Uniwersytet Gdański
Wydział Matematyki, Fizyki I Informatyki
Instytut Informatyki
Aplikacje Internetowe i Bazy Danych
Studia dzienne

Projekt Zespołowy – dokumentacja

Edytor tekstu on-line do współpracy programistycznej

Skład osobowy zespołu:

Lipkowski Paweł
Mazepa Mariusz (lider)
Zakrzewski Michał

Gdańsk 2018

SPIS TREŚCI

ROZDZIAŁ 1: CEL PRACY

Z góry założonym celem pracy podczas projektowania oraz tworzenia przez nas aplikacji webowej było stworzenie edytora tekstu on-line do współpracy dla programistów. Od początku powstawania coraz to większej ilości serwisów tego typu zastanawiało nas, na jakiej zasadzie one funkcjonują, dlaczego tak, a nie inaczej, czy można coś zrobić lepiej, aby usprawnić działanie takiego edytora. Nie od wczoraj wiadomo, że najlepszym sposobem na zrozumienie czegoś jest działanie, w związku z czym postanowiliśmy wykorzystać okazję, jaka nadarzyła się na przedmiocie Projekt Zespołowy i spróbować swoich sił w przedsięwzięciu, które rozwieje nasze wszelkie wątpliwości w tej kwestii oraz przede wszystkim wiele nas nauczy o działaniu edytorów w sieci i różnorodności rozwiązań zależnie od preferencji technicznych, wizualnych, a nawet sprzętowych.

ROZDZIAŁ 2: PODOBNE ROZWIĄZANIA

W sieci Internet krąży wiele różnorodnych rozwiązań, na bazie których narodził się pomysł o stworzeniu własnej wersji edytora tekstu on-line. Najpopularniejsze z nich – według nas – na których wzorowaliśmy się najbardziej, przedstawiają się następująco:

L.P.	Nazwa rozwiązania	Strona internetowa	Współpraca w czasie rzeczywistym	Kompilacja tekstu
1	Collabedit	collabedit.com	Tak	Nie
2	Ideone	ideone.com	Nie	Tak
3	WriteURL	writeurl.com	Tak	Nie

Z powyższej listy wybranych podobnych rozwiązań tylko dwa rozwiązania zawierają współpracę w czasie rzeczywistym, a trzecie posiada wbudowany kompilator kodu źródłowego.

ROZDZIAŁ 3: PROCES TWORZENIA OPROGRAMOWANIA

Do procesu tworzenia niniejszego systemu użyliśmy znanego z Inżynierii Oprogramowania **modelu kaskadowego**.

ROZDZIAŁ 3.1: PLANOWANIE SYSTEMU – SPECYFIKACJA WYMAGAŃ

Podczas projektowania realizowanego systemu postawiliśmy przed sobą następujące **wymagania funkcjonalne i niefunkcjonalne**:

1. Wymagania funkcjonalne
    a. Tylko dla administratora i częściowo dla moderatora
        i. Zarządzanie wszystkimi tabelami zawartymi w bazie danych – CRUD (administrator)
        ii. Zarządzanie wszystkimi tabelami zawartymi w bazie danych – CRUD – oprócz kont użytkowników i dostępnych ról (moderator)
    b. Dla wszystkich zalogowanych użytkowników serwisu
        i. Wprowadzanie i edycja tekstu
        ii. Rozpoznawanie (kolorowanie) składni zgodnie z wybranym przez użytkownika preferowanym rodzajem tekstu (językiem programowania)
        iii. Przesyłanie pliku z dysku użytkownika jako alternatywa do wprowadzania tekstu ręcznie lub na zasadzie „kopiuj-wklej”
        iv. Praca grupowa – wieloosobowa współpraca kilku użytkowników w obrębie grupy na tym samym pliku
        v. Komunikator dla użytkowników pracujących jednocześnie na pliku w obrębie grupy, do której razem przynależą
    c. Dla niezalogowanych użytkowników serwisu
        i. Rejestracja i logowanie – przechowywanie danych użytkowników w zewnętrznej bazie danych

2. Wymagania niefunkcjonalne
    a. Przejrzysty i intuicyjny interfejs
    b. Wygodna nawigacja w nagłówku serwisu pojawiającym się na każdej stronie w obrębie serwisu
    c. Uniwersalność i mobilność względem najpopularniejszych przeglądarek internetowych

Powyższa lista w pierwotnym wybrzmieniu różniła się od swej obecnej formy, bowiem wraz z rozwojem aplikacji rodziły się coraz to nowsze pomysły, czy rozwiązania, jak również stawaliśmy przed różnymi ograniczeniami, które spowalniały prace bądź czasem ją nawet hamowały w takim stopniu, że byliśmy zmuszeni zrezygnować z niektórych części funkcjonalnych.

ROZDZIAŁ 3.2: ANALIZA SYSTEMU – OGRANICZENIA

Głównym ograniczeniem, które najbardziej dało nam się we znaki, jest ograniczenie sprzętowe. Niedysponowanie dobrej klasy sprzętem odbiło się na prędkości działania implementowanego przez nas systemu oraz licznych problemach z połączeniem, przede wszystkim przez konieczność korzystania z zewnętrznych, darmowych serwerów, które mimo udostępniania dość sporej przestrzeni dyskowej i pamięciowej, jak na drobną aplikację webową, okazały się niewystarczające wobec wymagań naszego rozwiązania.

ROZDZIAŁ 3.3: PROJEKT SYSTEMU

Projektowanie aplikacji odbywało się w kilku fazach. Najważniejszymi z nich oraz najbardziej przydatnymi – po określeniu wymagań funkcjonalnych i niefunkcjonalnych – okazały się schematy, diagramy oraz z góry jasno założone technologie wykonania, o których kilkoma zdaniami wypowiemy się w dalszej części dokumentacji.

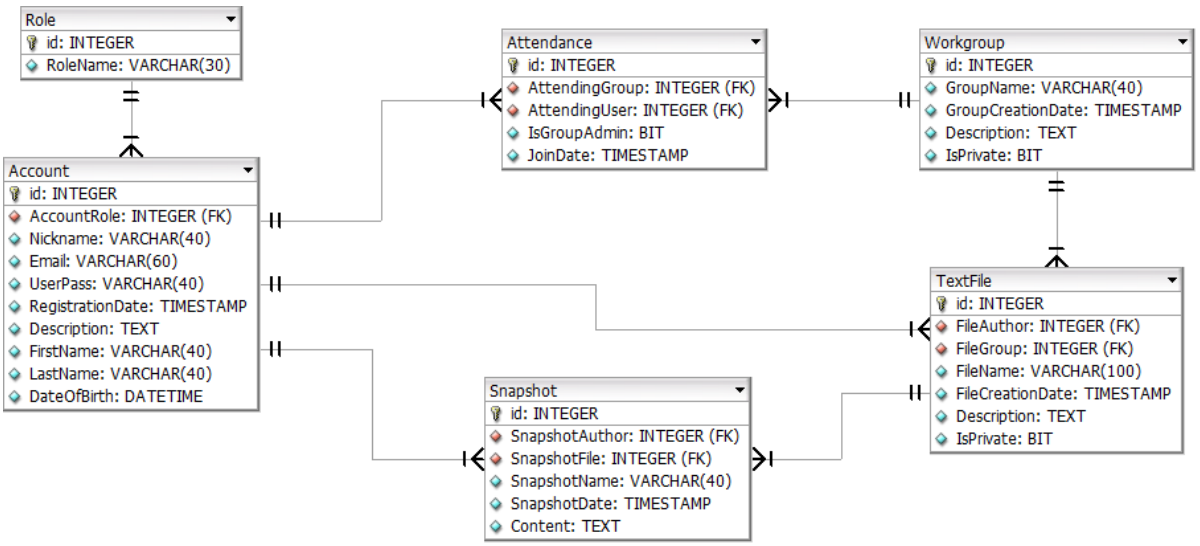
ROZDZIAŁ 3.3.1: SCHEMATY I DIAGRAMY

W początkowych fazach projektowania systemu zwizualizowaliśmy zarys naszej aplikacji za pomocą kilku schematów i diagramów, które ułatwiły nam wyobrażenie sobie koniecznego nakładu pracy, a co za tym idzie wstępny podział sprawowanych funkcji i obowiązków. Projekt naszego serwisu bazował przede wszystkim na:

1. **Schemacie ERD** wizualizującym niezbędne tabele i połączenia między nimi w bazie danych
2. **Diagramie klas**, wstępnie tożsamym ze schematem ERD, jednak z czasem rozwoju aplikacji uwydatniającym coraz to nowsze klasy niezbędne do funkcjonowania systemu
3. **Diagramie przypadków użycia**, który z założenia miał nam pomóc wyobrazić sobie wszelki możliwy kontakt użytkownika z serwisem, aby łatwiej było rozpisać i przydzielić wszelkie konieczne do wykonania zadania najlepiej znającym się na rzeczy osobom

Wspomniane diagramy i schematy utworzone podczas projektowania systemu przedstawiają się w sposób następujący:

Schemat ERD (diagram związków encji) bazy danych //WYMAGA DOPRACOWANIA

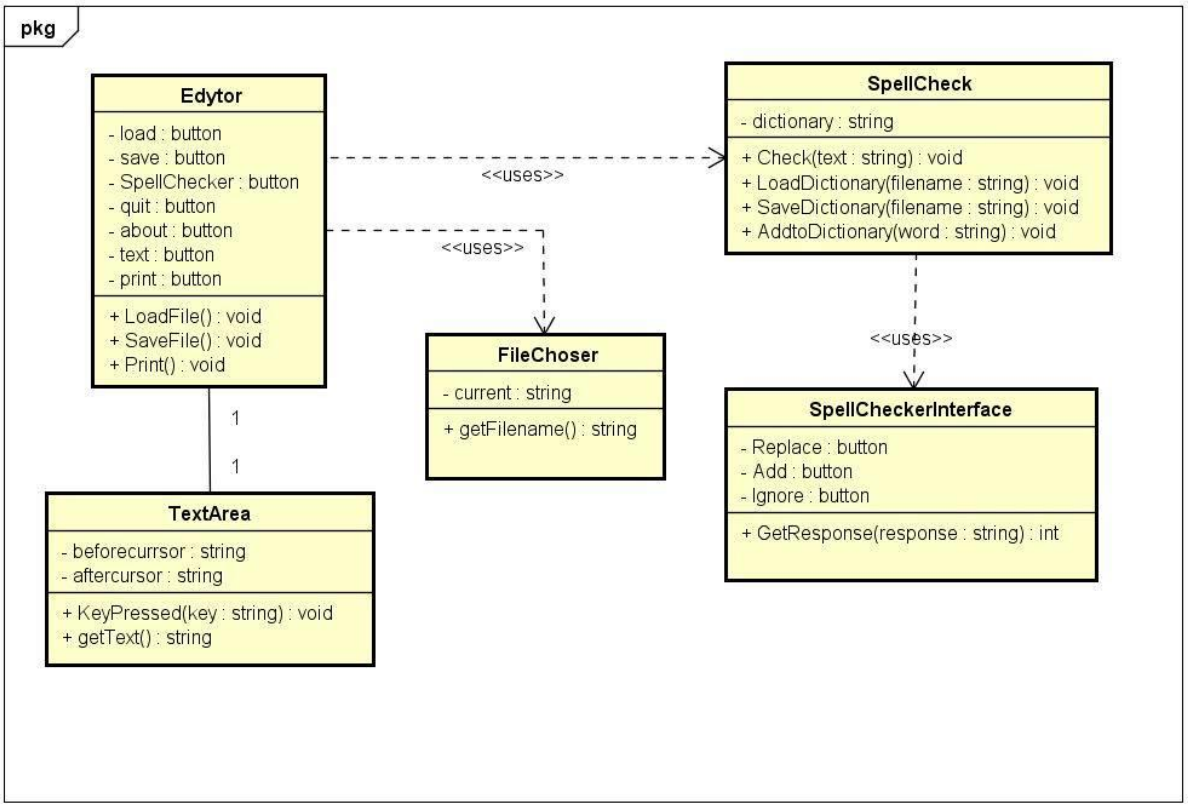


Rys.1: Schemat ERD (diagram związków encji) bazy danych

Dzięki schematowi ERD przedstawionemu na grafice na poprzedniej stronie (Rys.1) byliśmy w stanie określić, jakie są najważniejsze klasy, których potrzebujemy do stworzenia CRUDa do panelu administracyjnego.

Diagram klas

//WYMAGA DOPRACOWANIA

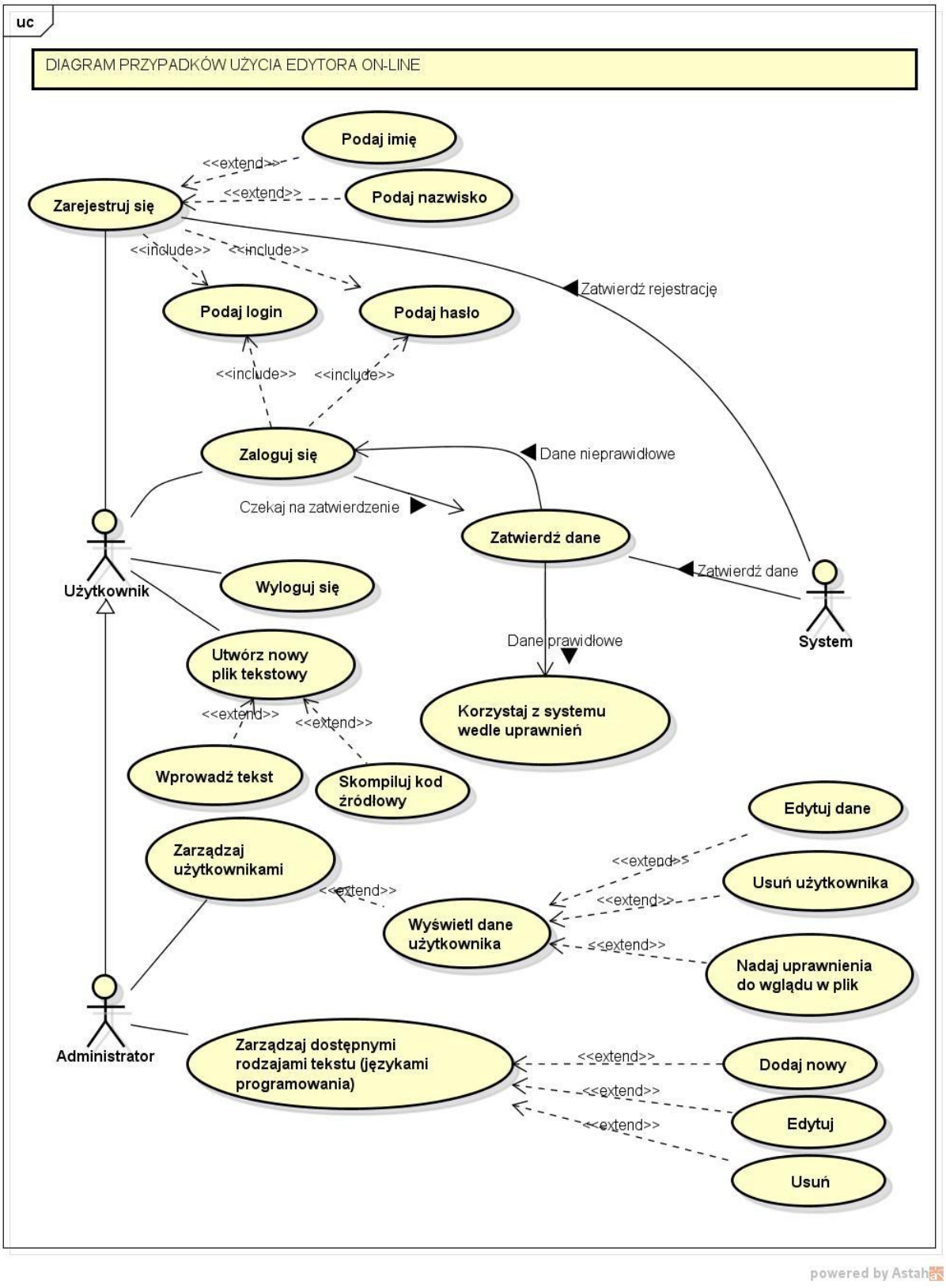


Rys.2: Diagram klas

Powyższy diagram klas (Rys.2) pozwolił zwizualizować sobie w głowie niezbędne klasy oraz zawarte w nich metody, bez których funkcjonowanie aplikacji byłoby niemożliwe.

Diagram przypadków użycia

//WYMAGA DOPRACOWANIA



Rys.3: Diagram przypadków użycia

Wszelkie przypadki użycia widoczne są na powyższym diagramie (Rys.3).

ROZDZIAŁ 3.3.2: TECHNOLOGIE WYKONANIA

Projektując system doszliśmy wspólnie do wniosku, że najprościej będzie nam się pracowało w obiektowym środowisku programowania udostępnianym przez twórców języka **Java** z wykorzystaniem możliwości **Java Platform Enterprise Edition** (JEE). Wyborem kierowały przede wszystkim znajomość wspomnianych technologii oraz ich wieloplatformowość. Sprzyjała temu również kompatybilność z narzędziem automatyzującym budowę oprogramowania na platformę Java, jakim jest **Apache Maven**.

Stronę klienta obsługuje znany wszystkim nawet początkującym webmasterom i webdeveloperom skryptowy język programowania **JavaScript**. Strukturalną częścią aplikacji zajmuje się **JavaServer Pages** (JSP) zawierający w sobie semantykę znaną z hipertekstowego języka znaczników **HTML** oraz umożliwiający wplatanie języka Java między wiersze kodu HTML. W kwestii wizualnej nikogo nie powinien dziwić wybór kaskadowych arkuszy stylów **CSS** (w naszym rozwiązaniu automatycznie generowany przez dynamiczny język arkuszy stylów **Less**).

Kwestie bazodanowe załatwiamy za pomocą wolnodostępnego systemu zarządzania relacyjnymi bazami danych **PostgreSQL** dostępnego względem aplikacji z poziomu zewnętrznego serwera (podobnież w kwestii samego umiejscowienia aplikacji w sieci, aby dostęp do niej odbywał się nie tylko lokalnie, a przede wszystkim zdalnie), co w znacznym stopniu ułatwiło współpracę na tych samych danych.

W procesie tworzenia serwisu internetowego niezmiernie przydatne stają się wszelkiego rodzaju frameworki, wspierające i ułatwiające rozwój projektu. Podczas tworzenia szaty graficznej aplikacji webowej najciekawszym wsparciem okazał się **Bootstrap** – framework CSS rozwijany przez programistów Twittera. Kwestie skryptowe natomiast urozmaicił otwarty framework oparty na języku JavaScript, wspierany i firmowany przez Google **AngularJS**. Obydwie wspomniane platformy programistyczne udostępniane są na licencji MIT.

ROZDZIAŁ 3.4: IMPLEMENTACJA

Implementacja systemu odbyła się w dziesięciu etapach implementacyjnych trwających mniej więcej tydzień, gdzie każdy kolejny etap wymagał 10% postępu począwszy od pierwszego. Każdy etap zawierał w sobie konkretne zadania do wykonania, a spisane na koniec sprawozdanie określało co z tych zadań zostało zaimplementowane. Poszczególne etapy przedstawiają się następująco (według sporządzonych sprawozdań):

Etap 1: Stworzenie bazy danych, repozytorium Git umieszczonego na serwerze serwisu GitHub (z przygotowaniem wstępnej konfiguracji pozwalającej rozpoczęcie prac), wstępny zarys szaty graficznej, modeli aplikacji, rozpoczęcie pracy nad metodami

Etap 2: Uzupełnienie modeli aplikacji o dodatkowe klasy, rozpoczęcie prac nad połączeniem z bazą danych, przygotowanie paneli logowania i rejestracji, rozpoczęcie prac nad panelem administracyjnym, interfejsy i „szkielety" odpowiadających im managerów, przygotowanie okienka czatu i obsługi skryptowej

Etap 3: Rozpoczęcie prac nad CRUDem do istniejących modeli klas, kwestie wizualne paneli logowania i rejestracji, walidacja pól tekstowych w formularzach (patterny/regexy), wstępny zarys servletów do panelu administratora, wygląd edytora tekstu

Etap 4: Uproszczenie komend typu INSERT (dodawania do bazy danych) do każdej z tabel, przygotowanie panelu administracyjnego, formularz edycji użytkownika

Etap 5: Przygotowanie metod transakcyjnych, rozpoczęcie prac nad komendami typu UPDATE i DELETE, sprzężenie panelu rejestracji z bazą danych, usprawnienie edytora tekstu o numerowanie wierszy i kolorowanie składni, formularze edycji do pozostałych klas, kasowanie i modyfikacja

Etap 6: Przygotowanie zewnętrznego serwera bazodanowego i przekierowanie nań aplikacji, sprzężenie panelu logowania z bazą danych, rozpoczęcie prac nad zabezpieczaniem odpowiednich części aplikacji przed nieautoryzowanym dostępem (zalogowanie lub odpowiednie uprawnienia), wizualne aspekty funkcjonowania czatu

Etap 7: Przygotowanie modelu na wiadomości czatu i dodanie do bazy odpowiedniej tabeli, usprawnienie sesyjności logowania użytkowników, haszowanie haseł, prace nad wczytywaniem pliku z dysku, własny error 404, rozwinięcie możliwości dostępnych z poziomu zalogowanego użytkownika

Etap 8: Zakończenie prac nad wczytywaniem zawartości pliku tekstowego z dysku do edytora, cenzura w czacie, wstępne prace nad paginacją do tabel, przygotowanie profilu użytkownika

Etap 9: Dokończenie paginacji do tabel, umożliwienie użytkownikowi obsługi własnych plików (bez edycji zawartości), modyfikacja szaty graficznej

Etap 10: Sporządzenie niezbędnych metod do funkcjonowania czatu, prace nad zapisywaniem treści z edytora do bazy danych, prace nad obsługą czatu, testowanie i ewentualne poprawki

Poza zadaniami wymienionymi powyżej, wszelkie zaimplementowane komponenty były na bieżąco testowane i – w razie potrzeby – były na bieżąco wprowadzane niezbędne poprawki do zauważonych podczas pracy błędów.

ROZDZIAŁ 3.4.1: MODUŁY

W skład wykorzystanych przez nas modułów wliczyć należy przede wszystkim **CodeMirror**, wspierający tworzenie aplikacji typu edytor tekstu w aplikacjach webowych. Jest to darmowy edytor kodu źródłowego przeznaczony dla przeglądarek internetowych napisany w JavaScripcie. Obsługuje wiele języków programowania (m.in. używane przez nas JavaScript i CSS) oraz kolorowanie składni. Dołączone do niego skrypty i arkusze stylów w znacznym stopniu ułatwiły przygotowanie naszego edytora.

ROZDZIAŁ 3.5: TESTOWANIE

Przeprowadzone przez nas testy aplikacji odbywały się przede wszystkim w znany wszystkim tradycyjny sposób. Wszelkie formularze zostały podczas testów wyposażone w odpowiednie wzorce, aby już po stronie klienta zminimalizować odsetek przesyłania błędnych danych na stronę serwera i zwiększyć tzw. „idiotoodporność" systemu. Dostępne z poziomu użytkownika funkcjonalności były przez nas sprawdzane na wiele różnych, nawet dość nietypowych sposobów, a wszelkie niedociągnięcia na bieżąco poprawiane.

ROZDZIAŁ 3.6: WDROŻENIE

Zaimplementowany system w obecnej wersji został wdrożony na zewnętrzny serwer, umożliwiający dostęp do niego bezpośrednio z sieci Internet. Wszelka konfiguracja umożliwiająca jego poprawne działanie została przeprowadzona i przetestowana, a dostęp do bazy danych pozostał również po stronie zewnętrznego serwera, co sprawiło, że migracja danych nie była konieczna.

PODSUMOWANIE

Zgodnie z niemal wszelkimi założeniami, udało nam się zaprojektować, zaimplementować i wdrożyć aplikację webową umożliwiającą współpracę programistów poprzez edycję plików umieszczonych w bazie danych.