



Przemysław Kasprzyk

1 EF-DU-A

132758

Email: p.kasprzyk2008@gmail.com

Przedmiot: Aplikacje internetowe

Wersja: 3

Data: 10.01.2017

Rodzaj zajęć: projekt

Prowadzący: dr inż. Tomasz Rak

**System Zarządzania Dokumentami**

1. Cel projektu

Celem projektu jest utworzenie aplikacji internetowej służącej do zarządzania dokumentami. Aplikacja przeznaczona jest do użytku z poziomu przeglądarki internetowej.

2. Wymagania Funkcjonalne

System obsługiwany jest przez 3 typy użytkowników - administratora, moderatora, zalogowanego i niezalogowanego.

* System powinien obsługiwać następujące funkcjonalności
* Logowanie/rejestrację
* Dodawanie i wersjonowanie dokumentów
* Pobieranie dokumentów
* Proste szukanie dokumentów oraz przeszukiwanie sprecyzowane
* Wprowadzenie daty od kiedy dokument ba być opublikowany
* System ocen
* System komentarzy
* Możliwość dodawania słów kluczowych i opisów do dokumentów przez ich autora
* Możliwość ustawienia moderatorów sprawdzających treści dokumentu przed publikacją
* Możliwość ustawienia użytkownika zaufanego, nie potrzebującego moderacji
* Dodawanie zdjęć okładek jako plik lub za pomocą hiperłącza do pliku w sieci
* Możliwość otrzymywania powiadomień mailowych.
* Możliwość udostępniania dokumentu przez email z hiperłączem, email z załącznikiem, bądź wygenerowanie hiperłącza do pliku.
* Wstrzymanie dostępu użytkownikom naruszającym regulamin serwisu.
* Otrzymywanie powiadomień o wstrzymaniu dostępu, publikacji dokumentu, lub akcjach

Podział funkcjonalności użytkowników

* Niezalogowany użytkownik - ma możliwość przeglądania i udostępniania hiperłączy jedynie do dokumentów widocznych publicznie, może się zarejestrować, lub zalogować.
* Zalogowany użytkownik - rozszerza możliwości niezalogowanego, ma dostęp do pełnej bazy dokumentów. Nie ma ograniczeń w funkcjach udostępniania. O ile jego dostęp nie jest wstrzymany, może dodawać, publikować, komentować i oceniać dokumenty. Może edytować swój profil.
* Moderator - rozszerza możliwości użytkownika niezalogowanego ma możliwość sprawdzenia treści dokumentów i zatwierdzania ich do publikacji, a także zawieszenia uprawnień użytkownika
* Administrator - ma uprawnienia moderatora, a ponad to ma możliwość zmiany uprawnień użytkowników.

1. Wymagania niefunkcjonalne

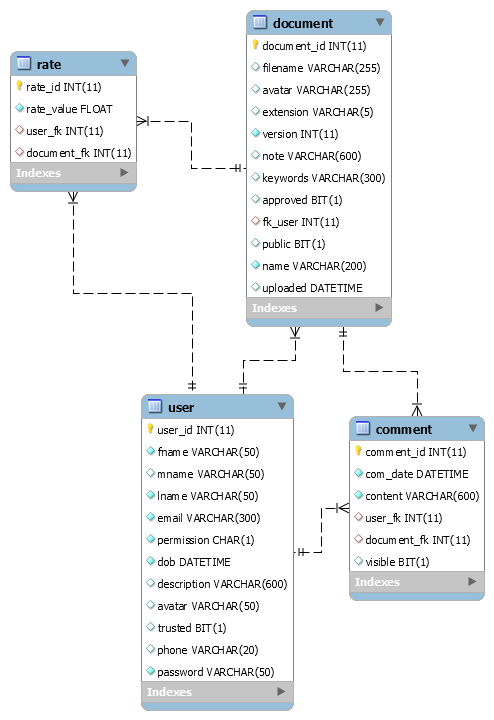
Warstwa serwerowa:

* Napisana w języku Java.
* Wykonana z użyciem frameworków Hibernate (wersja 4), Spring (wersja 4), Spring Data.
* Zastosowany system zarządzania zależnościami Apache Maven.
* Zastosowana implementacja serwera - Apache Tomcat/TomEE
* Relacyjna baza danych MySQL
* Pliki przechowywane bezpośrednio na serwerze (nie używana jest do przechowywania baza danych)
* System maszyny developerskiej - Windows 10

Warstwa prezentacji

* Napisana w języku TypeScript
* Wykonana z użyciem frameworka jQuery
* Bootstrap
* Zakładana kompatybilność z przeglądarkami Internet Explorer 11, Edge, Chrome 50, Firefox 49.

1. Diagram encji bazy danych



1. Struktura projektu

Projekt składa się z komponentu serwerowego i webowego. Komponent serwerowy składa się z 3 paczek - integration, domain i persistence. Dla paczek domain i persistence zastosowano domyślny modyfikator dostępu w miejscu gdzie nie ma zachodzić bezpośrednie połączenie pomiędzy warstwami aplikacji.

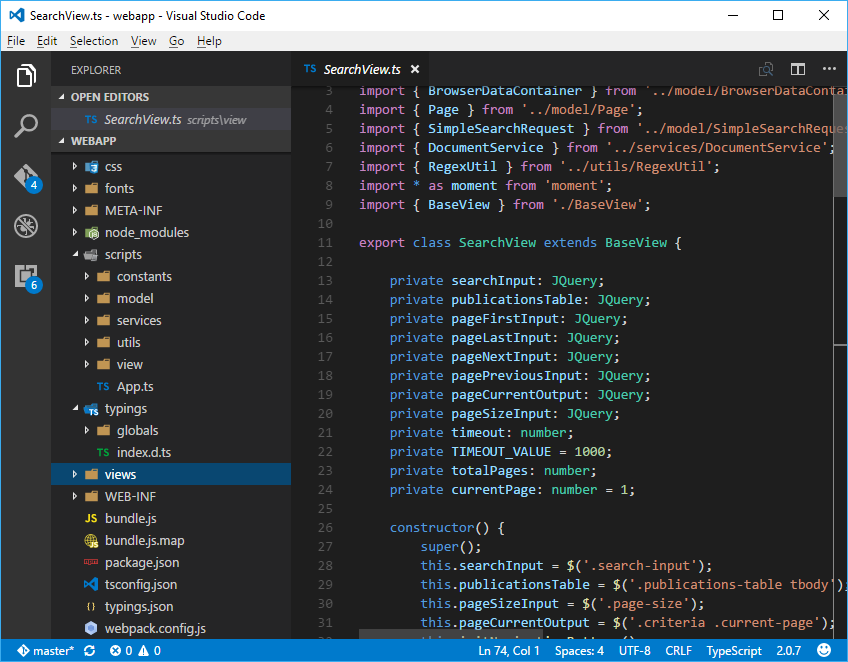
Integration - przechowuje elementy wspólne dla części serwerowej. Znajdują się tutaj elementy systemu zabezpieczeń aplikacji, konfiguracja projektu, a także interfejsy i klasy abstrakcyjne do wykorzystania w innych warstwach.

Persistence - zawiera modele JPA, repozytoria Spring Data JPA, a także serwisy których jedynym zadaniem jest zaciągnięcie danych.

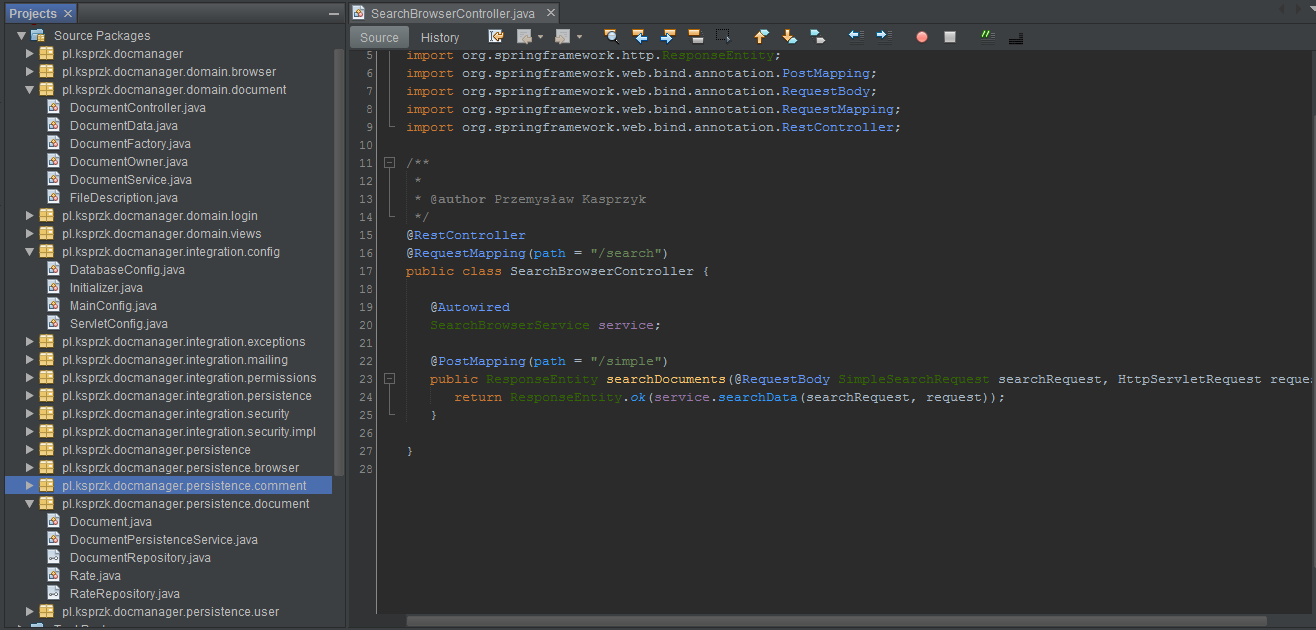
Domain - zawiera DTO, Kontrolery i mapping dla serwisów RESTowych, serwisy agregujące dane z serwisów paczki persistence, oraz fabryki przetwarzające JPA w DTO.

Komponent webowy został rozbity na klasy i interfejsy rozlokowane w katalogach model, service i view. Model zawiera interfejsy ułatwiające mapowanie DTO z serwera. Klasy z folderu service odpowiadają za wysyłanie i odbiór danych udostępnianych przez punkt dostępu serwera. Klasy z folderu view odpowiadają za zachowanie strony, oraz bindowanie danych pochodzących od użytkownika z serwisami.

Całość komponentu webowego uzupełniają widoki napisane w języku HTML, biblioteki JavaScript, zbiory styli CSS, i czcionek. Wszystkie utworzone w języku TypeScript skrypty są kompilowane a następnie składane do jednego dużego pliku bundle.js, celem ograniczenia ilości zapytań przeglądarki do serwera.



Rysunek 1 Widok struktury projekty od strony frontendu



Rysunek 2 Widok struktury projektu od strony backendu

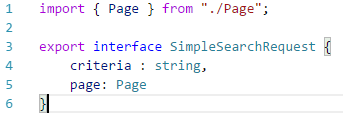
1. Wykonanie frontendu

Frontend wykonany jest modułowo. Jednym z bardziej rozbudowanych modułów jest mechanizm służący do wyszukiwania dokumentów na serwerze.

Pierwszym elementem struktury tego widoku jest serwis służący do komunikacji z serwerem znajdujący się w klasie BrowserService.



Warto odnotować iż nie jest konieczne by w oddzielnych plikach zamieszczać kolejne klasy, bądź interfejsy, jednakże stosowanie tej zasady pomaga w utrzymaniu schludnego kodu, który jest łatwiejszy do późniejszej rozbudowy, bądź debuggowania. Eksport może też dotyczyć elementów dołączanych poprzez użycie słów kluczowych import bądź require, jednakże wprowadza to ryzyko, że do kodu zostaną wprowadzone elementy bardziej przypominające strukturalne elementy JavaScript, co również źle może wpłynąć na skalowalność takiego kodu. Sam serwis składa się z jednej metody statycznej udostępniającej żądanie Ajax wykonane przez bibliotekę jQuery. Dane wprowadzane jako argument metody mają przypisany interfejs SimpleSearchRequest. Niewątpliwą zaletą języka TypeScript jest możliwość ustalenia interfejsu, dzięki czemu można się upewnić co do zawartości obiektu, co jest trudniejsze w przypadku JavaScriptu. Dodatkowym atutem tego rozwiązania jest rozszerzona pomoc środowisk programistycznych, przewidujących co można zrobić z danym obiektem.

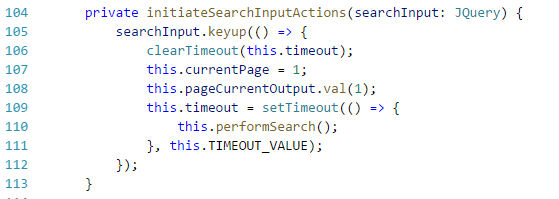


Interfejsy przechowujące dane przechowywane są w katalogu model. Ich podstawową funkcją jest dbanie o poprawne nazewnictwo elementów, zapobiegające problemom z błędami, gdzie przez błąd w pisowni, lub inną pomyłkę na serwer zostaną wysłane niekompletne dane, lub zostaną błędnie odczytane z żądania.

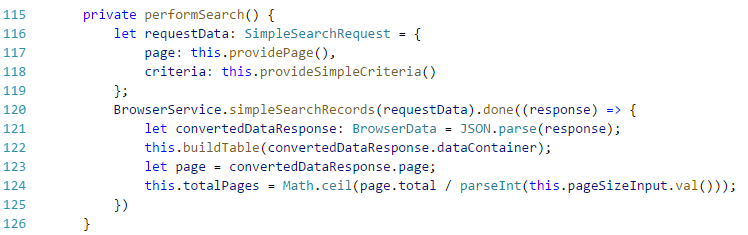
Serwisy wykorzystywane są przez komponenty view, odpowiadające za wyświetlenie danych użytkownikom.



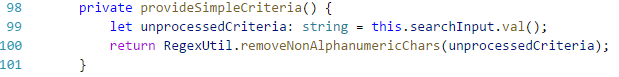
Język pozwala na dziedziczenie klas, co bywa przydatne jeśli potrzebne jest upewnienie się co do funkcjonalności danego elementu bez tworzenia niepotrzebnych warunków, bądź gdzie wymagane jest powielenie części kodu. W przypadku klas sterujących widokami, zachodzi dziedziczenie z klasy BaseView, która odpowiada za sterowanie elementami powtarzającymi się na każdej stronie aplikacji, takich jak panel nawigacji, kontrolka wylogowania. Do części bibliotek istnieją w sieci pliki d.ts, które pozwalają na korzystanie z typowania używając bibliotek napisanych w JavaScripcie, jak na przykład pola będące elementami typu JQuery.

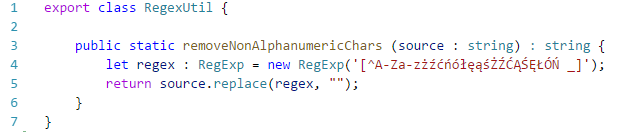


Proces wyszukiwania dokumentów wywoływany jest poprzez upływ czasu od ostatniego wpisanego znaku do pola wyszukiwarki. Pozwala to na pominięcie użycia przycisku szukania i przypomina w działaniu nowocześniejsze wyszukiwarki internetowe.



Metoda wyszukiwania jest asynchroniczna, co sprawia że użytkownik nie ma świadomości działań zachodzących w tle, oraz nie jest widoczne zamrożenie okna przeglądarki, co sprawia iż dalsze kryteria wyszukiwania mogą być wpisywane, gdy poprzednie są jeszcze w fazie zaczytywania.

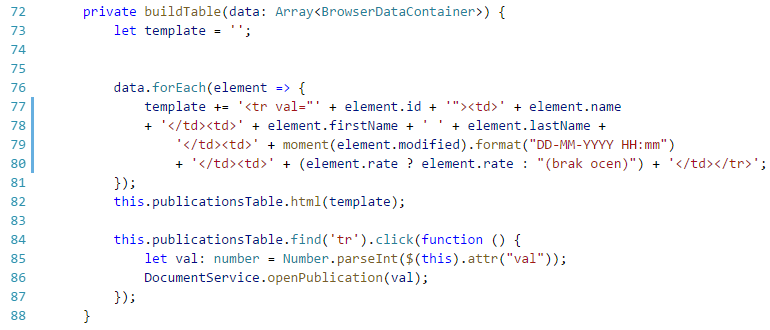




Aplikacja została wzbogacona o różne dodatkowe narzędzia nie mieszczące się w kryteriach żadnej warstwy. Zostały one zapisane w katalogu Utils. Przedstawiona klasa RegexUtil odpowiada za obróbkę łańcuchów znakowych do akceptowalnej formy. Zamieszczona metoda usuwa z łańcucha znakowego wszystkie znaki, które nie powinny być z pewnych powodów użyte do wyszukiwania.



Podział wyników na strony wymusił implementację dynamicznego sterowania przełączania pomiędzy stronami z wynikami.



Finalnie zbudowane zostało ciało tabeli przechowującej wyniki. Nie została użyta żadna biblioteka do tworzenia wzorców widoków, ani też nie ma elementów odpowiadających za bindowanie danych, więc użyte zostały jedynie funkcje JQuery podmieniające kod HTML na stronie i dodające zdarzenie kliknięcia.

Większość opisanych fragmentów kodu pokazuje TypeScript od jego dobrej strony, jednakże w chwili obecnej język ten ma mnóstwo wad, nad którymi trzeba się poważnie zastanowić zanim wdroży się jakikolwiek element tego języka do projektu. Problemem jest skonfigurowanie projektu, gdyż istnieje wiele narzędzi kompilujących i składających projekt do plików js, których obsługa na początku wymaga dodatkowego nakładu pracy. Debuggery nie zawsze dobrze radzą sobie z kodem TypeScript, a dodatkowo nie jest możliwe wykonywanie elementów takiego kodu bezpośrednio z przeglądarki bez wcześniejszego otwierania wersji skompilowanej. Nie do każdej biblioteki można znaleźć odpowiedni plik d.ts, więc nieraz wymagane jest tworzenie ich od podstaw, a dodatkowo nawet w znalezionych w oficjalnych repozytoriach plikach znajdują się błędy dyskwalifikujące taki plik z użycia. Pojawia się problem z zasięgami, gdzie w funkcji JavaScript słowo kluczowe „this” oznacza co innego niż w TypeScript. Czasami istnieją też problemy z odpowiednim dobraniem rzutowanego typu. Projekt zawiera mechanizm wysyłania plików, którego mała część została napisana w natywnym JavaScripcie, gdyż było to szybsze i bardziej pewne od użycia typowania. Element z tego samego żądania mógł być rzutowany na kilka różnych interfejsów co sprawiało, iż zbyt czasochłonny okazał się jego wybór, gdyż sugerowany przez środowisko interfejs był nieprawidłowy do tego konkretnego zadania. Z programowania w TypeScripcie skorzystają najwięcej twórcy dużych projektów wymagających skalowalności i długotrwałego prowadzenia wsparcia technicznego, dla mniejszych projektów jego wprowadzenie może być zbyteczne, a nawet niewskazane.