UNIWERSYTET GDAŃSKI

Wydział Matematyki, Fizyki i Informatyki

Łukasz Ekiert

nr albumu: 187 310

Aplikacja do wspomagania nauki języków obcych z wykorzystaniem frameworka Angular 2

Praca magisterska na kierunku:

INFORMATYKA

Promotor:

dr Włodzimierz Bzyl

Gdańsk 2017

Streszczenie

Aczkolwiek w ciągu ostatnich lat skład tekstu wspomagany komputerowo całkowicie wyeliminował stosowanie tradycyjnych technik drukarskich, to podobny proces w przypadku publikacji elektronicznych czyli publikacji, które w ogóle nie wykorzystują papieru, a nośnikem informacji staje się ekran komputera nie jest obserwowany.

Słowa kluczowe

języki obce, e-learning, Angular 2, Ruby on Rails 5

Spis treści

Wprowadzenie

Motywacją dla powstania pracy była chęć poznania stabilnej wersji frameworka*Angular 2*, wydanej 15. września 2016 roku, oraz wypróbowania dedykowanych dla niego narzędzi wspomagających proces tworzenia aplikacji. W ramach projektu wykorzystującego tę technologię stworzono aplikację wspomagającą naukę języków obcych.

W procesie nauki języka obcego zazwyczaj wykorzystuje się podręczniki dostosowane do konkretnej metody nauczania. W przypadku języka angielskiego można w ramach przykładu przytoczyć materiały udostępniane przez wydawnictwo *Pearson* (dawniej *Pearson Longman*), czy *Cambridge University Press*. Ich zawartość i stopień trudności różnią się w zależności od grupy docelowej, jednak ich generalną cechą jest organizacja materiału według przyjętego przez autorów programu nauczania [1]. Zazwyczaj jest to udogodnienie dla lektorów, gdyż nie muszą przygotowywać materiałów do zajęć (lub przygotowują ich mniej). Uczniowie również korzystają z tego, że przy tworzeniu takich podręczników biorą udział profesjonalni metodycy nauczania, zatem mają dostęp do pomocy dydaktycznych wysokiej jakości.

Podręczniki wydawane zarówno w formie papierowej, jak i elektronicznej, składają się z dwóch głównych części: teoretycznej, wprowadzającej osobę uczącą się w nowy materiał, oraz ćwiczeń praktycznych. Te ostatnie zawierają testy służące sprawdzeniu i utrwaleniu zdobytej wiedzy. Ich forma z reguły opiera się na sprawdzonych schematach ćwiczeń, takich jak: wypełnianie luk, parowanie fraz, testy jedno- i wielokrotnego wyboru, krzyżówki. Dodatkowo wykorzystuje się pomoce multimedialne w formie obrazków, zdjęć, nagrań audio i wideo, jak również aplikacje e-learningowe.

Aplikacje takie można podzielić na dwie zasadnicze grupy: dedykowane dla konkretnych podręczników oraz samodzielne oprogramowanie, które zazwyczaj funkcjonuje w formie serwisów webowych. Istniejącymi na rynku rozwiązaniami z tej drugiej grupy są na przykład *busuu* i *Duolingo*. Umożliwiają one rozwiązywanie ćwiczeń oraz zapoznawanie się z przygotowanymi materiałami dydaktycznymi,

ponadto zawierają szereg udogodnień, takich jak: spersonalizowane testy, kontrola postępów, konwersacje z *native speakers* (zazwyczaj w formie płatnej), zgrywalizowany proces nauki.

Warto zwrócić uwagę na fakt, że takie aplikacje i podręczniki zazwyczaj skierowane są do szerokiego grona odbiorców. W wielu przypadkach można taką cechę uznać za zaletę, jednak należy mieć na uwadze, iż istnieją również grupy wymagające materiałów do nauki o bardzo konkretnej tematyce lub sposobie przedstawiania oraz sprawdzania wiedzy. W tym przypadku często korzysta się z autorskich podręczników i programów nauczania, stworzonych pod kątem tych wymagań. Z reguły osoby tworzące takie rozwiązania nie mają środków ani umiejętności, aby stworzyć do swoich materiałów warstwę aplikacji e-learningowej, która wspomagałaby proces dydaktyczny.

Celem niniejszej pracy było stworzenie aplikacji, dzięki której podmioty zajmujące się nauczaniem języków obcych przy pomocy autorskich rozwiązań mogłyby skorzystać z możliwości e-learningu. Twórca podręcznika może tworzyć własne testy (zgrupowane w kursy) zawierające ćwiczenia oparte na przygotowanych uprzednio szablonach. Uczniowie mogą rozwiązywać testy z przypisanych im kursów oraz śledzić własne postępy. Każdy podmiot uruchamia własną instancję aplikacji, więc ma możliwość ograniczenia zależności od zewnętrznych usługodawców.

Główną częścią projektu jest aplikacja kliencka napisana we frameworku *Angular 2*. Zawiera ona interfejsy dla wszystkich przyjętych rodzajów kont (administratorów, kierowników, nauczycieli oraz uczniów) i zawiera większość logiki projektu. Jest skomunikowana z API zgodnym ze standardem JSON API [3].

Warstwą API jest aplikacja napisana we frameworku *Ruby on Rails 5*. Został on wybrany ze względu na szybkość wytwarzania gotowego produktu, ponadto posiada on rozszerzenia (*gemy*) wspierające wystawianie końcówek (*endpointów*) zgodnych ze standardem JSON API. Służy on głównie jako interfejs komunikacyjny pomiędzy aplikacją kliencką, a bazą danych, zawiera jednak w niektórych miejscach elementy logiki biznesowej, jak np. weryfikacja poprawności rozwiązań.

W części opisującej założenia projektu przedstawiono podstawowe terminy,

które zostały przyjęte w ramach konkretyzacji pojęciowej głównych elementów projektu. Następnie zostały opisane główne założenia przyjęte podczas projektowania aplikacji, wraz z analizą potrzeb użytkowników. Opisano typy kont: administratora, kierownika, nauczyciela, ucznia, wraz z wymaganiami funkcjonalnymi dla każdego rodzaju konta, podano zaimplementowane typy ćwiczeń.

W szczegółach implementacyjnych przedstawiono architekturę i wykorzystane technologie każdej z warstw aplikacji. Opisano biblioteki i narzędzia wspomagające proces wytwarzania aplikacji, umieszczono diagramy klas. Następnie poruszono problematykę testowania aplikacji. Przedstawiono scenariusze testów funkcjonalnych oraz opisano wybrane testy jednostkowe. Uwagę poświęcono również testom manualnym. W ostatniej części podsumowano rezultaty testów.

ROZDZIAŁ 1

Projekt aplikacji

W projekcie aplikacji przedstawione są definicje terminów związanych z aplikacją, wymagania niefunkcjonalne oraz funkcjonalne. Z racji tego, że projekt przewiduje wyodrębnienie czterech typów kont użytkowników, wymagania funkcjonalne zostały pogrupowane według odpowiadających im typów kont.

1.1. Użyte terminy

- **Klient** - warstwa kliencka projektu konsumująca JSON API,
- **Użytkownik** - konto w aplikacji o przypisanym typie.
- **Uczeń** - typ konta użytkownika końcowego. Zorientowane jest na rozwiązywanie ćwiczeń i podgląd własnych akcji oraz postępów.
- **Nauczyciel** - typ konta lektora. Służy do nadzorowania rezultatów testów rozwiązywanych przez uczniów do przypisanych mu grup.
- **Kierownik** - typ konta przewidziany do podglądu wszystkich akcji użytkowników aplikacji. Przykładem osoby z kontem kierownika jest pracownik szkoły językowej nadzorujący wywiązywanie się z obowiązków przez lektorów.
- **Administrator** - konto o najszerszych uprawnieniach. Osoby z kontem administratora odpowiedzialne są za tworzenie treści dostępnej dla pozostałych kont użytkowników: dodawanie kursów, testów, ćwiczeń, zarządzanie kontami użytkowników.
- **Grupa** - zbiór uczniów oraz przypisanych im nauczycieli oraz kursów.
- **Kurs** - zbiór testów. Organizacja testów zależy od administratora.

- **Test** - zbiór ćwiczeń, z założenia związanych tematyką i stopniem trudności.

- **Ćwiczenie** - jednostka testowa służąca do weryfikacji posiadanej wiedzy oparta o przygotowany szablon.

- **Szablon** - typ ćwiczenia.

1.2. Wymagania aplikacji

1.2.1. Użytkownicy aplikacji

Do korzystania z aplikacji niezbędne jest posiadanie konta. Konta są zakładane przez Administratora, w aplikacji nie ma przewidzianej możliwości samodzielnej rejestracji, jednak przy spełnieniu założenia prostego i otwartego API, taka funkcja może zostać łatwo zaimplementowana. Każde konto Użytkownika musi mieć przypisany konkretny typ determinujący zarówno stopień uprawnień do wyświetlania, tworzenia, edycji oraz usuwania poszczególnych zasobów, jak i wyświetlany po zalogowaniu interfejs. Krytycznymi dla poprawnego działania aplikacji typami kont są Administrator oraz Uczeń. Typy Nauczyciela oraz Kierownika pełnią funkcję pomocniczą.

Przyczyną takiej klasyfikacji jest to, że Administrator jest z założenia kontem o największych możliwościach i to on odpowiada za treść oraz konta pozostałych Użytkowników, zaś Uczeń jest ostatecznym konsumentem aplikacji, gdyż to z myślą z nich powstała. Zadaniem nauczyciela jest nadzorowanie postępów własnych podopiecznych, poza tym nie posiada on innych szczególnych uprawnień. Kierownik ma możliwość wglądu w akcje wszystkich Użytkowników aplikacji. Intencją powstania tego typu konta było zapotrzebowanie na możliwość kontrolowania zaangażowania uczniów oraz wywiązywania się ze swoich obowiązków przez lektorów. Z powyższych opisów wypływa zatem wniosek, iż to Administratorzy oraz Uczniowie z założenia mają główny wpływ na zmiany stanu modelu, zaś Nauczyciel oraz Kierownik są, co do zasady, jedynie obserwatorami.

1.2.2. Wymagania funkcjonalne

Wymagania wspólne dla wszystkich typów kont

- Użytkownik musi dokonać autentykacji, by korzystać z aplikacji. Administrator zakłada każdej osobie, która będzie korzystać z aplikacji, konto o odpowiednim poziomie uprawnień.
- Użytkownik musi mieć dostęp do logów akcji. Każdy użytkownik musi mieć możliwość wyświetlenia widoku, w którym zawarte są logi akcji oraz ich wykres. Zakres danych wyświetlanych w takim widoku jest determinowany przez typ konta.
- Użytkownik musi mieć możliwość zmiany swojego hasła.

Wymagania dla konta Ucznia

- W widoku podsumowania muszą być wyświetlane wyłącznie logi danego Ucznia.
- Uczeń musi mieć możliwość:

  bycia przypisanym do grupy.

  rozwiązywania testów, które są przypisane do jego grupy.
- Po przesłaniu rozwiązania testu, Uczeniowi musi zostać od razu wyświetlony wynik wyrażony w punktach procentowych oraz muszą być zaznaczone błędne odpowiedzi.
- Uczeń musi mieć dostęp do historii rozwiązanych testów.

Wymagania dla konta Nauczyciela

- Nauczyciel musi mieć:

  - możliwość bycia przypisanym do Grupy jako lektor,

  - dostęp do wszystkich rozwiązań Uczniów w grupach, w których jest zapisany jako lektor.

- W widoku podsumowania muszą być wyświetlane logi zarówno danego Nauczyciela, jak i wszystkich Uczniów z grup do niego przypisanych.

Wymagania dla konta Kierownika

- Kierownik musi mieć możliwość:

  - wyświetlenia list kont Nauczycieli i Uczniów,

  - wyświetlenia listy Grup.

- Nauczyciel musi mieć dostęp do wszystkich przesłanych rozwiązań testów.

- W widoku podsumowania muszą być wyświetlane logi wszystkich Użytkowników aplikacji.

Wymagania dla konta Administratora

- Administrator musi mieć możliwość:

  zarządzania Użytkownikami, Grupami, Kursami oraz Testami,

  tworzenia Testów z ćwiczeniami, które mogą być rozwiązywane przez Uczniów oraz Nauczycieli, o ile są przypisane do tych samych Grup, co konta,

  zmiany haseł wszystkich Użytkowników na samodzielnie wybrane lub automatycznie wygenerowane,

  edycji danych osobowych wszystkich Użytkowników.

- W widoku podsumowania muszą być wyświetlane logi wszystkich Użytkowników aplikacji.

1.2.3. Wymagania niefunkcjonalne

- **API oparte na *Ruby on Rails 5***

- **Klient oparty na frameworku *Angular 2*** Z racji tego, że zasadniczą motywacją powstania projektu była chęć poznania tej technologii, to

główna część musi zostać w niej napisana. Należy mieć na uwadze, że jest wciąż dynamicznie rozwijana, dlatego

- **Klient napisany w języku TypeScript**
- **Otwarte źródło** - aby zagwarantować możliwość korzystania zainteresowanym podmiotom z projektu, aplikacja musi być dostępna w formie darmowej i gotowej do własnoręcznego uruchomienia. Całość powinna znajdować się w publicznym repozytorium w serwisie *GitHub*.
- **Responsywność** - typ konta przewidziany do podglądu wszystkich akcji użytkowników aplikacji. Przykładem osoby z kontem kierownika jest pracownik szkoły językowej nadzorujący wywiązywanie się z obowiązków przez lektorów.
- **Otwartość na rozszerzenia** - konto o najszerszych uprawnieniach. Osoby z kontem administratora odpowiedzialne są za tworzenie treści dostępnej dla pozostałych kont użytkowników: dodawanie kursów, testów, ćwiczeń, zarządzanie kontami użytkowników.
- **Ćwiczenia z treściami audiowizualnymi** - niezbędna jest możliwość dodawania do testów ćwiczeń, w których umieszczone są pliki graficzne oraz dźwiękowe. Materiały o charakterze multimedialnym są szczególnie istotne w przypadku kształcenia dzieci i młodzieży, jak również do treningu umiejętności związanych z rozumieniem słuchanych wypowiedzi.
- **Niezależność** - aplikacja powinna działać poprawnie na popularnych przeglądarkach. Przyjęto, że takimi przeglądarkami są wymienione w oficjalnej dokumentacji frameworka *Angular 2*. Ponadto użytkownik nie powinien być zmuszony do instalacji w swoim systemie żadnych rozszerzeń ani wtyczek.

1.2.4. Przypadki użycia

Poniżej zamieszczone są kluczowe przypadki użycia aplikacji:

TODO

- *Administrator dodaje zasób (konto, kurs, grupę)*
- *Administrator przypisuje zasób do grupy*
- *Administrator dodaje test do kursu*
- *Administrator dodaje ćwiczenia do testów*
- *Administrator dodaje ćwiczenie ”luki” do testu*
- *Administrator dodaje ćwiczenie ”wybór” do testu*
- *Użytkownik rozwiązuje test*
- *Użytkownik wyświetla rozwiązany w przeszłości test*

Tabela 1.1. **UC1** - Dodawanie zasobu

UC1	Dodawanie zasobu
Aktor	Administrator
Warunki	• Administrator zalogowany w aplikacji. • Zasoby: konta, testy, kursy, grupy.
Przebieg	1. Administrator wybiera widok odpowiedniego zasobu. 2. Administrator wybiera widok formularza dodawania zasobu. 3. Administrator wypełnia formularz i zatwierdza dodanie zasob 4. Wyświetlany jest komunikat o pomyślnym wykonaniu operacj
Przypadki szczególne	W przypadku dodawania użytkownika, Administrator mo wygenerować hasło lub wybrać własne.
Wyjątki	Wpisane dane są nieprawidłowe - wyświetlany jest komunikat niepowodzeniu.

Tabela 1.2. **UC5** - Dodawanie ćwiczenia do testu

UC5	Dodawanie zasobu
Aktor	Administrator
Warunki	Administrator zalogowany w widoku edycji testu.
Przebieg	1. Administrator wybiera rodzaj ćwiczenia, które chce dodać do testu i zatwierdza wybór. 2. Administrator wpisuje wypełnia formularz nowo dodanego ćwiczenia (**UC5.1**, **UC5.2**). 3. Administrator zatwierdza dane ćwiczenia. 4. Administrator może wrócić do **UC5.0.1**, aby dodać kolejne ćwiczenie, lub zakończyć procedurę zapisując test, by zatwierdzić zmiany.

Tabela 1.3. **UC5.1** - Dodawanie ćwiczenia *luki*

UC5.1	Dodawanie ćwiczenia typu *luki*
Aktor	Administrator
Warunki	• Administrator w widoku edycji testu. • Administrator w **UC5** wybrał
Przebieg	1. Administrator dodaje nowe zdanie do ćwiczenia, oznaczaj brakujące frazy (luki) ustalonym ciągiem znaków specjalnych 2. Z każdym wystąpieniem luk, pod polem zdania pojawiają odpowiadające im pola. 3. Administrator w tych polach wpisuje prawidłowe rozwiązan dla każdej odpowiadającej luki. W przypadku, gdy dla danej lu poprawna jest więcej niż jedna fraza, oddziela je przecinkiem 4. Administrator może przejść ponownie do **UC5.1.1** lub **UC5.0.4**

ROZDZIAŁ 2

Szczegóły implementacji

2.1. API

2.1.1. Wykorzystane biblioteki

JSONAPI::Resources

JSONAPI::Resources jest biblioteką rozwijaną na licencji MIT, służącą jako zestaw narzędzi pomocniczych dla frameworka *Ruby on Rails* (wersji 4.2 i nowszych) do implementacji standardu *JSON API*. W ich skład wchodzą m.in. funkcje definiujące ścieżki dla kontrolerów, zasoby (*resources*) i służące do definiowania relacji pomiędzy nimi metody, jak również wsparcie dla mechanizmów filtrowania, dołączania podzasobów, paginacji oraz innych operacji wchodzących w skład standardu. Odpowiedzi na żądania HTTP są zautomatyzowane - programista nie musi martwić się o formatowanie danych wyjściowych, czy zwracane kody dla zapytań.

- **Definiowanie ścieżek** - funkcje definiujące ścieżki zasobów aplikacji, które wchodzą w skład biblioteki, rozszerzają zachowanie natywnych dla *Ruby on Rails* o uwzględnianie relacji pomiędzy klasami zasobów.
- **Definiowanie kontrolerów** - dziedziczenie po klasie *ResourceController* niemal całkowicie automatyzuje logikę przetwarzania żądań do API oraz odpowiedzi. Walidują nagłówki i format przesłanych danych, przekazują przetworzone żądanie do klas modelowych, formatują zwracane dane.
- **Definiowanie zasobów** - każda końcówka API musi być reprezentowana przez klasę zasobu. Klasy zasobów zawierają główną logikę odpowiedzialną za spełnianie wymagań *JSON API*, do których należą m.in.: definiowanie relacji pomiędzy zasobami, strategie filtrowania, paginacji oraz sortowania,

określanie atrybutów i reguł dostępu (np. *read-only*), rodzaju zasobu (np. *singleton*).

Knock

Gem *Knock* służy do autentykacji użytkowników aplikacji opartych o Rails API przy pomocy standardu *JSON Web Token* (*JWT*). Zastosowanie takiego podejścia gwarantuje bezstanowość, wymaganą przez *REST*, a więc również *JSON API*. Z szeregu udogodnień zapewnianych przez bibliotekę można wymienić:

Paperclip

TODO: napisać o gemie do załączników

2.2. Aplikacja kliencka

Aplikacja kliencka jest główną częścią projektu i konsumentem API. Dostarcza wszystkich niezbędnych mechanizmów i funkcji dla każdego z rodzajów użytkowników.

Angular 2

Wykorzystane biblioteki i narzędzia

Angular CLI

Ważnym narzędziem pomagającym przy procesie wytwarzania aplikacji napisanych we frameworku *Angular 2* jest narzędzie CLI. Nie jest integralną częścią repozytorium frameworka, więc musi być instalowane w systemie odrębnie. Zawiera skrypty inicjalizujące nowe projekty, generatory dla wszystkich rodzajów obiektów (klas, komponentów, dyrektyw, usług itp.), zintegrowany server deweloperski (oparty na *webpack-dev-server*), budujące projekt oraz testujące kod (wykorzystujące narzędzia *Karma*, *Jasmine* oraz *Protractor*).

Zakończenie

Wnioski

Spis tabel

Spis rysunków

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....	.....
data	podpis