji WebTicket

Tabela 1 zawiera krótki opis każdej z encji.

Nazwa	Opis
AccountType	Rola użytkownika, jaką może posiadać (gość, pracownik, kierownik, administrator)
Department	Informacje o departamencie
Invitation	Dane potrzebne do zaproszenia nowego użytkownika do aplikacji
Message	Wiadomości pisane w ramach zgłoszeń, wraz z informacją o autorze i dacie utworzenia
Milestone	Kamienie milowe zgłoszeń
Session	Tokeny sesji użytkowników, używane przez nich do potwierdzania swojej tożsamości
Ticket	Dane poszczególnego zgłoszenia, takie jak jego tytuł, autor, wykonawca, ocena, a także daty: utworzenia, ostatniej modyfikacji i zamknięcia
User	Informacje o użytkownikach, a także ich hasła przechowywane w postaci ciągów SHA-256

Tabela 1 Opis encji programu WebTicket

7.2. Serwer

Kod Javy został podzielony na pakiety według odpowiedzialności. W poniższej tabeli 3 przedstawiono krótki opis każdego z pakietów. Szczegółowe informacje o każdej z klas znajdują się w dokumentacji, wygenerowanej przy pomocy narzędzia JavaDoc.

Nazwa	Opis
api	Klasy, odpowiadające za udostępnianie przez serwer zasobów. Zazwyczaj ich metody są powiązane z tzw. punktami końcowymi, czyli adresami, pod którymi serwer oferuje dane lub wykonanie operacji. Dziedziczą one po abstrakcyjnej klasie AbstractEndpoint, udostępniającej im podstawową funkcjonalność, używaną w większości z nich.
exception	Wyjątki, używane w aplikacji. Sygnalizują błąd autentykacji, nieprawidłową liczbę sesji użytkownika czy brak praw dostępu do danego zasobu.
model	Klasy modelu danych. Większość z nich to encje, używane do mapowania obiektowo relacyjnego.
response	Obiekty tych klas mogą być zwracane do użytkownika. Reprezentują one podobne byty, jak encje, ale zawierają ograniczoną ilość zagnieżdżonych relacji.
rest	Klasa dostarczająca zasoby aplikacji do głównego mechanizmu REST w Java EE.
tool	Klasy narzędziowe aplikacji – do generowania ciągów SHA-256 z podanego wejścia oraz do logowania informacji do plików.
wtaspect	Pakiet zawierający aspekty języka AspectJ, dzięki którym można rozszerzać oraz modyfikować działanie aplikacji.

Tabela 2 Pakiety serwera WebTicket

Diagram klas całej aplikacji był zbyt duży, aby umieścić go w tej dokumentacji. Został więc dołączony jako załącznik nr 2. Poniższe diagramy przedstawiają wybrane pakiety programu.

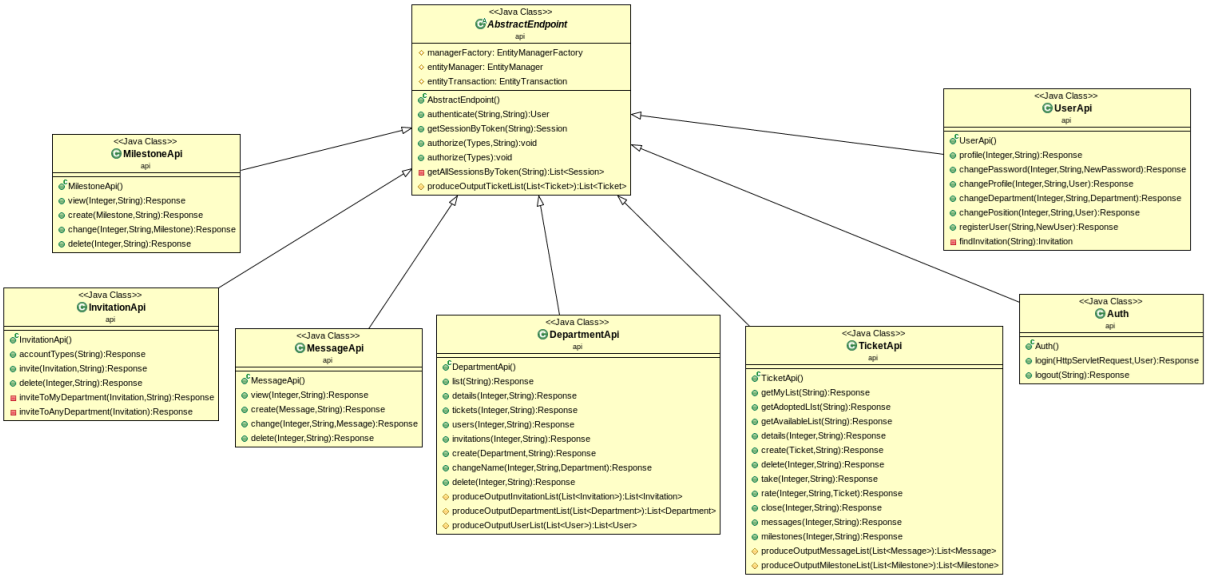


Diagram 3 Diagram klas pakietu api

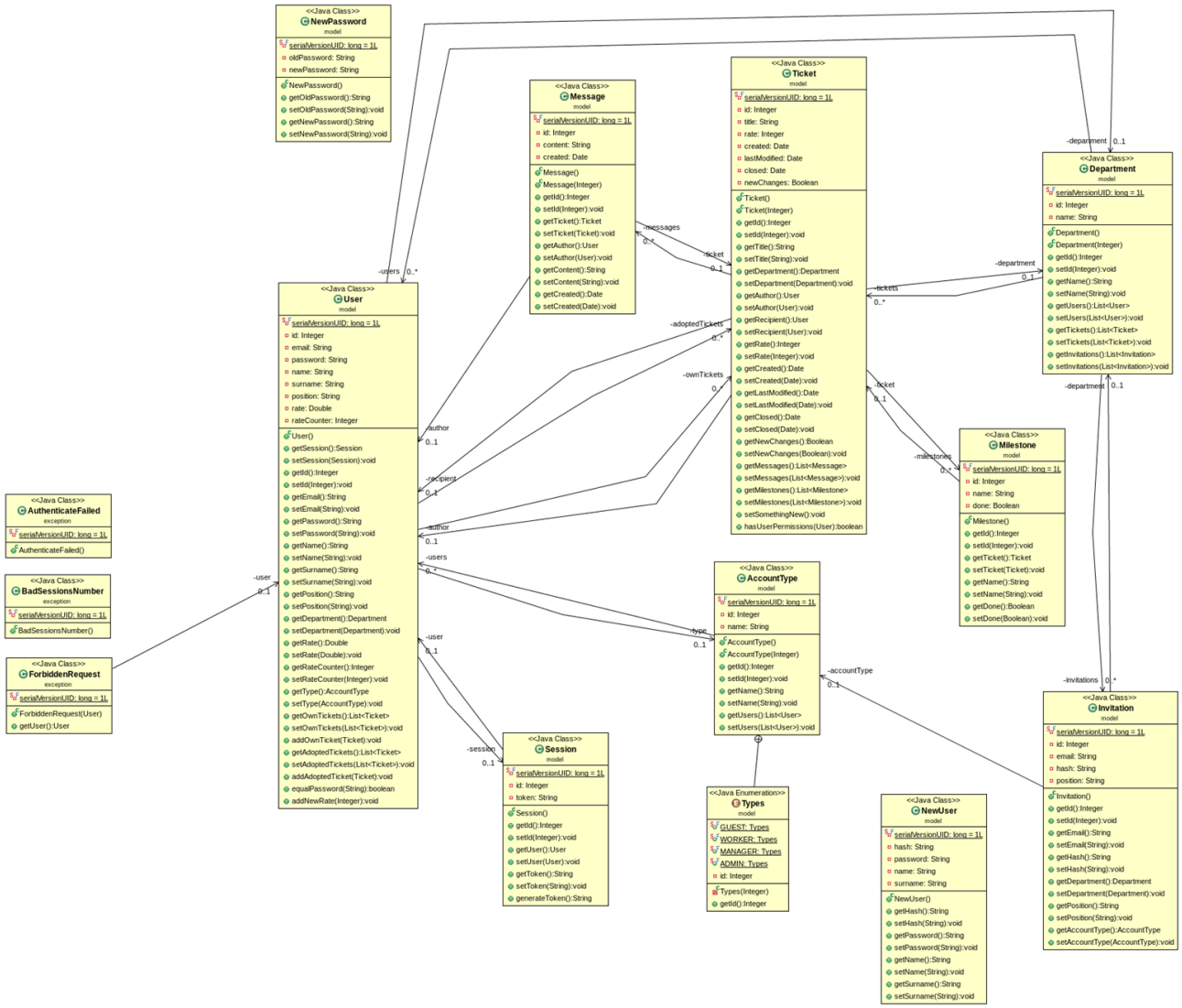


Diagram 4 Diagram klas pakietów model oraz exception

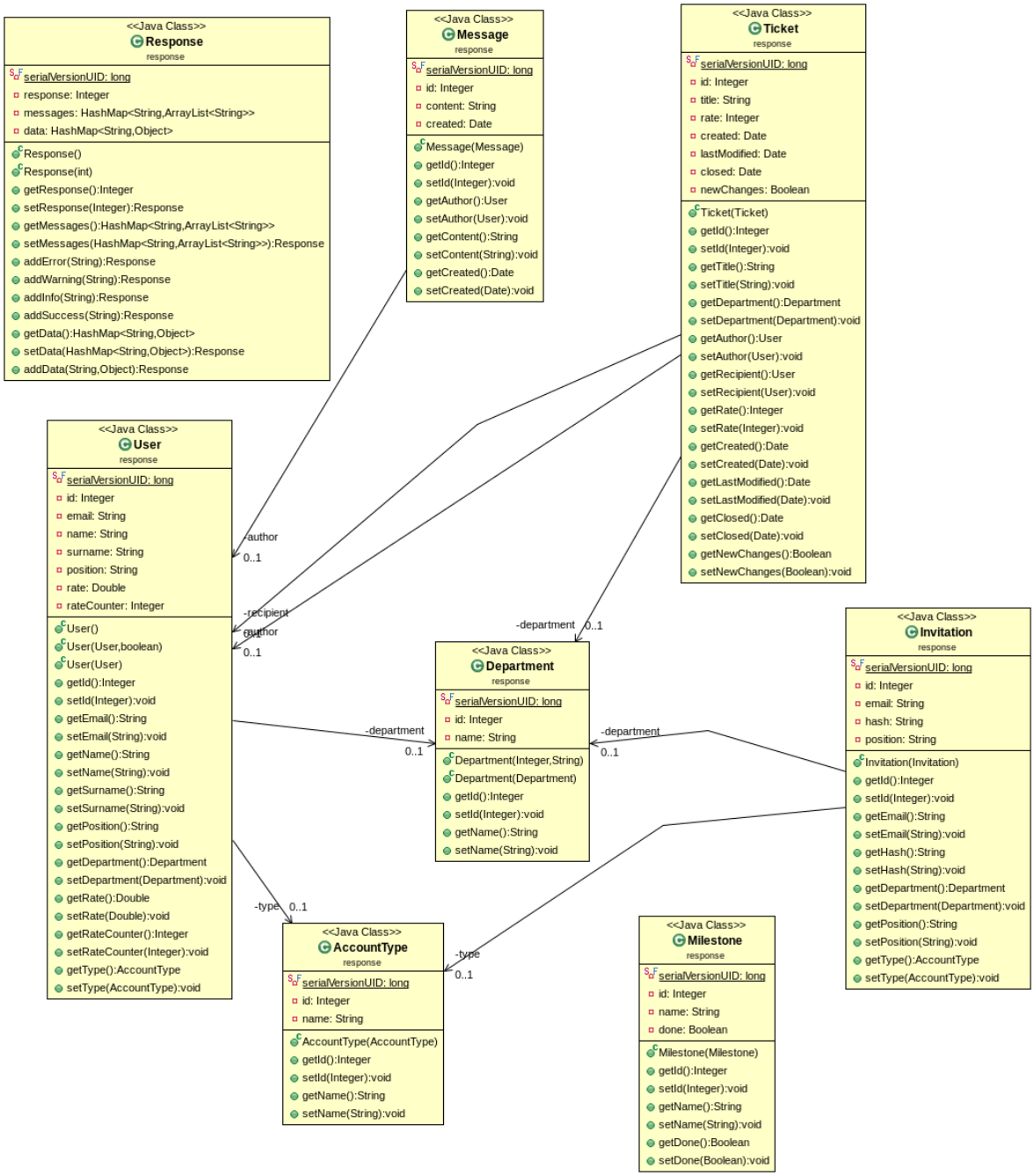


Diagram 5 Diagram klas pakietu response

7.3. Klient

Zastosowanie nowej wersji języka ECMA 2016 umożliwiło budowę klienta w sposób modularny dzięki użyciu klas, niedostępnych w podstawowym JavaScripcie. Dodatkowo użycie biblioteki React pozwoliło budować aplikację w sposób komponentowy. Każdy element aplikacji to klasa albo komponent bezstanowy.

Aplikacja pracuje w ramach jednej strony internetowej, zawartej w pliku *index.html*. Dzięki narzędziu React Router jest możliwe tworzenie wirtualnych podstron. Dzięki temu można korzystać zarówno z paska adresu przeglądarki, jak również z historii przeglądania tak, jak w przypadku innych normalnych stron internetowych.

Komponenty z głównego katalogu definiują główne elementy widoku aplikacji. Są to nagłówek, blok treści oraz stopka. Bezpośrednio do React Router są podłączone komponenty, znajdujące się w katalogu *pages*. Określają one główny wygląd strony i dołączają konkretne komponenty.

W katalogu *actions* znajdują się definicje akcji, które mogą być wywoływane na tzw. storach, przechowywanych w katalogu stores. One odpowiadają za zarządzanie danymi i są związane z użytą biblioteką RefluxJS. Dzięki temu, iż dane mogą być modyfikowane tylko po wystąpieniu akcji, co jest zgodne z wzorcem Flux, dużo łatwiej jest je kontrolować i zapobiegać pojawianiu się błędów, niż w przypadku globalnych stanów, zmienianych w różnych miejscach aplikacji.

Zdecydowana większość komponentów znajduje się w katalogu *components*. Zostały one pogrupowane według ich funkcjonalności, chociaż niektóre są wykorzystywane w więcej niż jednym miejscu. Możliwość tworzenia wielokrotnie używanych modułów jest jedną z dobrych cech biblioteki React. Część z komponentów jest zwykłymi funkcjami (ang. React Stateless Functional Component), natomiast inne to klasy, dziedziczące po React.Component.

Wszystkie komponenty posiadają krótki opis swojej funkcjonalności wewnątrz pliku z kodem źródłowym.

8. Testowanie

Większość zasobów została przetestowana przy pomocy narzędzia Postman. Jest to wtyczka do przeglądarki Google Chrome, która pozwala tworzyć zestawy testowych zapytań oraz badać rezultat ich przebiegu. Dzięki temu możliwe jest zdefiniowanie parametrów oczekiwanych na wyjściu i porównywanie uch z otrzymanymi wartościami.

Tak duże pokrycie testami zapewnia bezpieczne wprowadzanie zmian na serwerze, gdyż każda modyfikacja może być następnie sprawdzona przy pomocy testów. Po najnowszych poprawkach wszystkie testy zakończyły się powodzeniem.

Testy klienta są wykonane przy pomocy narzędzi Mocha, Enzyme oraz Chai. Polegają one na renderowaniu poszczególnych komponentów aplikacji i sprawdzaniu, czy są one zgodne z oczekiwanymi dla podanych na wejściu danych. W ten sposób można np. sprawdzić, czy na liście użytkowników wyświetlają się wszystkie pozycje oraz czy menu posiada poprawne przyciski. Ze względu na rozbudowaną strukturę aplikacji nie jest możliwe pokrycie jej testami w 100% objętości kodu. Wybrane zostały więc pewne jej fragmenty.

Testy klienta można przeprowadzić, wywołując komendę:

```
npm test
```

w katalogu *client/* .

9. Wykorzystanie programowania aspektowego

W programie zostało wykorzystane programowanie aspektowe przy pomocy języka AspectJ. Pozwala on modyfikować działanie aplikacji bez ingerencji w główną część kodu, a jedynie poprzez wykonywanie dodatkowych operacji w tzw. punktach przecięcia.

Serwer systemu WebTicket używa czterech stworzonych po napisaniu właściwego kodu aspektów. Znajdują się one w pakiecie *wtaspect*. Zostały one opisane w tabeli 2.

Pliki, o których mowa w tej tabeli znajdują się w katalogu *logs* na serwerze i można je odczytać po uzyskaniu dostępu do serwera np. przez SSH.

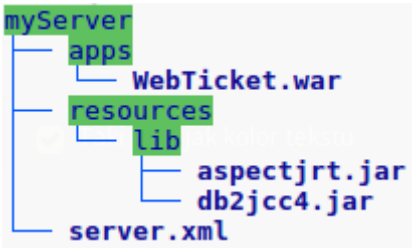
Aspekt	Opis
Authorization	Punkt przecięcia jest ustawiony na moment w którym wykonuje się metoda *login()* z klasy *api.Auth*. Aspekt jest stosowany po wykonaniu tej metody i loguje do pliku *wt_auth_details.log* wszystkie próby logowania wraz z datą, rezultatem, adresem e-mail oraz IP, z którego nastąpiła próba logowania.
Forbidden	Punkt przecięcia znajduje się wewnątrz wykonywania dowolnej metody wszystkich klas z pakietu *api.** z wyjątkiem *AbstractEndpoint*, wówczas gdy zostanie zgłoszony wyjątek *ForbiddenRequest*, czyli w sytuacji nieuprawnionego żądania zasobu. Przed zgłoszeniem tego wyjątku dzięki aspektowi sytuacja zostanie zalogowana do pliku *wc_forbidden.log* wraz z informacją, w jakiej metodzie ta sytuacja zaistniała. Jeżeli została wywołana przez zalogowanego użytkownika, w logu znajdzie się również informacja o nim.
Rate	Punkt przecięcia jest ustawiony na aktywność oceny zgłoszenia przez użytkownika, czyli wywołanie metody *rate()* z klasy *api.TicketApi*. Aspekt ten zawiera dwie porady. Pierwsza z nich przed punktem przecięcia loguje do *pliku wc_rating.log* informacje o tym, kto oceniał kogo, a także jakie zgłoszenie zostało ocenione i na ile. Wewnątrz tej porady jest używane połączenie z bazą danych, aby pobrać dodatkowe informacje o ocenianym zgłoszeniu. Druga z porad aktywuje się po punkcie przecięcia i modyfikuje przesyłaną do użytkownika odpowiedź. W zależności od wystawionej oceny pracownikowi zostanie dodany jeden z trzech komunikatów.
Registration	Punkt przecięcia jest ustawiony na wywołanie metody *registerUser()* z klasy *api.UserApi*. Dzięki aspektowi można logować, kiedy i jacy użytkownicy zostali poprawnie zarejestrowani. Dane to zostają zapisane do pliku *wc_register.log*. Aspekt korzysta z kodu odpowiedzi, pochodzącego z wykonania metody, którą obserwuje.

Tabela 3 Opis aspektów użytych w aplikacji WebTicket

10. Wdrożenie

Program może być uruchomiony w środowisku WAS Libery zarówno na zwykłym serwerze jak również w chmurze. Aby spełnić założone wymagania projektowe został on wdrożony w chmurze obliczeniowej Bluemix przy pomocy konsolowego narzędzia Cloud Foundry CLI.

Aby poprawnie opublikować aplikację, należy stworzyć plik WAR z jej zawartością. Przed uruchomieniem w chmurze Bluemix serwera z programem WebTicket należy przygotować biblioteki: aspectjrt.jar do obsługi aspektów oraz db2jcc4.jar do połączenia z bazą danych dashDb. Struktura plików i katalogów przed publikacją powinna wyglądać podobnie, jak na schemacie 1.



Schemat 1Struktura katalogów i plików, potrzebna do wysłania projektu na serwer

Po wcześniejszym zalogowaniu się wdrożenie aplikacji może zostać dokonane przy pomocy komendy:

```
cf push webticket -m 512M -p myServer -b liberty-for-java
```

OdD chwili wdrożenia aplikacja jest dostępna pod adresem:

http://webticket.mybluemix.net

W taki sposób program może być wdrożony na dowolnym koncie chmury Bluemix. Można zmienić nazwę projektu, będącą również nazwą subdomeny w domenie *.mybluemix.net, a także ilość przydzielonej pamięci czy nazwę katalogu, w którym znajduje się publikowany program.

Należy pamiętać, że publikacja aplikacji wymazuje bazę danych i tworzy surową instancję wraz z zapisanymi początkowymi ustawieniami. Dlatego przed ewentualną ponowną publikacją należy wykonać kopię zapasową bazy danych.

Wersja opublikowana w tym miejscu jest jedynie demonstracją programu. W przypadku wdrożenia w pełni produkcyjnego właściciel powinien zapewnić obsługę bezpiecznego protokołu HTTPS, aby nie dopuścić do możliwości podsłuchiwania danych w komunikacji klient – serwer.

Kod programu znajduje się w repozytorium, dostępnym pod adresem:

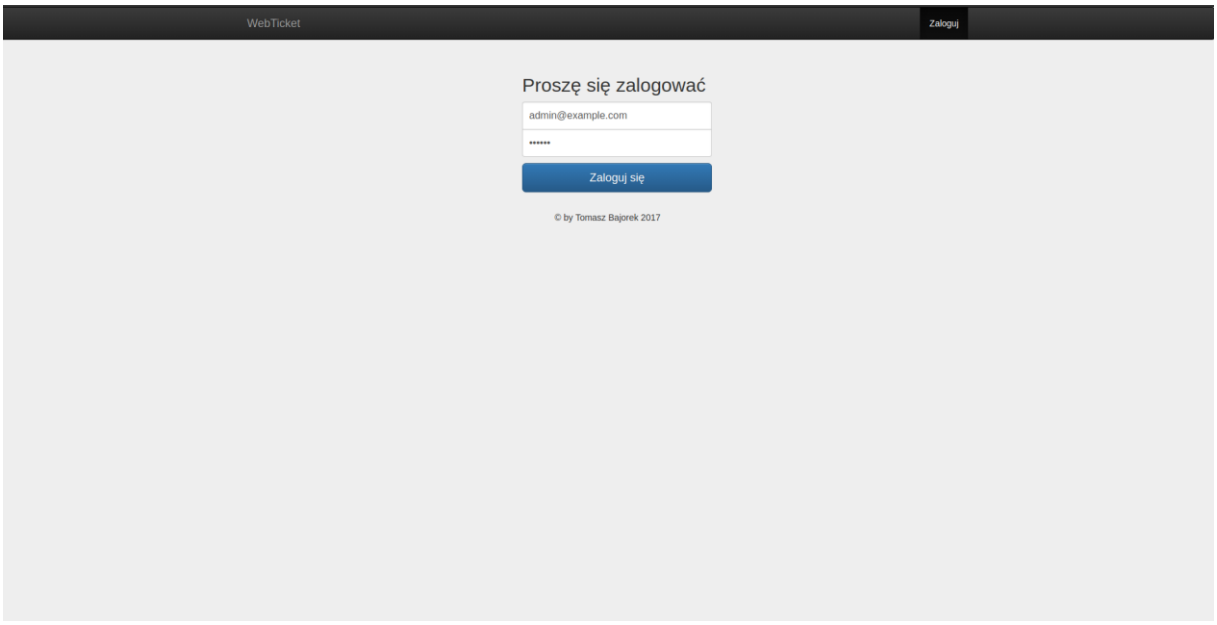
https://github.com/tbajorek/WebTicket

11. Instrukcja użytkownika

Przed przeczytaniem tego rozdziału warto zapoznać się z szczegółowym opisem funkcjonalności programu WebTicket, opisanymi w rozdziale 5. Przedstawia on dokładny opis wszystkich możliwości systemu oraz informacji, jak je używać.

Aplikacja posiada utworzone konto administratora. Są one następujące:

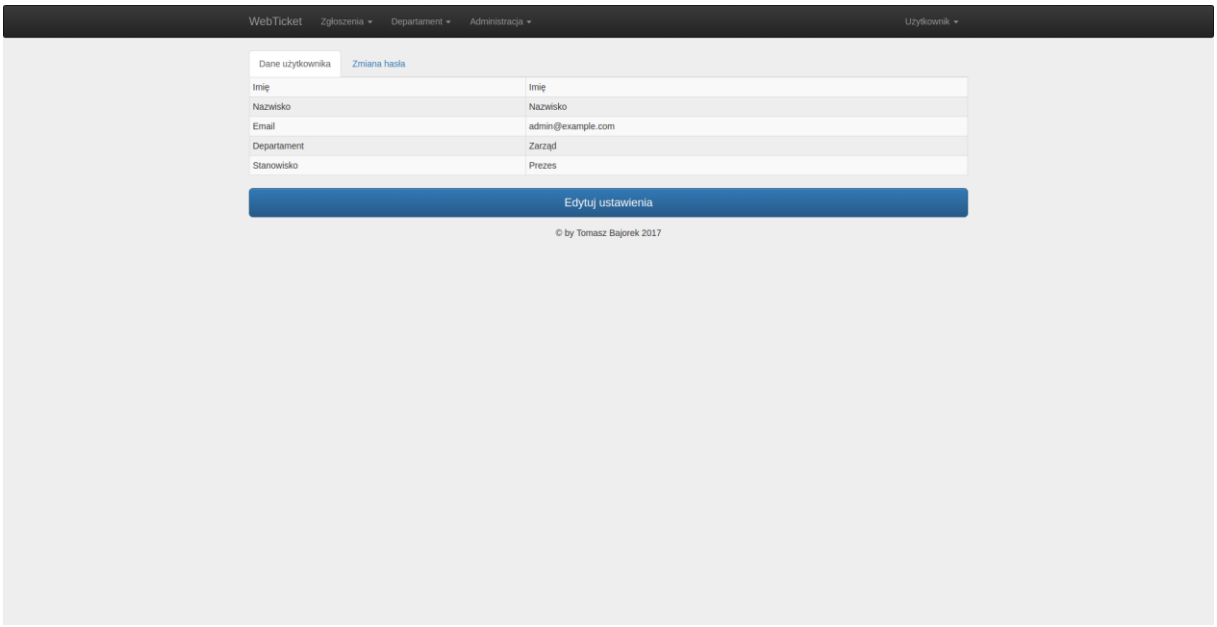
- e-mail: admin@example.com
- hasło: qwerty



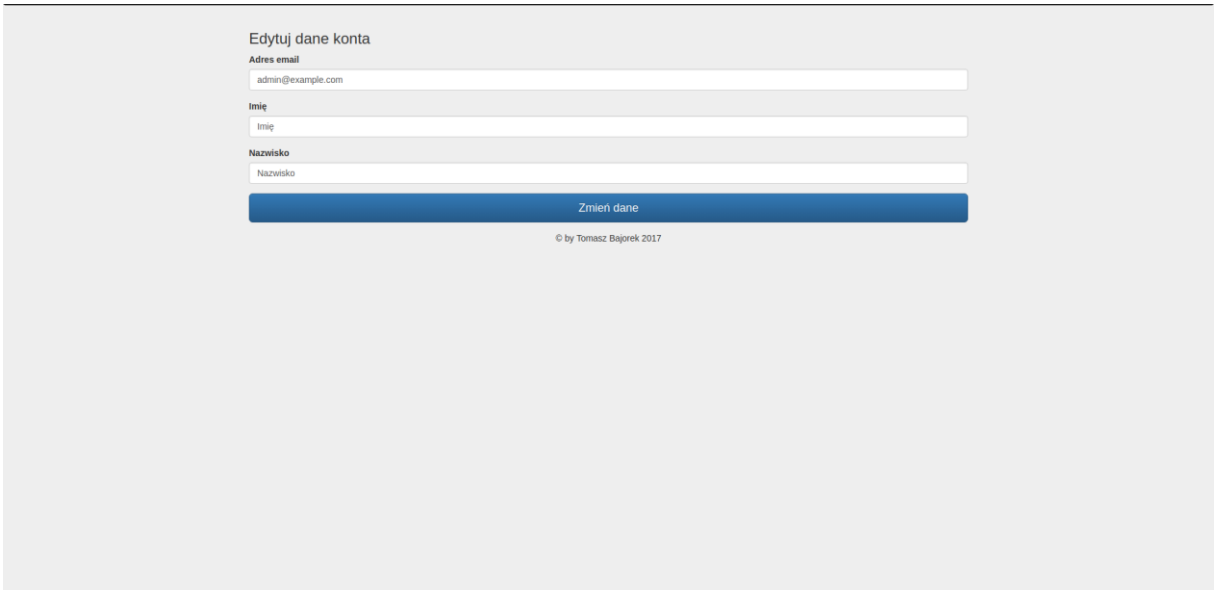
Rysunek 1Ekran logowania aplikacji WebTicket

Czasem podczas pierwszego logowania zapytanie do serwera może trwać dłużej ze względu na operacje, jakie dokonują się w bazie danych podczas jej pierwszego użycia.

Po uruchomieniu aplikacji jej właściciel powinien się zalogować przy ich użyciu do systemu WebTicket i jak najszybciej zmienić je na własne. Początkowe konto administratora ma przypisaną pozycję „Prezes” oraz departament „Zarząd”.

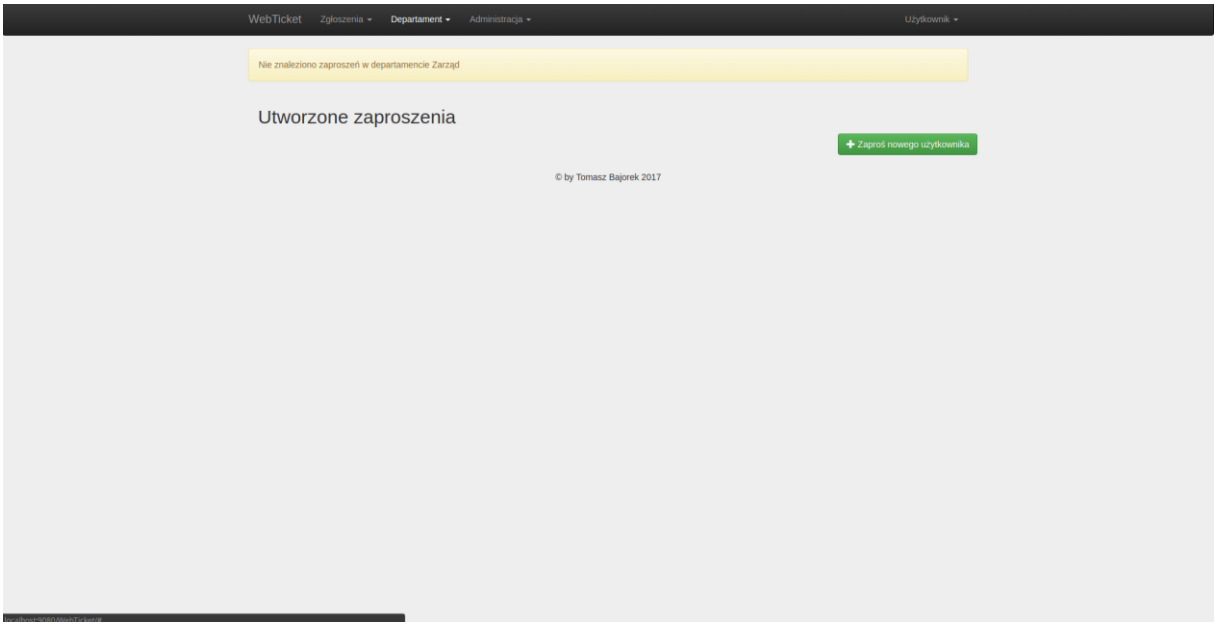


Rysunek 2 Podgląd ustawień konta. W drugiej zakładce jest formularz do zmiany hasła

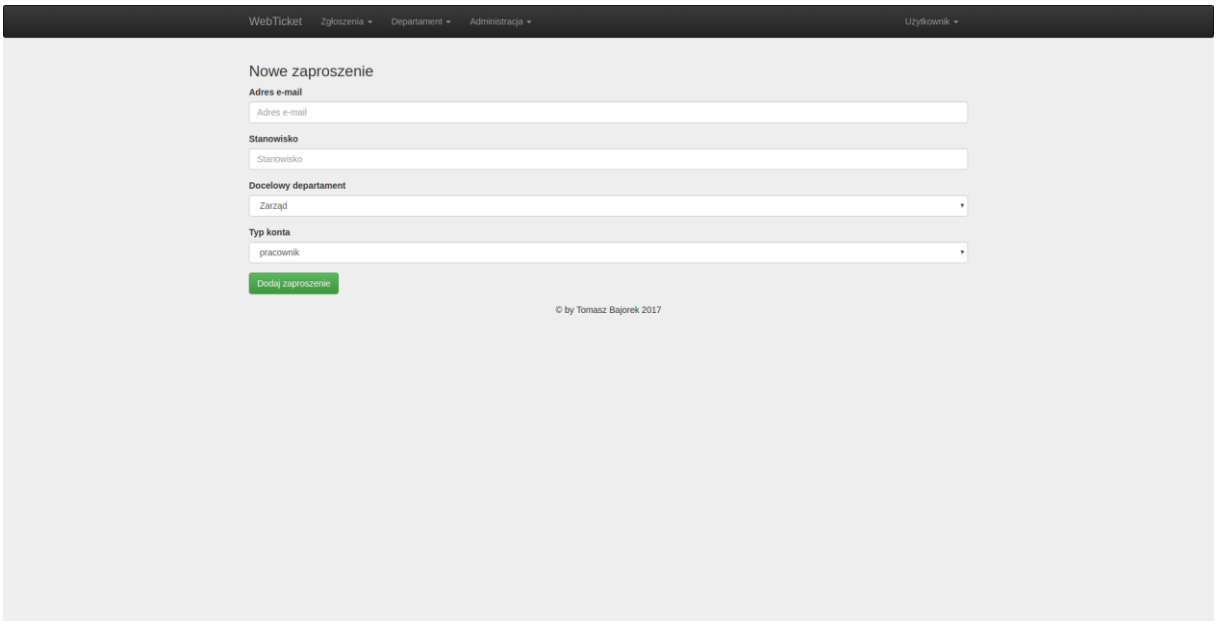


Rysunek 3 Formularz zmiany ustawień kotna zalogowanego użytkownika

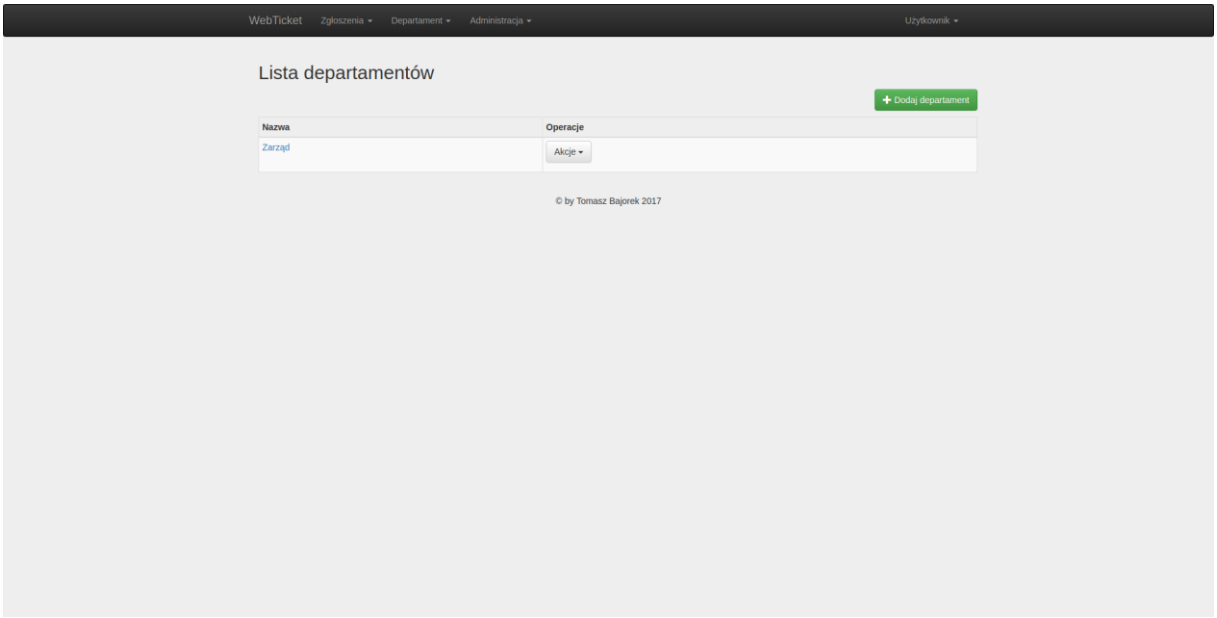
Jedną z pierwszych czynności będzie prawdopodobnie zaproszenie nowych użytkowników. Po zatwierdzeniu nowego zaproszenia administrator wysyła im link. Wszystkie dane podane przy zapraszaniu pracownika zostaną przypisane do jego konta w momencie rejestracji.



Rysunek 4 Lista zaproszeń do departamentu; administrator może zapraszać do dowolnego departamentu, wybierając docelowy z listy

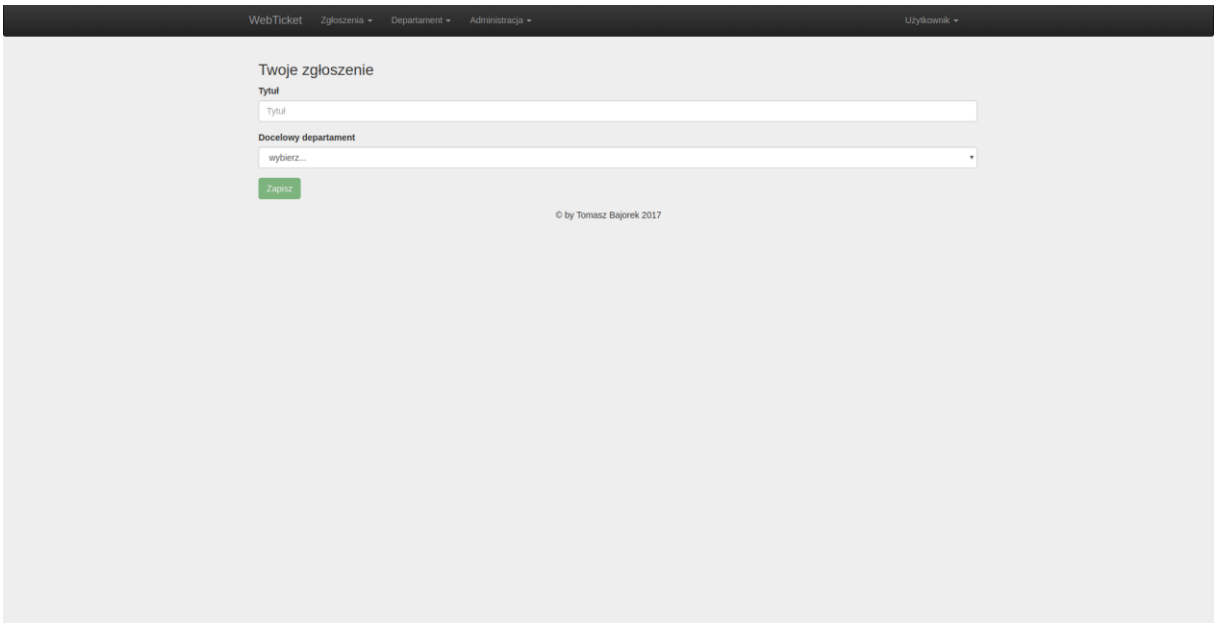


Rysunek 5 Formularz dodawania nowego zaproszenia



Rysunek 6 Lista departamentów; administrator może dodać nowy

Dodawanie zgłoszenia jest dwuetapowe. Najpierw należy podać tytuł i docelowy departament.



Rysunek 7 Formularz dodawania nowego zgłoszenia

Zgłoszenie może istnieć bez początkowej wiadomości, może być przyjęte, a także zamknięte. Aby mogło zostać ocenione, musi być najpierw przyjęte przez pracownika z docelowego departamentu, a potem zamknięte przez autora.

12. Podsumowanie

Projekt został wykonany zgodnie z początkowymi założeniami. Spełnia postawione przez prowadzącego przedmiot wymagania, jak również funkcjonalności, zaproponowane przez piszącego aplikację. Brak implementacji jednej z funkcjonalności serwera w kliencie nie wpływa zdaniem autora na przydatność całej aplikacji w zastosowaniu jej do wspomagania działalności firmy.

Wykonanie projektu pozwoliło zapoznać się w praktyce z licznymi technologiami informatycznymi, opisanymi w rozdziale 4. Jest to cenna wiedza, która może się przydać zarówno w pracy zawodowej, jak też w dalszej studenckiej edukacji.

Projekt został zaplanowany, zrealizowany oraz przetestowany, co pozwoliło zastosować w praktyce zdobytą wcześniej wiedzę na temat procesu wytwarzania oprogramowania.

Spis treści