

**UNIwersYTET RZESZOWSKI**

**Kolegium Nauk**



Krzysztof Staniszewski

Nr albumu: 104995

Informatyka

**Aplikacja internetowa ułatwiająca naukę programowania**

Praca inżynierska

Praca wykonana pod kierunkiem

dr inż. Wojciech Kozioł

Rzeszów, 2021

## Spis treści

|   |           |
|---|-----------|
| <b>1. Wstęp .....</b>   | <b>3</b>  |
| <b>1.1 Opis problemu .....</b>                                  | <b>3</b>  |
| <b>1.2 Cel, założenia i zakres pracy .....</b>                  | <b>3</b>  |
| <b>1.3 Opis podobnych rozwiązań istniejących na rynku .....</b> | <b>4</b>  |
| <b>2. Projekt aplikacji .....</b>                               | <b>5</b>  |
| <b>2.1. Architektura aplikacji .....</b>                        | <b>5</b>  |
| <b>2.3 Projekt i struktura bazy danych .....</b>                | <b>11</b> |
| <b>3. Implementacja. ....</b>                                   | <b>12</b> |
| <b>3.1 Opis użytych w pracy narzędzi i technologii .....</b>    | <b>12</b> |
| <b>3.2 Struktura aplikacji .....</b>                            | <b>13</b> |
| <b>3.3 Implementacja części biznesowej aplikacji .....</b>      | <b>16</b> |
| <b>3.4 Implementacja części frontendowej aplikacji .....</b>    | <b>21</b> |
| <b>3.5 Opis uruchomienia aplikacji .....</b>                    | <b>23</b> |
| <b>4. Prezentacja działania aplikacji. ....</b>                 | <b>29</b> |
| <b>5. Zakończenie .....</b>                                     | <b>51</b> |
| <b>6. Literatura .....</b>                                      | <b>51</b> |
| <b>Spis tabel i rysunków .....</b>                              | <b>52</b> |
| <b>Streszczenie .....</b>                                       | <b>55</b> |

## **1. Wstęp**

### **1.1 Opis problemu**

Znalezienie portalu do nauki programowania nie jest trudne, wystarczy użyć wyszukiwarki internetowej. Pomimo tego problemem początkujących programistów jest brak wiedzy uniemożliwiający im znalezienie kursu odpowiedniego do ich umiejętności. Zaawansowany programista także może mieć problem ze znalezieniem interesującego go zagadnienia do nauki, gdyż większość treści przeznaczona jest dla początkujących lub jest zbyt niszowa, aby je łatwo znaleźć. Jednak największym problemem jest niezrozumienie kodu podanego przez autora. Dla początkującego nawet jedna niezrozumiana linijka może spowodować błędne zrozumienie algorytmu. Nawet jeśli kod jest wytłumaczony w treści „lekcji” to wyszukiwanie odpowiedzi może nie być oczywiste i skutkować frustracją, brakiem zrozumienia czy po prostu zaprzestaniem poszukiwania odpowiedzi, dlatego bardzo ważne jest wytłumaczenie poszczególnych linii programu tam, gdzie się one znajdują. Najczęściej komentowanie kodu aplikacji w kodzie źródłowym jest niewystarczające lub autor go nie stosuje. Wiele kursów posiada obszerne wytłumaczenia omawiające działanie kodu w skondensowanych opisach poniżej. Rozwiązanie takie prowadzi do mało czytelnego i nieprzyjemnego dla użytkownika kursu.

### **1.2 Cel, założenia i zakres pracy**

Celem pracy jest stworzenie aplikacji, która umożliwi tworzenie kursów programistycznych będących przyjaznymi dla użytkownika, a proces ich tworzenia jest łatwy do zrozumienia i nie sprawia większych problemów.

Aplikacja umożliwia tworzenie lekcji z kodem oraz treścią rozwijającą zagadnienie, gdzie kod nie jest przedstawiany jako czysty tekst, lecz jako interaktywne linijki kodu. Każda linia może być dowolnie zaprogramowana przez autora tak, aby zawrzeć w niej wytłumaczenie każdej jej części, przykładowo opisać zagnieżdżone funkcje lub ich parametry. Jest to szczególnie ważne, gdyż często brak zrozumienia małej części kodu może znacznie zmniejszyć zrozumienie treści.

Jeśli użytkownik po zakończeniu lekcji czegoś nie rozumie, może w łatwy sposób zaznaczyć linijki, które sprawiły mu trudność i dopytać o nie w postaci komentarza.

Pytania takie są łatwo dostępne do przeczytania obok zaznaczonych linii, co pozwoli przyszłym użytkownikom łatwo znaleźć odpowiedź na pytania, które już zostały zadane.

Wprowadzona jest także funkcjonalność tworzenia kursów złożonych z wielu lekcji. Pozwala to porządkować wiedzę i przedstawiać złożoność tematyki, co z kolei pomoże w dobraniu zagadnienia według preferencji użytkownika. Umożliwione jest ustawianie kursów w odpowiedniej kolejności dla ułatwienia grupowania zagadnienia w całość. Każdy kurs ma przypisany jeden z trzech stopni trudności. Umożliwi to użytkownikowi racjonalne dobranie trudności do własnych możliwości. Zainteresowany grupą kursów użytkownik może ją łatwo obserwować i dostawać powiadomienia o publikacji nowej zawartości.

Wyszukiwanie kursu jest kluczowym elementem strony, który powinien być odpowiednio dobrany do celów i możliwości użytkownika. Aplikacja pozwala na różne kryteria przeszukiwania:

- a) Wyszukiwanie przy użyciu słów kluczy
- b) Wyszukiwanie kursów od konkretnego autora
- c) Wyszukiwanie po kategorii i języku programowania
- d) Wyszukiwanie kursów w określonej strefie trudności

### **1.3 Opis podobnych rozwiązań istniejących na rynku**

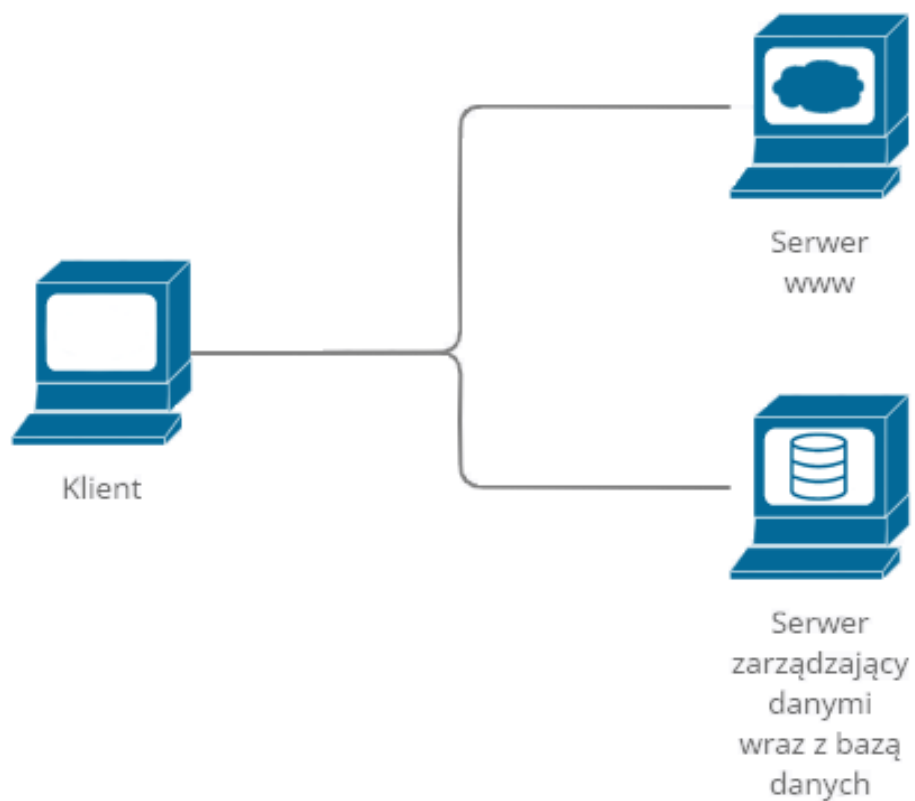
Podobne rozwiązania na rynku zawierają dobrze zaplanowane kursy, w których autorzy starają się dobierać przykłady, tak aby uczeń o określonych umiejętnościach mógł łatwo śledzić i rozumieć treść. Przykładowo kursy dla początkujących nie zawierają wielu linii kodu, czy zaawansowanych algorytmów. Autorzy starają się przedstawić podstawowe zagadnienia w możliwie izolowany sposób. Jeden z dostępnych kursów online to „Code Academy”, gdzie treści są bardzo dobrze wytłumaczone, a przedstawiony kod jest czysty, bez zbędnych elementów. Jednak pomimo tego portalowi brakuje pewnych funkcjonalności. Kod zawarty w kursie jest co prawda niezbyt obszerny i przejrzysty, ale autorowi zdarza się tłumaczyć fragmenty kodu w treści znajdującej się na kolejnych stronach w sporym oddaleniu od tłumaczonego kodu, co utrudnia jego zrozumienie. Aplikacja przedstawiona w pracy pomaga rozwiązać ten problem poprzez wytłumaczenie części kodu w miejscu, gdzie się on znajduje. Treść kursu „Code Academy” zawiera wybrane przez autorów pytania,

które mógł zadać użytkownik, lecz jeśli użytkownik nie rozumie i ma zamiar zadać inne pytanie jest odsyłany na osobną stronę forum, gdzie może dostać odpowiedź. Kursant powinien móc zadać pytanie bez konieczności zmiany adresu strony internetowej. Aplikacja przedstawiona w pracy rozwiązuje ten problem poprzez udostępnienie możliwości zapytań obok kodu w formie komentarza. Kolejnym przykładem strony z kursami online jest „Udemy”. Autorzy tworzą kursy w postaci wideo, gdzie piszą kod, tak aby użytkownik mógł go pisać razem z nimi. Jest to dobre rozwiązanie, gdyż autor może tłumaczyć kod werbalnie podczas pisania go. Kursy zawierają kody źródłowe przedstawiające stan kursu po każdej lekcji. Jednak pomimo tego, autorzy często pomijają część linii kodu, pozostawiając użytkownikowi sprawdzenie ich znaczenia. Przykładem może być przypadek, gdzie autor pisze algorytm dotyczący tematu kursu, lecz nie omawia w ogóle procesu konfiguracji projektu. Użytkownik nierozumiejący tej części kodu będzie w przyszłości przepisywał go bezmyślnie nie zastanawiając się nad jego znaczeniem. Aplikacja przedstawiona w pracy pomaga rozwiązać ten problem, gdyż autor nawet jeśli nie porusza danego tematu w tekście, może go pobieżnie opisać w postaci komentarza kodu.

## **2. Projekt aplikacji**

### **2.1. Architektura aplikacji**

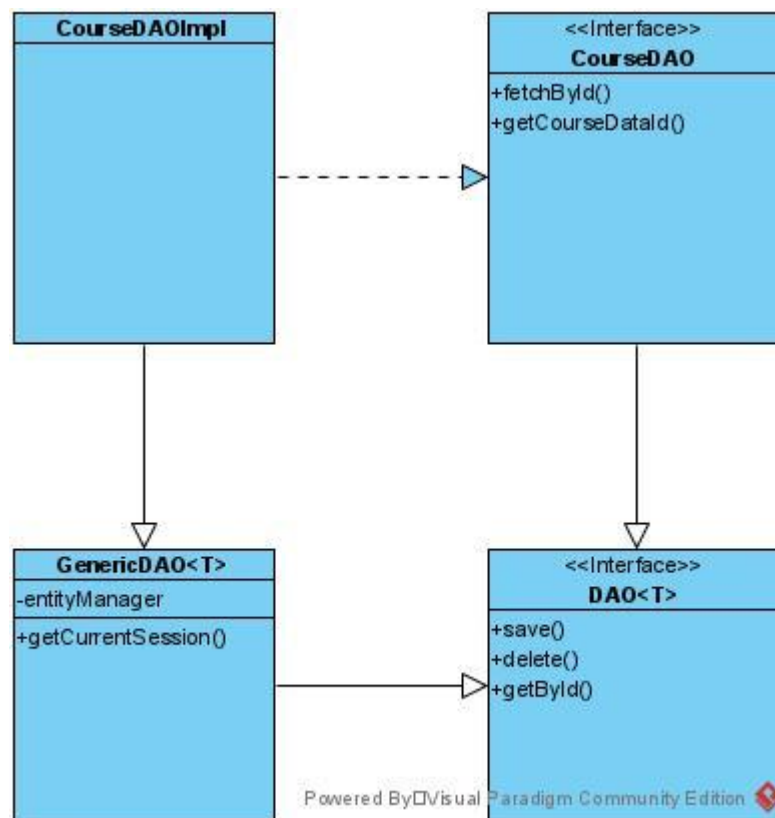
Aplikacja oparta jest na architekturze, gdzie klient łączy się w dwoma serwerami (zobacz rysunek 1). Jednym z nich jest serwer www, który udostępnia stronę internetową aplikacji. Klient pobiera stronę www i wykonuje ją w przeglądarce internetowej. Strona łączy się z serwerem zarządzającym danymi, aby pobrać dane do wyświetlenia. Połączenie jest obsługiwane poprzez protokół http, gdyż serwer danych opiera się o standard REST [2].



*Rysunek 1 – Ogólna architektura aplikacji (źródło: opracowanie własne)*

## **2.2 Diagramy UML**

Poniższy diagram klas przedstawia strukturę klas funkcjonalnych projektu *DAO*, które poprzez dziedziczenie oraz implementację innych klas pozwalają na komunikację z bazą danych (zobacz rysunek 2).

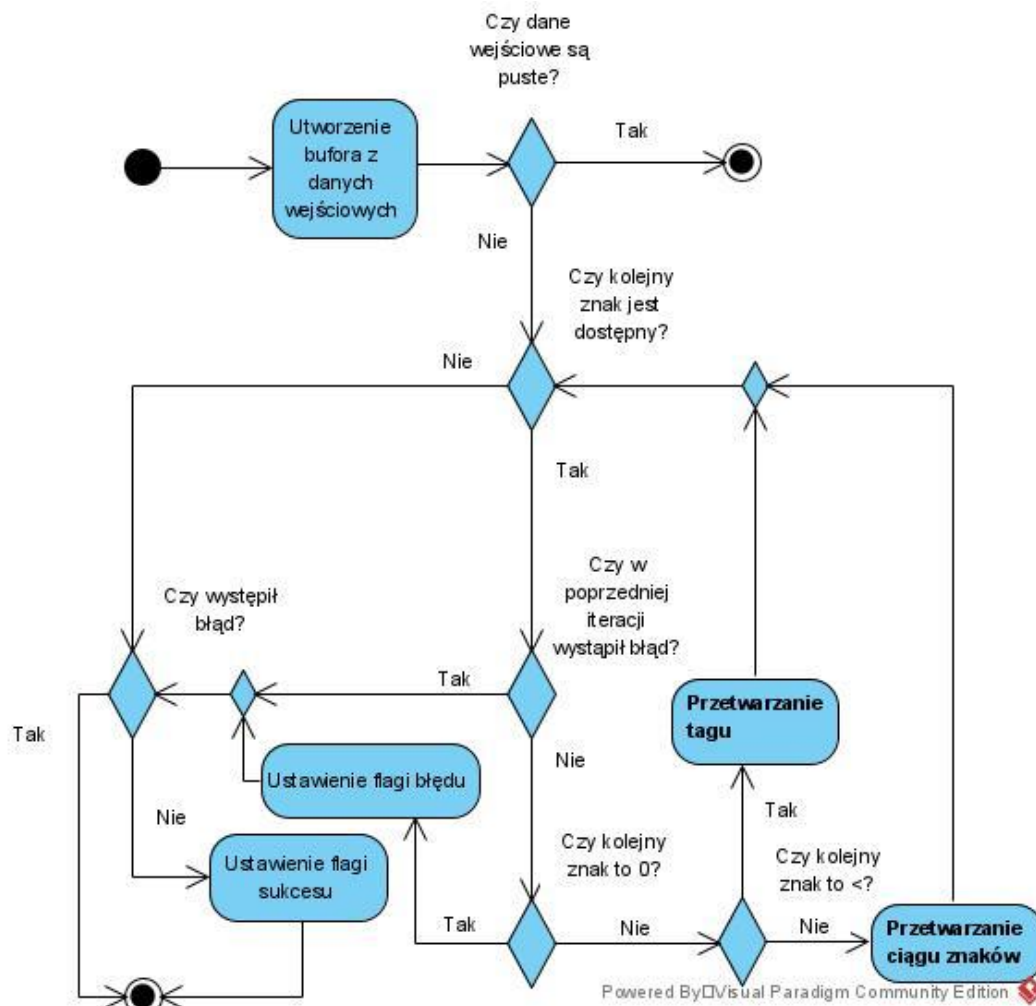


Rysunek 2 – Przykład diagramu klas DAO (źródło: opracowanie własne)

Interfejs *DAO<T>* jest typu ogólnego rozszerzany przez parametr *T*. Parametr *T* będzie reprezentował obiektowy model danych zmapowany z odpowiednią tabelą w bazie danych. Pozwala on na deklarację podstawowych działań na tym obiekcie, takich jak zapis, usuwanie i odczyt rekordu po jego identyfikatorze. Interfejs *DAO<T>* stanowi również bazę dla klasy *GenericDAO<T>*, która implementuje połączenie z bazą danych i funkcję dostępu do sesji połączenia.

Interfejs *CourseDAO* dziedziczy z interfejsu *DAO* i nakłada specyfikację typu *T* na typ *Course*. Interfejs ten deklaruje wszelkie publicznie dostępne metody, które będą implementowane przez klasę nadrzędną oraz używane przez funkcje usług. Klasa *CourseDAOImpl* dziedziczy z *GenericDAO* oraz implementuje interfejs *CourseDAO*. Dzięki dziedziczeniu otrzymuje ona dostęp do bazy danych. Wymaga ona implementacji funkcji z interfejsów *DAO* oraz *CourseDAO*.

Przedstawiony na rysunku diagram aktywności (zobacz rysunek 3) przedstawia sposób zamiany danych wejściowych z postaci ciągu znaków na listę indeksów potrzebnych do dalszej pracy parsera.



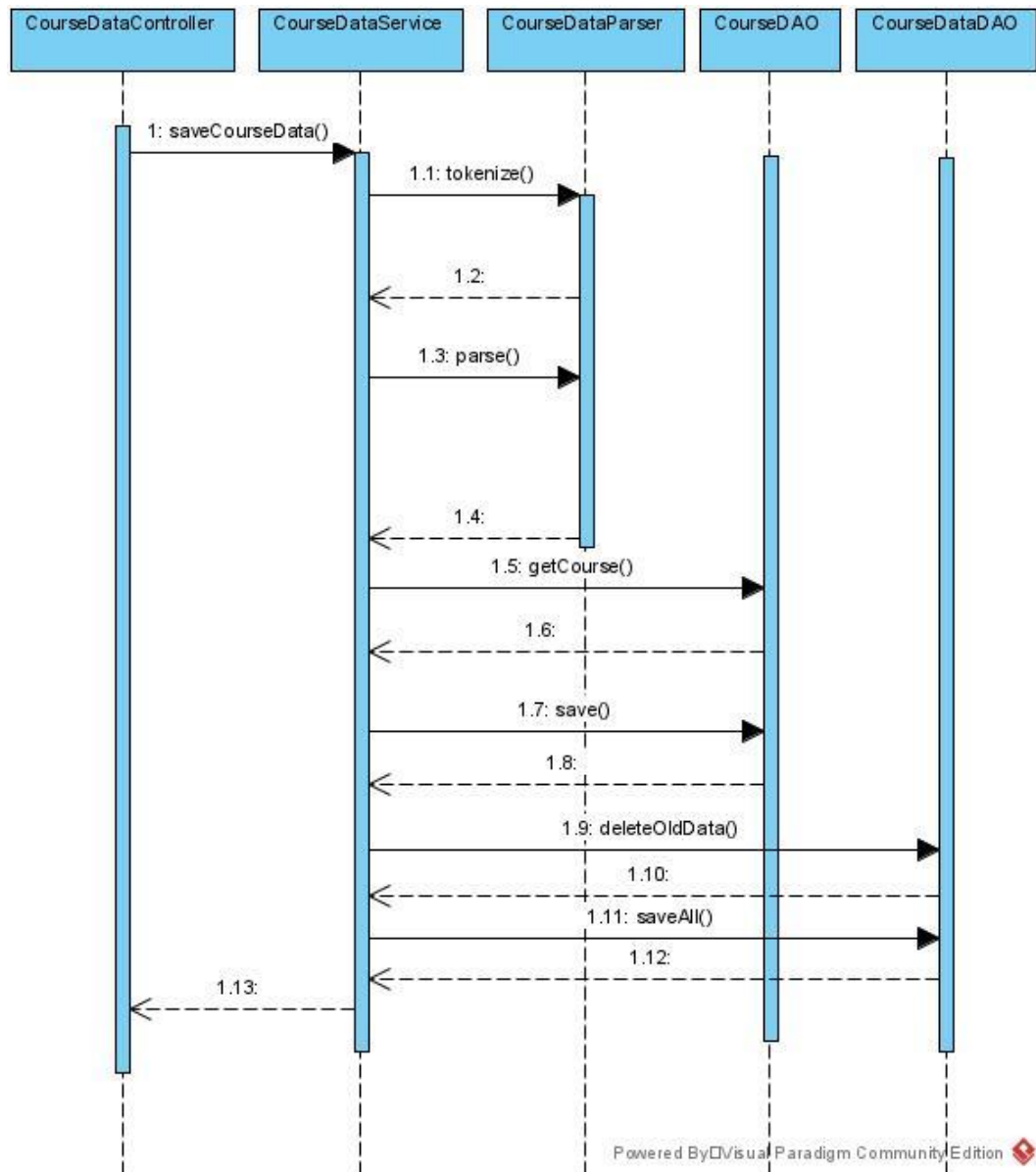
Rysunek 3 – Diagram aktywności tokenizer (źródło: opracowanie własne)

Początkowo dane są konwertowane z ciągu znaków do listy znaków, poprzez utworzenie bufora. Jeśli wprowadzone dane były puste, algorytm kończy pracę. Jeśli dane źródłowe nie były puste, pracę rozpoczyna pętla, której warunkiem działania jest dostępność kolejnych danych, w tym przypadku kolejnego znaku w buforze. Wewnątrz pętli sprawdzane są flagi parsera, wystąpienie błędu w poprzedniej iteracji przerywa pętlę. Kolejnym krokiem jest wstępne sprawdzenie czy kolejny element jest przewidywany jako tag, czy ciąg znaków. Zgodnie z warunkiem wykonywana jest



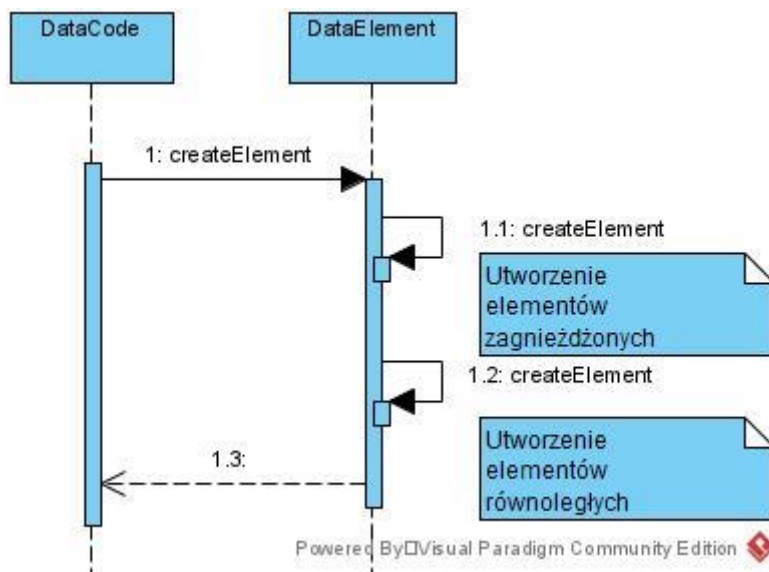
odpowiednia funkcja. Jeżeli bufor jest pusty, pętla zostaje przerwana. Sprawdzana jest flaga błędu, jeśli taki nie wystąpił ustawiana jest flaga sukcesu, następnie proces jest kończony.

Załączony poniżej diagram sekwencji przedstawia proces zapisu nowych danych kursu (zobacz rysunek 4).



Rysunek 4 – Diagram sekwencji zapisu treści kursu (źródło: opracowanie własne)

Obiekt usług *CourseDataService* współpracuje z obiektami praserów oraz obiektami dostępu do danych, aby odpowiednio obsłużyć zapis. Obiekt klasy usług<sup>1</sup> wykonuje na obiekcie parsera funkcję *tokenize()*. Część tokenizera w parserze przetwarza dane wejściowe i udostępnia na zewnątrz flagi błędów. Jeżeli nie wystąpiły błędy, klasa usług może użyć funkcji *parse()*, aby parser rozpoczął próbę tworzenia obiektu z wygenerowanej wcześniej przez tokenizer listy indeksów. Poprzez funkcję *getCourse()* pobierany jest obiekt z bazy danych, a po wprowadzeniu zmian jego zawartość jest zapisywana do bazy poprzez metodę *save()*. Obiekt stworzony w procesie parsowania jest zapisywany do bazy danych zastępując stare dane, które są usuwane. Dokonywane jest to poprzez użycie metod *deleteOldData()* oraz *saveAll()*. Komunikacja z bazą danych następuje poprzez obiekty DAO.

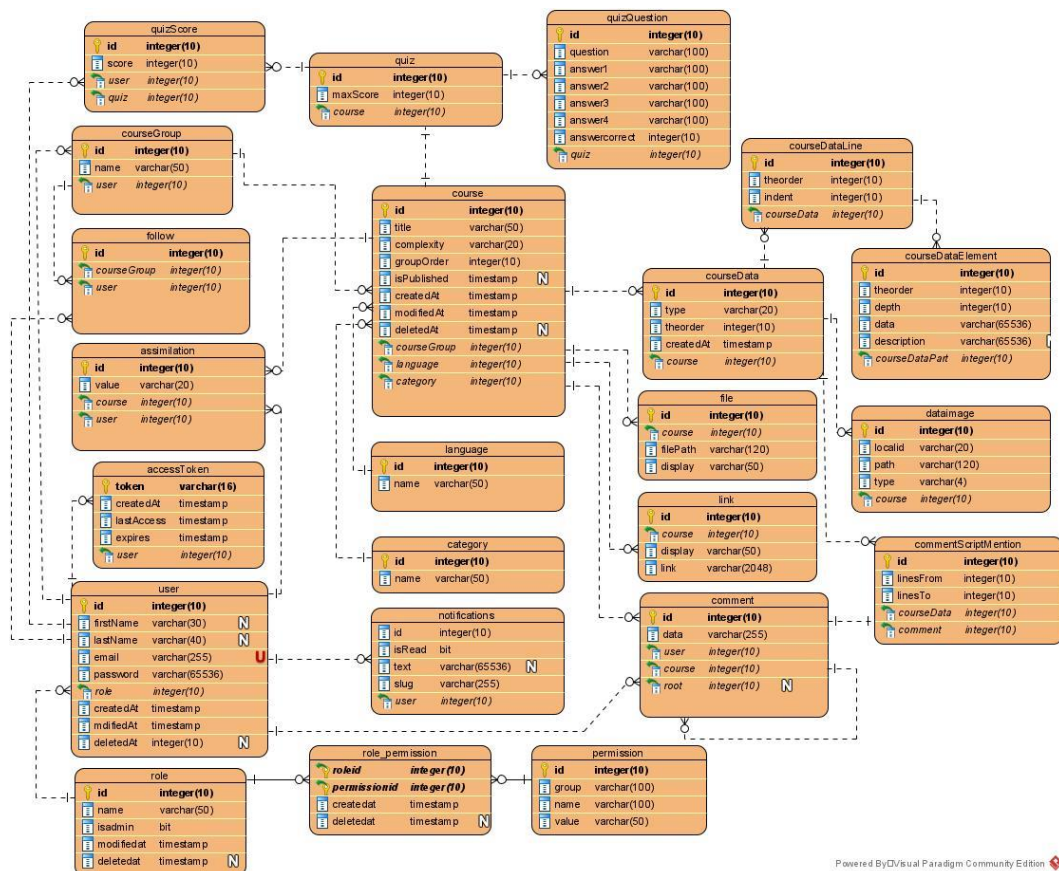


<sup>1</sup> Klasy usług – klasy, w których znajduje się główna logika biznesowa.

Obiekt DataCode tworzy pierwszy obiekt DataElement. Obiekt elementu w pierwszej kolejności tworzy obiekt html zawierający własne dane, a następnie elementy zagnieżdżone, które powstają rekurencyjnie dopóty, dopóki głębsze elementy istnieją. Następnie wstawiane są elementy równoległe. Każdy element może zawierać dalsze elementy.

### 2.3 Projekt i struktura bazy danych

Diagramy UML mogą również przedstawić strukturę bazy danych. Pozwala to na wizualizację sposobu przechowywania danych na serwerze. Jeżeli używana jest baza relacyjna diagram przedstawia także zależności pomiędzy tabelami. Diagram reprezentujący strukturę bazy danych aplikacji został stworzony i wygenerowany przy użyciu narzędzia Visual Paradigm. Pokazuje on rozkład danych, ich właściwości i zależności między nimi.



Rysunek 6 - Diagram ERD (źródło: opracowanie własne)

Główną tabelą aplikacji jest *course*, która zawiera podstawowe informacje o kursie takie jak tytuł czy kolejność w grupie kursu (zobacz rysunek 6). Tabela *course* jest ściśle związana z tabelą *courseData*, która jest główną częścią przechowującą właściwy tekst kursu. Tabela *courseData* jest powiązana z tabelami *courseDataLine* oraz *courseDataElement*, aby odpowiednio podzielić i przetrzymać te dane. Dane plików obrazu przechowywane są w strukturze *dataImage*. Tabela użytkownika *user* zawiera informacje o nim oraz dane logowania, a w połączeniu z tabelami *role*, *permission* i *role\_permission* pozwala na jego autentykację i autoryzację. Element uwierzytelniający przechowany jest w tabeli *accessToken*. Dane funkcji komentarzy zawarte są w tabeli *comment*, która w połączeniu z tabelą *commentScriptMention* przechowuje treść komentarza oraz załączony fragment kodu kursu. Encje *link* oraz *file* zawierają odpowiednio hiperłącza oraz dane potrzebne do pobrania plików załączanych przez autora do kursu. Jeżeli użytkownik śledzi grupę kursów jego dane są dodane do tabeli *follow*. Tabela *assimilation* przechowuje informacje o przyswojeniu materiału kursów, brak danych oznacza brak asymilacji. Dane quizu przetrzymywane są w dwóch tabelach: *quiz* oraz *quizQuestion*. Tabela *quiz* zawiera referencję do odpowiadającego jej kursu, a tabela *quizQuestion* pytania wraz z odpowiedziami. Wyniki użytkowników zapisane są w strukturze *quizScore*.

### 3. Implementacja.

#### 3.1 Opis użytych w pracy narzędzi i technologii

Strona biznesowa aplikacji została wykonana w języku Java. Jest to język obiektowy oparty na klasach, a jego głównym założeniem jest kompatybilność międzyśrodowiskowa. Oznacza to, że kod jest pisany raz, kompilowany do kodu bajtowego, a później wykonywany przez maszynę wirtualną na środowisku docelowym [8]. Podstawą działania części biznesowej jest framework Spring. Bardzo mocno bazuje on na odwróceniu sterowania, czego przykładem jest wstrzykiwanie zależności. Chociaż można go używać do tworzenia podstawowych aplikacji, jego moduły pozwalają na dużo więcej, na przykład budowanie aplikacji webowych, łączenie z bazami danych, itp. Spring jest otwartym oprogramowaniem, jego dokumentację można znaleźć na stronie „[docs.spring.io](https://docs.spring.io)” [2, 4]. Do ułatwienia interakcji z bazą danych został użyty moduł Spring Data Access zawierający narzędzie Hibernate, które jest frameworkiem umożliwiającym

mapowanie obiektowo-relacyjne (ORM). Hibernate oferuje także język hql (podobny do sql), który pozwala na pisanie zapytań do bazy odwołując się do obiektów w aplikacji, a nie struktury bazy danych. Dokumentacja jest dostępna na „[hibernate.org](http://hibernate.org)” [5]. Obsługa migracji bazy danych jest obsługiwana przez narzędzie Flyway. Pozwala ono na wersjonowanie struktury bazy danych, a także na ustawianie danych, które powinny być zaaplikowane w bazie danych przy każdej zmianie. Baza danych oparta jest na PostgreSQL, otwartym środowisku relacyjnych baz danych. Postgre jest rozwijane przez społeczność od ponad dwudziestu lat. Obsługuje ona nie tylko zapytania SQL, ale także JSON<sup>2</sup>. System baz danych Postgre używany jest głównie do obsługi aplikacji internetowych, mobilnych czy analitycznych [6]. Zastosowanie menadżera pakietów Maven umożliwiło łatwy dostęp do poszczególnych modułów, a także środowiska uruchomieniowego. Maven można znaleźć na stronie „[maven.apache.org](http://maven.apache.org)”.

Strona prezentacji została wykonana przy użyciu języka Javascript, jest to język skryptowy, który jest interpretowany lub kompilowany podczas wykonania programu (kompilacja metodą JIT<sup>3</sup>). Język ten jest dynamicznie typowany. Wraz z html oraz css jest on podstawowym językiem używanym do tworzenia stron www. Biblioteką użytą do utworzenia frontendowej aplikacji jest Vue.js opartą na strukturze model-view-viewmodel<sup>4</sup>. Vue jest używane do tworzenia interaktywnych interfejsów sieciowych. Vue udostępnia sposób na łączenie danych z komponentami widoku. Vue nie jest pełnoprawnym frameworkiem, gdyż jest skupiony na warstwie widoku [3, 7].

### 3.2 Struktura aplikacji

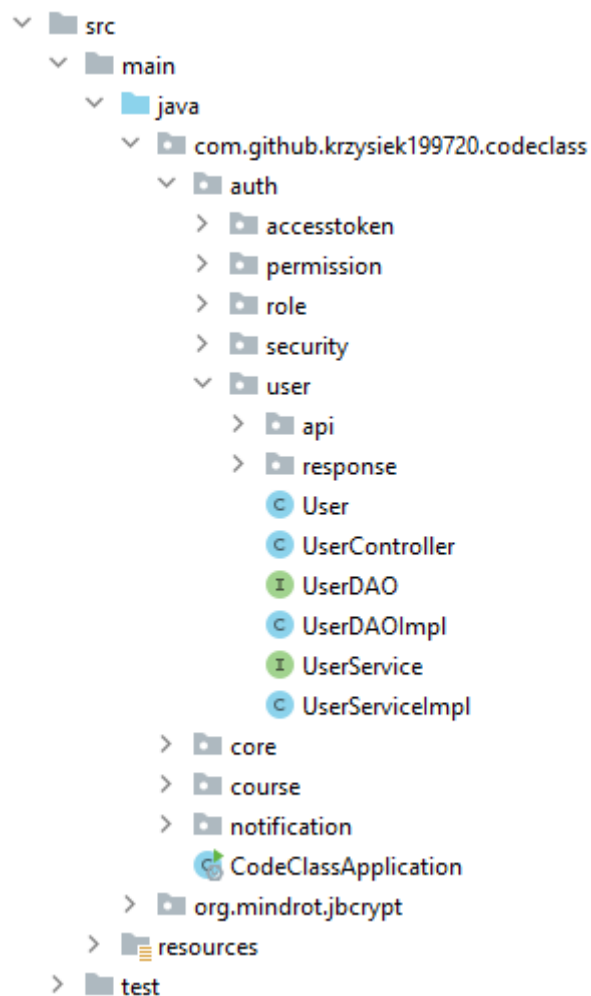
Aplikacja jest podzielona na dwie niezależnie działające części: biznesową oraz prezentacyjną. Każda z nich podąża za przyjętą w projekcie konwencją grupowania kodów źródłowych. Na rys. 7 pokazano strukturę projektu części serwerowej aplikacji.

---

<sup>2</sup> JSON (ang. „JavaScript Object Notation”) – standard formatowania pliku pozwalający na czytelne dla człowieka przekazywanie danych w postaci par atrybutów i wartości oraz tablic.

<sup>3</sup> JIT (ang. „Just-in-time compilation”) – metoda wykonywania kodu komputerowego składająca się z kompilacji kodu źródłowego lub bajtowego bezpośrednio przed jego wykonaniem.

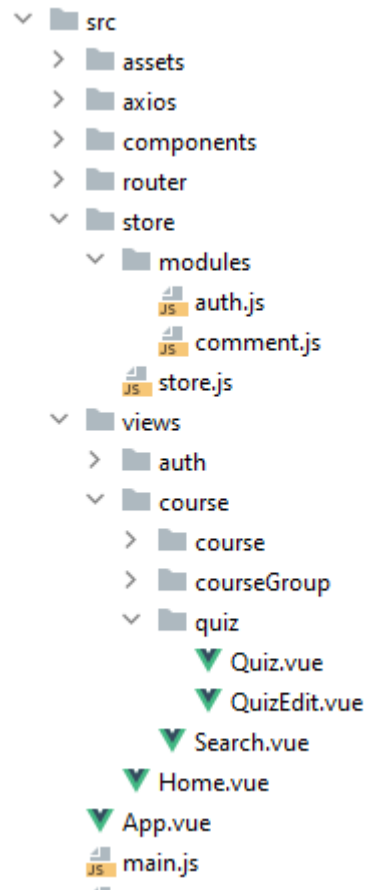
<sup>4</sup> model-view-viewmodel - wzór architektoniczny rozbijający produkcję na części: model, view oraz viewmodel. Model odpowiada za przetrzymywanie danych oraz wykonywanie logiki biznesowej. Viewmodel jest pośrednikiem przekazującym dane z części model do części view. View odpowiada za wyświetlanie danych zawartych w części viewmodel.



Rysunek 7 – Struktura plików serwerowej części aplikacji (źródło: opracowanie własne)

Struktura folderów aplikacji biznesowej (serwerowej) dzieli się na następujące części: *auth*, *core*, *course*, *notification*. Katalog *core* odpowiada za nadanie pozostałym klasom szkieletu, zawiera on głównie interfejsy, klasy abstrakcyjne oraz konfigurację frameworku Spring. Katalog *auth* odpowiada za operacje użytkownika oraz autentykację, *course* definiuje funkcjonalności do obsługi kursów, a *notification* obsługuje powiadomienia. Katalogi *auth*, *course* oraz *notification* rozwijane są na kolejne katalogi oraz strukturę plików. Każda z nich zawiera foldery z klasami reprezentującymi dane żądań oraz odpowiedzi serwera. Niepogrupowane pliki zawierają klasy ORM, kontrolery, klasy usług oraz DAO, a także używane przez nie interfejsy.

Struktura folderów aplikacji strony internetowej (frontendowej części projektu) dzieli się na katalogi zawierające poszczególne elementy strony www wraz z funkcjonalnościami do ich obsługi (zobacz rysunek 8).



*Rysunek 8 - Struktura plików frontendowej części aplikacji (źródło: opracowanie własne)*

Powyższa struktura rozdziela elementy obsługujące połączenia http (katalog *axios*) i mapujące url na widoki (katalog *router*) od elementów właściwych aplikacji znajdujących się w katalogach *views* i *components*. Folder *views* zawiera wszystkie elementy będące widokami stron www dla danych adresów url, a folder *components* zawiera elementy używane w innych komponentach i widokach.

### 3.3 Implementacja części biznesowej aplikacji

Część biznesowa aplikacji składa się z wielu współpracujących obiektów. Jednym z nich jest kontroler, który odpowiada za reagowanie na zapytania ze strony internetowej. Kontroler odpowiada za częściowe sprawdzenie poprawności danych, przede wszystkim ich typów. Funkcje bezpieczeństwa są realizowane poprzez rozszerzanie funkcji kontrolera przy użyciu adnotacji. Framework Spring pozwala na wykrycie adnotacji i odpowiednią obsługę wykonywanej funkcji dzięki możliwości przechwycenia jej wykonywania.

Kolejną warstwą są klasy usług. Odpowiadają one za pełną obsługę zapytań przekazanych przez kontroler. Klasy usług są miejscem, gdzie dokonywane są wszelkie obliczenia biznesowe, zapewniają również dostęp do danych poprzez obiekty DAO.

Warstwa dostępu do bazy danych bazuje na DAO (Data Access Object), gdzie każdy obiekt ma strukturę odpowiednią do kontekstu zapytania kierowanego do bazy danych. Obiekty te są ściśle powiązane z mapowaniem obiektu na tabelę, którą obsługują. Przykładowo *DAO<User>* jest powiązane z klasą *User*, mapującą tabelę *User* w bazie danych. Obiekty DAO zapewniają łatwy dostęp do często używanych funkcjonalności takich jak zapis, czy wyszukanie po kluczu głównym, a także pozostałe funkcje, potrzebne do sprawnego posługiwania się bazą danych.

Framework Spring pozwala na przechwytywanie zaistniałych w aplikacji wyjątków. Funkcjonalność ta została wykorzystana do możliwości obsługi zaistniałych błędów oraz zwracania niestandardowych dla nich odpowiedzi.

Jedną z głównych funkcjonalności mieszczących się w warstwie biznesowej jest funkcjonalność tworzenia kursów. Treść kursu jest obsługiwana przez klasy usług, lecz głównym elementem pozwalającym na przetwarzanie ich zawartości jest analizator składni (ang. "Parser"). Parser umożliwia obsługę przypominającego język xml kodu kursu. Kod ten zawiera wiele znaczników pozwalających na rozdrobnienie kodu programistycznego na linie, a później elementy opisywane. Sposób działania parsera jest oparty o artykuł „Implementing High Performance Parsers in Java” [1]. Analizator podzielony jest na dwie główne części: tokenizer oraz parser. Tokenizer odpowiada za podzielenie kodu na niepodzielne części pierwsze. Indeksy stworzone przez tokenizer



(zobacz rysunek 9) przechowują typ elementu, który będzie używany przez parser do przetworzenia struktury i utworzenia obiektu.

```
public class Index {  
    int position;  
    int length;  
    ElementType type;  
}
```

Rysunek 9 – Klasa index (źródło: opracowanie własne)

Indeks zawiera także położenie elementu w danych źródłowych oraz jego długość. Pozwala to na wydajne indeksowanie zawartości kodu źródłowego, gdyż nie wymaga tworzenia wielu kopii danego fragmentu. Przetworzenie całego źródła powoduje stworzenie tablicy indeksów w kolejności ich występowania w danych.

Aplikacja rozróżnia typ ciągu znaków stosowany do przekazania tekstu do odbiorcy, typ image zawierający lokalny link do obrazka oraz typ tag (znacznik) obsługujący strukturę kodu kursu. Typ tag może zawierać parametry pozwalające lepiej dostosować wygląd kodu do potrzeb autora. Przykładem parametru jest dodanie wcięcia na początku linii kodu. Element może nie posiadać parametrów. Tokenizer nie odpowiada za dobrą strukturę kodu kursu, ale tylko za poprawną składnię każdego elementu. Rozpoznaje on elementy składniowe oraz ich parametry. Najmniejszy błąd składniowy uniemożliwi kontynuację pracy tokenizera powodując błąd.

Przykładem działania tokenizera jest rozpoznawanie tagu <code> (zobacz rysunek 10).

```

111 switch(next){
112     case 'c':{
113         sb.append('c');
114         char c;
115         while(data.hasNext()){
116             c = data.next();
117             if(c != '>')
118                 sb.append(c);
119             else
120                 break;
121         }
122         if(data.get() != '>'){
123             state = ParserState.ERROR;
124             break;
125         }
126         if(!sb.toString().equals("code")){
127             state = ParserState.ERROR;
128             break;
129         }
130
131         indexBuffer.addElement(startingPosition,
132                               length: data.getPosition()-startingPosition+1,
133                               isEnding ? ElementType.CODE_END : ElementType.CODE);

```

Rysunek 10 – Wykrywanie znacznika code (źródło: opracowanie własne)

Funkcja jest uruchamiana po wykryciu znaku rozpoczęcia znacznika (mniejszy niż - "<"), następnie iteruje dane źródłowe po jednym znaku tekstowym do momentu wykrycia znacznika zamknięcia (większy niż - ">"). Porównywana jest zawartość realna do przewidywanej, jeżeli są zgodne pozycja oraz długość elementu są obliczane i dodane do listy indeksów. Przypadek niezgodności danych powoduje ustawienie flagi błędu oraz zakończenie funkcji. Typ tagu zawsze rozpoczynany jest przez znak "<". W celu użycia go jako zwykłego tekstu użytkownik musi to zaznaczyć poprzedzając go znakiem "\". Wymusza to na tokenizerze rozpoznawanie go jako część tekstu. Znak backslash jest pomijany w końcowym tekście. W analogiczny sposób obsługiwane są również inne znaczniki.

Kod kursu podzielony na indeksy przekazywany jest do części odpowiadającej za ich analizę i stworzenie struktury odpowiedniej dla aplikacji. Parser iteruje tablicę indeksów analizując kolejne elementy rozróżniając je po ich typie, zobacz tabelę 1.

Tabela 1. Typy elementów, z których składa się kod kursu

|                     |  |
|---------------------|--|
| UNKNOWN             | Może wystąpić tylko w przypadku błędu.           |
| TEXT                | Element jest typu ciągu znaków.                  |
| CODE                | Rozpoczyna sekcję kodu.                          |
| CODE_END            | Kończy sekcję kodu.                              |
| LINE                | Rozpoczyna nową linię kodu.                      |
| LINE_INDENT         | Ustala przesunięcie linii względem jej początku. |
| LINE_END            | Kończy linię kodu.                               |
| ELEMENT             | Rozpoczyna nowy element wyjaśniający.            |
| ELEMENT_DESCRIPTION | Parametr ustalający opis elementu.               |
| ELEMENT_END         | Zamyka element wyjaśniający.                     |
| IMAGE               | Wstawia element zawierający rysunek.             |
| IMAGE_ID            | Ustala parametr id rysunku.                      |

Jeśli typ elementu jest użyty w niedozwolonym miejscu powoduje to błąd. Przykładowymi elementami są `<code>` oraz `<line>`. Znacznik `<code>` pozwala parserowi wykryć intencje użytkownika do zazęczenia części zawierającej kod programistyczny, a element `<line>` odpowiada za grupowanie tego kodu w linie. Jeśli użytkownik użyje znacznika `<line>` przed znacznikiem `<code>` wystąpi błąd. Niektóre typy tagu muszą posiadać zamknięcie swojego obszaru. Pozwala to na łatwe wykrycie błędów oraz tworzenie docelowego rezultatu. Przykładem może być `</line>` sugerujący zakończenie linii. Typ elementu obrazkowego jest możliwy do użycia tylko poza elementami kodu programistycznego, więc jest obsługiwany tak, aby nie został użyty w sekcji kodu. Wszelkie niezgodności z założeniami parsera powodują błąd i zatrzymanie działania parsera. Część znaczników ma możliwość ustalenia ich parametrów. Przyporządkowanie wartości parametru do jego elementu jest obsługiwane w parserze, gdy zostanie wykryty element mogący posiadać parametry. Na rysunku 11 przedstawione zostało działanie analizatora w przypadku wykrycia znacznika `<line>`. Parser sprawdza kolejny element, jeśli jest on parametrem musi on pasować do poprzedzającego go znacznika (w tym przypadku do znacznika `<line>` przypisany jest parametr *indent*, który

odpowiada za przesunięcie linii). W przypadku użycia błędnego typu parametru względem poprzedzającego go znacznika, parser zwróci błąd.

```
520 // check for indent
521 try{
522     if(indexBuffer.indexes.get(resultIndex+1).type == ElementType.LINE_INDENT){
523         ++resultIndex;
524         if(indexBuffer.indexes.get(resultIndex).length != 0){
525             Index indexIndent = indexBuffer.indexes.get(resultIndex);
526             Integer indent = Integer.parseInt(
527                 new String(data.getData(), indexIndent.position, indexIndent.length)
528             );
529             line.setIndent(indent);
530         }
531     }
532     if(index.type == ElementType.IMAGE){
533         resultState = ParserResultState.ERROR_UNEXPECTED_TAG;
534         return null;
535     }
536 }catch (IndexOutOfBoundsException exception){
537     resultState = ParserResultState.ERROR_MISSING_LINE_END;
538     return null;
539 }catch (NumberFormatException exception){
540     resultState = ParserResultState.ERROR_INDENT;
541     return null;
542 }
```

*Rysunek 11 – Fragment kodu parsera sprawdzający istnienie parametru indent*  
(źródło: opracowanie własne)

Kolejnym typem elementu może być ciąg znaków. Można go używać zarówno w kodzie programu jak i poza nim. Użycie go na zewnątrz powoduje to, że jest on traktowany jako osobny obiekt. Kod programistyczny może zawierać tekst tylko w określonych miejscach. Niedozwolone jest użycie go między znacznikami określającymi rozkład linii. Poprawnym miejscem na jego użycie jest natomiast wewnątrz linii. Użycie tekstu bez znacznika elementu powoduje rozpatrzenie go jako elementu nieposiadającego opisu. Element jest częścią linii zawierającą dane do wyświetlenia, dlatego wymagane jest, aby zawierał tekst. Możliwe jest dodanie komentarza do jego wnętrza. Rysunek 12 przedstawia przykład poprawnie utworzonego kodu kursu. Przykład pokazuje, jak rozpisać nagłówki pętli *for* używając opisów do wytłumaczenia działania poszczególnych jego elementów.

```

1 Przykład kodu kursu tworzego przez autora
2 <code>
3     <line indent="0">
4         for(
5             <element desc="inicjacja zmiennej i">int i=0; </element>
6             <element desc="warunek zakończenia pętli">i<10;</element>
7             <element desc="zmiana wartości i po iteracji">++i</element>
8         )
9     </line>
10 </code>

```

*Rysunek 12 – Przykład kodu kursu (źródło: opracowanie własne)*

### 3.4 Implementacja części frontendowej aplikacji

Działanie części frontendowej aplikacji oparte jest na modułach - elementach niezależnych zarządzających własnym zestawem danych. Moduł składa się z części html pozwalającej na nadanie elementowi jego struktury w przeglądarce internetowej, części kodu Javascript pozwalającego na wykonywanie działań na dostępnych danych, obsługę zdarzeń, itp. Zawiera on także część css pozwalającą na nadanie elementom html odpowiedniego wyglądu. Vue pozwala na tworzenie modułów nieposiadających struktury html. Elementy takie są używane jako globalne przekazywacze danych i zdarzeń. Vue pozwala programiście na dowolność użycia stworzonych przez niego modułów, co umożliwia łatwe rozwiązanie problemu z rekurencją elementów kursu.

Część wyświetlająca kod kursu wymagała zaawansowanego zagnieżdżania elementów, gdyż powinna ona umożliwiać zaznaczanie przez autora dowolnej części linii poprzez dodanie do nich opisu. Wymagało to odpowiedniego dobierania elementów wymagających zagnieżdżenia oraz będących niezagnieżdżoną kontynuacją. Struktura elementu zagnieżdżanego przedstawiona na rysunku 13 pozwala na rozróżnienie elementów zagnieżdżonych od elementów równoległych (występujących na tym samym poziomie).

```

3 <div class="element" @mouseover.stop.self="onHover"
4   @mouseleave.stop.self="hovering = false"
5   :class="{hovered: hovering}">
6   <div class="desc-tooltip" v-if="elem.description !== null">
7     <div class="shadow">
8       {{elem.description}}
9     </div>
10  </div>
11  <template>{{elem.data}}</template>
12  <dataElement v-if="deepeerElem !== null"
13    :data="deepeerElem.list"
14    @hov="hovering = false"></dataElement>
15 </div>
16 <dataElement v-for="elem in list" :data="elem.list"
17   :key="elem.order"
18   @hov="hovering = false"></dataElement>

```

*Rysunek 13 – Kod źródłowy elementów (tagów), z których składa się kod programu wyjaśnianego w kursie (źródło: opracowanie własne)*

Element pobiera dane z obiektu rodzica, po czym dzieli je na dwie listy elementów: listę elementów zagnieżdżonych oraz listę elementów równoległych. Kod elementu wypełnia pole danych oraz pole opisu, jeśli jest ono dostępne. Następnie sprawdzane jest istnienie elementów niżej zagnieżdżonych. Elementy takie są wklejane na stronę rekurencyjnie. Kolejnym krokiem jest wyświetlenie elementów niezagnieżdżonych. Listy używane przez element tworzone są na podstawie danych wejściowych, które są przetwarzane przez wykonywaną przez element funkcję. Pierwszym i najważniejszym etapem podziału jest wykrywanie elementów o tej samej głębokości, zobacz rysunek 14.

```

52   for (; i < list.length; i++) {
53       if (list[i].depth === initialDepth) {
54           result.push({
55               list: list.slice(start, i),
56               ord: list[start].order,
57               dep: list[start].depth
58           })
59           start = i
60       }
61   }
62   if (start !== i) {
63       result.push({
64           list: list.slice(start, i),
65           order: list[start].order,
66           dep: list[start].depth
67       })
68   }

```

Rysunek 14 – Podział danych na listy względem zagnieżdżenia (źródło: opracowanie własne)

Lista elementów równoległych wykorzystywana jest do inicjacji tworzenia elementów niezagnieżdżonych. Posiadanie przez element elementów potomnych (zagnieżdżonych) sprawdzane jest po ustaleniu listy elementów równoległych (zobacz rysunek 15).

```

72   if (result[0].dep > initialDepth) {
73       this.deeperElem = result[0]
74       result.splice( start: 0, deleteCount: 1)
75   }

```

Rysunek 15 – Sprawdzenie głębokości następnego elementu (źródło: opracowanie własne)

Sprawdzana jest tutaj głębokość następnego elementu. Jeżeli element jest położony głębiej od aktualnego zostaje oznaczany jako element do wyświetlania rekurencyjnego.

### 3.5 Opis uruchomienia aplikacji

Proces instalacji i uruchomienia aplikacji przedstawiony jest na podstawie systemu operacyjnego Ubuntu 20.04, zainstalowanego na maszynie nie będącej

maszyną wirtualną. Używanym użytkownikiem jest *ubuntu*, a dane źródłowe programu zostały umieszczone w folderze domowym: */home/ubuntu/cc*. Kod strony biznesowej umieszczono w folderze *codeclass*, a kod warstwy prezentacji w folderze *codeclassfront*. Pierwszym krokiem instalacji jest zalogowanie jako administrator i sprawdzenie dostępności aktualizacji repozytoriów oraz składników systemu. Jeżeli aktualizacje istnieją, należy je zainstalować w następujący sposób (zobacz rysunek 16).

```
sudo su
apt update
apt upgrade
```

*Rysunek 16 – Aktualizacja składników systemu (źródło: opracowanie własne)*

Następnie przeprowadzana jest instalacja systemu bazodanowego *PostgreSql* oraz zmiana użytkownika na *postgres*, zobacz rysunek 17.

```
apt install postgresql postgresql-contrib
sudo -i -u postgres
```

*Rysunek 17 – Instalacja serwera bazy danych (źródło: opracowanie własne)*

Kolejnym etapem jest sprawdzenie działania usługi bazy i nasłuchiwanie przez nią na porcie 5432, przedstawione na rysunku 18.

```
psql
\q

lsof -i:5432
service postgresql status
```

*Rysunek 18 – Sprawdzenie działania usługi serwera baz danych (źródło: opracowanie własne)*

W następnym kroku uruchamiana jest konsola systemu bazy danych, wprowadzane są dane nowego użytkownika oraz tworzona jest nowa baza *codeclass* (zobacz rysunek 19). Następnie przeprowadzana jest próba zalogowania na nowo utworzonego użytkownika oraz wylogowanie konta systemowego *postgres*.



```
psql

create user codeclass password '321GiftedSidewalks' superuser;
create database codeclass owner codeclass;

\q
psql -h localhost -U codeclass -d codeclass
\q

exit
```

*Rysunek 19 – Utworzenie użytkownika i bazy danych (źródło: opracowanie własne)*

Pomyślna konfiguracja systemu bazodanowego pozwala na rozpoczęcie instalacji części biznesowej aplikacji. Pierwszym etapem jest instalacja środowiska Java. Wykorzystano w tym celu *Java Development Kit* w wersji 13 (zobacz rysunek 20). Po jego zainstalowaniu należy wylogować użytkownika *root*.

```
apt install openjdk-13-jre-headless

exit
```

*Rysunek 20 – Instalacja Java JDK 13 (źródło: opracowanie własne)*

Plikowi wykonywalnemu narzędzia Maven nadawane są uprawnienia do wykonywania. Następnie uruchamiana jest edycja pliku konfiguracyjnego projektu (zob. rysunek 21).

```
cd codeclass
sudo chmod +x ./mvnw

nano ./src/main/resources/application-dbc cred.properties
```

*Rysunek 21 – Nadanie uprawnień wykonywania dla pliku Maven oraz otwarcie pliku ustawień serwera danych (źródło: opracowanie własne)*

Plik *application-dbc cred.properties* zawiera dane konfiguracyjne pozwalające na połączenie z bazą danych, użytkownik powinien edytować go zmieniając adres *url* bazy danych, użytkownika oraz hasło. Przykład konfiguracji został przedstawiony na rysunku 22.

```
spring.datasource.url=jdbc:postgresql://127.0.0.1:5432/codeclass
spring.datasource.username=codeclass
spring.datasource.password=321GiftedSidewalks
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
```

*Rysunek 22 - Zmiana ustawień serwera danych (źródło: opracowanie własne)*

Następnie wykonywana jest weryfikacja, kompilacja oraz pakowanie kodu źródłowego (zobacz rysunek 23). Kończącym etapem jest włączenie aplikacji.

```
./mvnw verify
./mvnw compile
./mvnw package

java -jar target/code-class-1.0.jar
```

*Rysunek 23 - Kompilacja i uruchomienie serwera danych (źródło: opracowanie własne)*

Ostatnią częścią instalacji jest uruchomienie serwera www. Zostanie w tym celu wykorzystany serwer *Apache2*. Na rysunku 24 przedstawiono proces instalacji serwera *Apache2* oraz jego podstawową konfigurację. Do pliku konfiguracyjnego *000-defaults.conf* w zakresie jego głównego elementu dodany zostaje podany poniżej fragment kodu.

```
sudo apt install apache2
sudo nano /etc/apache2/sites-enabled/000-defaults.conf

<Directory /var/www/html/>
    Require all granted
    Order allow,deny
    Options Indexes FollowSymLinks
    AllowOverride All
    allow from all
</Directory>
```

*Rysunek 24 - Instalacja serwera www (źródło: opracowanie własne)*

Po zmianie ustawień konieczne jest ponowne uruchomienie usługi, zobacz rysunek 25. Dodatkowo użytkownik może sprawdzić poprawne nasłuchiwanie przez Apache2 na portach www (domyślnie port 80).

```
sudo systemctl restart apache2  
  
lsof -i:80
```

*Rysunek 25 - Sprawdzenie działania serwera www (źródło: opracowanie własne)*

Kolejnym etapem (zob. rys. 26) jest zainstalowanie środowiska *node.js* oraz menadżera pakietów *node*. Ostatnia komenda instaluje program *yarn* w środowisku *node*.

```
sudo apt install nodejs  
sudo apt install npm  
  
npm install --global yarn
```

*Rysunek 26 - Instalacja node.js oraz yarn (źródło: opracowanie własne)*

Jeżeli niemożliwe jest zainstalowanie *yarn* przez problemy z uprawnieniami, zalecane jest użycie sekwencji komend naprawiających problem (zobacz rysunek 27). Jeśli problem powraca naprawa może sprowadzać się do dodania ścieżki uruchomieniowej do zmiennej *PATH*.

```
mkdir ~/.npm-global  
npm config set prefix '~/.npm-global'  
export PATH=~/.npm-global/bin:$PATH  
source ~/.profile
```

*Rysunek 27 - Naprawa problemu dostępu podczas instalacji pakietów (źródło: opracowanie własne)*

Następnie wykonywana jest instalacja potrzebnych modułów oraz kompilacja aplikacji używając poleceń przedstawionych na rysunku 28.

```
cd cc/codeclassfront  
  
yarn install  
yarn build
```

*Rysunek 28 - Instalacja zależności projektu oraz kompilacja aplikacji strony internetowej  
(źródło: opracowanie własne)*

Zbudowana aplikacja może zostać przeniesiona do folderu pozwalającego serwerowi www na jego publikację. Poprawne działanie aplikacji umożliwia utworzenie pliku .htaccess oraz dodanie do niego podanego poniżej na rysunku 29 kodu.

```
sudo mv dist/* /var/www/html/  
cd /var/www/html  
sudo nano .htaccess  
  
<IfModule mod_rewrite.c>  
  RewriteEngine On  
  RewriteBase /  
  RewriteRule ^index\.html$ - [L]  
  RewriteCond %{REQUEST_FILENAME} !-f  
  RewriteCond %{REQUEST_FILENAME} !-d  
  RewriteRule . /index.html [L]  
</IfModule>
```

*Rysunek 29 - Konfiguracja udostępniania strony www (źródło: opracowanie własne)*

Ostatnim etapem jest nadanie serwerowi www własności do aplikacji (zobacz rysunek 30).

```
cd ..  
sudo chown -R www-data:www-data html
```

*Rysunek 30 - Zmiana uprawnień folderu strony (źródło: opracowanie własne)*

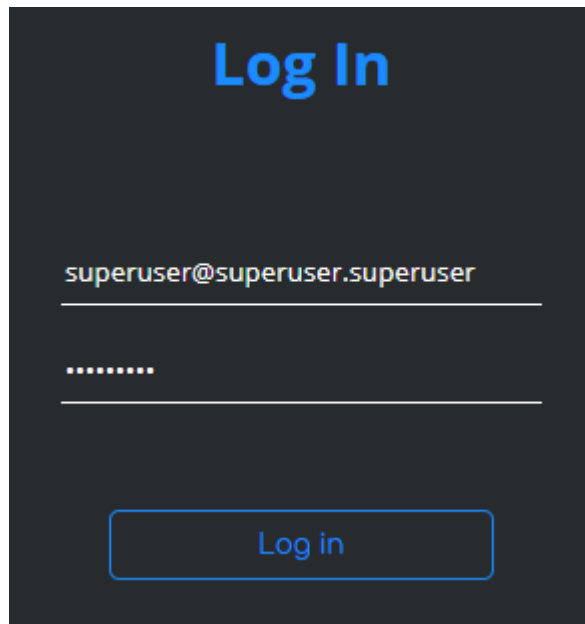
Aplikacja domyślnie ustawiona jest, aby działać poprawnie w obszarze jednej maszyny. Uruchomienie jej w formie serwisu zewnętrznego wymaga zmiany obsługiwanego IP w plikach źródłowych przed kompilacją. Aplikacja biznesowa posiada parametr IP ustawiony w pliku konfiguracyjnym, usunięcie go powoduje uruchomienie jej na wszystkich dostępnych adresach. Zmiana adresu IP serwera danych wymaga aktualizacji

adresu połączeń strony internetowej, można go zmienić w pliku konfiguracyjnym narzędzie axios.

#### 4. Prezentacja działania aplikacji.

##### Autentykacja użytkownika

Rejestracja w aplikacji pozwala na dodanie użytkownika do systemu. Nowy użytkownik otrzymuje rolę zwykłego użytkownika pozwalającą mu tylko na podstawowe działania na stronie. Aplikacja przyznaje dostęp do zasobów poprzez weryfikację kodu przesyłanego przez użytkownika. Kod ten jest unikalnym ciągiem znaków. Pozyskanie tokenu odbywa się poprzez proces logowania (zobacz rysunek 31), użytkownik podaje email oraz hasło, a w zamian przesyłany jest mu token. Proces logowania zwraca także inne potrzebne do działania dane, takie jak okres życia tokenu, czy listę pozwoleń (zobacz rysunek 32).



Rysunek 31 – Formularz logowania (źródło: opracowanie własne)

```

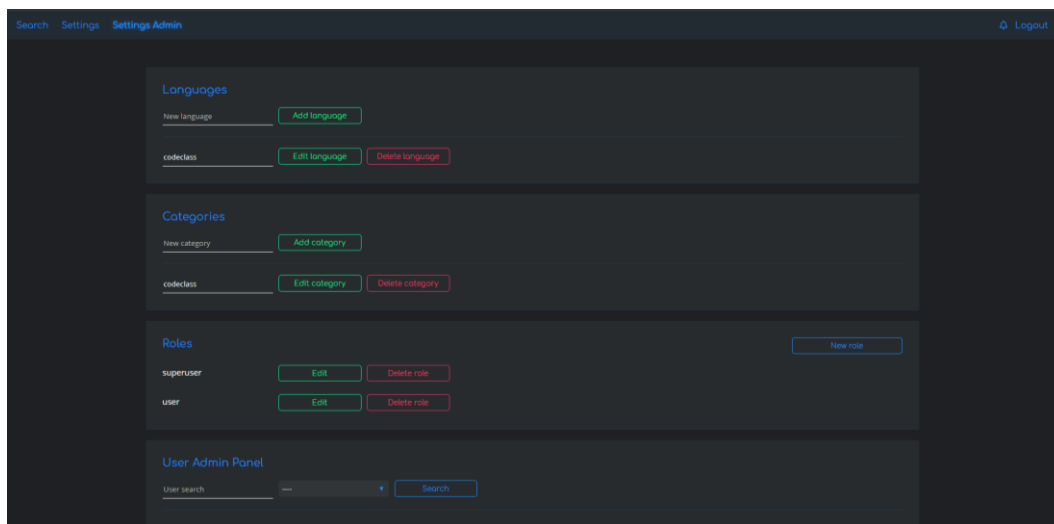
{token: "e5702b6e-b26c-4293-8b99-9caa0e5da664",
 email: "superuser@superuser.superuser"
 expires: "2021-02-09T14:09:00.9972721"
 firstName: "superuser"
 isAdmin: true
 lastName: "superuser"
 permissions: [{id: 7, group: "user", name: "up
 roleId: 1
 roleName: "superuser"
 token: "e5702b6e-b26c-4293-8b99-9caa0e5da664"
 userId: 1

```

Rysunek 32 - Dane zwrotne logowania (źródło: opracowanie własne)

### Panel ustawień administratora

Jeżeli zalogowany użytkownik ma rolę administratora oraz odpowiednie pozwolenia, zostaje mu udostępniona strona zarządzania serwisem, zobacz rysunek 33.



Rysunek 33 - Wygląd strony ustawień administratora (źródło: opracowanie własne)

Na stronie ustawień administratora znajduje się zakładka umożliwiająca zarządzanie dostępnymi językami programowania i kategoriami, zob. rys. 34.

**Languages**

codeclass  Add language

codeclass  Edit language Delete language

**Categories**

codeclass  Add category

codeclass  Edit category Delete category

Rysunek 34 - Zarządzanie językami programowania i kategoriami (źródło: opracowanie własne)

Administrator może także zarządzać rolami (zobacz rysunek 35). Pozwala mu to na nadawanie ról użytkownikom, tworzenie nowych, a także edytowanie. Panel użytkowników pozwala na wyszukiwanie użytkowników przy użyciu roli, emaila czy imienia i nazwiska.

**Roles** New role

|           |                       |                              |
|-----------|-----------------------|------------------------------|
| superuser | <button>Edit</button> | <button>Delete role</button> |
| user      | <button>Edit</button> | <button>Delete role</button> |

**User Admin Panel**

User search  ▼ Search

superuser superuser superuser@superuser.superuser superuser ▼

Rysunek 35 - Zarządzanie rolami (źródło: opracowanie własne)

Edycja oraz tworzenie roli składają się z kilku głównych elementów. Przyciski nawigujące pozwalają na zapis oraz opuszczenie edycji. Edycja roli sprowadza się do wybrania odpowiedniej nazwy, przypisania roli flagi administratora oraz wybrania dla niej uprawnień. Uprawnienia podzielone są na grupy odpowiednio opisujące ich działanie, np. obsługa kursu czy roli (zobacz rysunek 36).

Rysunek 36 - Edycja roli (źródło: opracowanie własne)

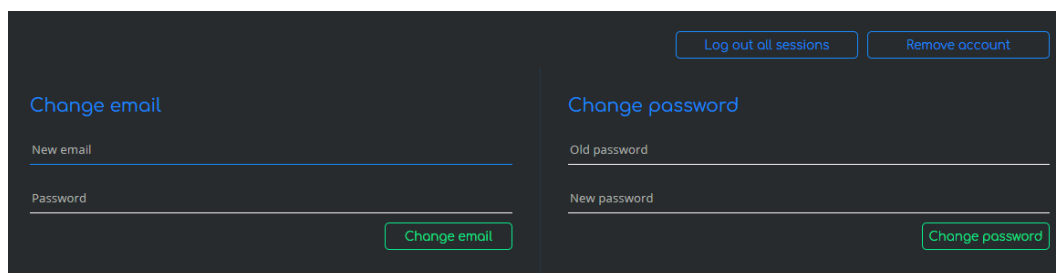
## Panel ustawień użytkownika

Każdy zalogowany użytkownik dostaje dostęp do strony ustawień osobistych, zob. rys. 37.

Rysunek 37 - Wygląd strony ustawień użytkownika (źródło: opracowanie własne)

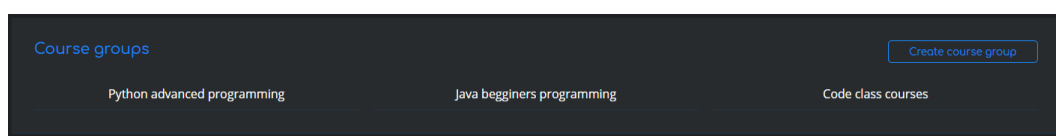


Zawiera ona elementy ustawień pozwalające na zmianę adresu email, hasła, a także na wylogowanie wszystkich sesji i usunięcie konta, zobacz rysunek 38.

The screenshot shows a dark-themed user settings interface. At the top right, there are two buttons: "Log out all sessions" and "Remove account". The main area is divided into two columns. The left column is titled "Change email" and contains two input fields labeled "New email" and "Password", followed by a green "Change email" button. The right column is titled "Change password" and contains two input fields labeled "Old password" and "New password", followed by a green "Change password" button.

*Rysunek 38 - Zmiana danych użytkownika (źródło: opracowanie własne)*

Jeżeli użytkownik posiada prawa do tworzenia kursów zakładka ustawień zawiera także spis wszystkich stworzonych przez niego grup kursów oraz możliwości stworzenia nowej grupy kursów. Wygląd listy jest pokazany na rysunku 39.

The screenshot shows a dark-themed "Course groups" page. At the top right is a button labeled "Create course group". Below the header, there is a list of three course groups: "Python advanced programming", "Java beginners programming", and "Code class courses". Each group name is followed by a horizontal line, likely representing a list of courses within that group.

*Rysunek 39 - Lista własnych kursów (źródło: opracowanie własne)*

### **Tworzenie kursów**

Tworzenie grupy kursów sprowadza się do wprowadzenia jej nazwy, zobacz rysunek 40.

The screenshot shows a dark-themed "Create course group" form. It features a single input field with the placeholder text "Code class courses". At the bottom right of the form is a blue "Save" button.

*Rysunek 40 - Tworzenie grupy kursów (źródło: opracowanie własne)*

Grupa kursów może zawierać wiele kursów. Przedstawione są one w postaci listy zawierającej najważniejsze informacje na ich temat (zobacz rysunek 41). Kursy są przedstawione w kolejności ustawionej przez autora. Jeśli autor przegląda stronę grupy kursu dostaje on możliwość dodania do niej kolejnego kursu oraz edycji nazwy grupy.

| Code class courses |                         |              |           |           |            |
|--------------------|-------------------------|--------------|-----------|-----------|------------|
| 0                  | Code class course       | BEGINNER     | codeclass | codeclass | 02/08/2021 |
| 1                  | Code class intermediate | INTERMEDIATE | codeclass | codeclass |            |
| 2                  | Codeclass advanced      | ADVANCED     | codeclass | codeclass |            |

*Rysunek 41 - Widok grupy kursów (źródło: opracowanie własne)*

Tworzenie kursu automatycznie przypisuje nowo zakładany kurs do grupy, w której autor rozpoczął jego tworzenie. Twórca kursu przypisuje mu nazwę, kolejność lekcji w grupie, a także złożoność oraz język i kategorię (zobacz rysunek 42).

Create course

Code class course

Group order 0

Complexity BEGINNER

Language codeclass

Category codeclass

Save

*Rysunek 42 - Tworzenie kursu (źródło: opracowanie własne)*

### Edycja podstawowych informacji dotyczących kursu

Utworzenie kursu przekierowuje aplikację do dalszej edycji nowo utworzonego kursu (zobacz rysunek 43).

The screenshot shows a dark-themed web interface for editing a course. At the top, there is a navigation bar with 'Search', 'Settings', 'Settings Admin', and a 'Logout' button. The main content area is titled 'Edit course' and includes several sections:

- Course Details:** Fields for 'Title' (with 'Code class course' below it), 'Complexity' (set to 'BEGINNER'), 'Language' (set to 'codeclass'), and 'Category' (set to 'codeclass'). There is also a 'Group order' field set to '6'. Buttons for 'Edit data', 'Edit quiz', 'Return to course', and 'Save details' are present.
- Edit useful links:** A table with two columns: 'Displayed text' and 'Link'. It contains one entry for 'Google' with the link 'https://www.google.com'. Buttons for 'Add link' and 'Remove' are shown.
- File Upload:** A section with 'Displayed text' and 'File' fields. It includes buttons for 'Choose file', 'Add file', and 'Remove'.

*Rysunek 43 - Wygląd strony edycji kursu (źródło: opracowanie własne)*

Użytkownik może ponownie zmienić ustawione wcześniej podstawowe dane kursu (zobacz rysunek 44). Dostaje także dostęp do edycji zawartości kursu oraz quizu, które są opisane poniżej.

This screenshot shows the 'Edit course' page with a focus on the basic course data section. The fields and buttons are similar to the previous screenshot, but the 'Group order' field is now set to '0'. The 'Save details' button is highlighted in green.

*Rysunek 44 - Edycja podstawowych danych kursu (źródło: opracowanie własne)*

Autor dostaje możliwość dodania linków oraz plików, zobacz rysunek 45. Zamieszczone odnośniki do stron oraz pliki są możliwe do pobrania na właściwej stronie kursu. Załączniki te mogą być dowolnie usuwane.

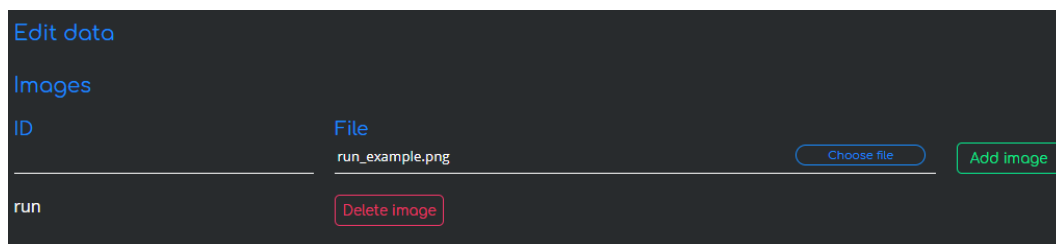
Rysunek 45 - Edycja linków i plików kursu (źródło: opracowanie własne)

## Edycja zawartości kursu

Strona edycji zawartości kursu (zob. rys. 46) pozwala na modyfikację właściwej treści kursu.

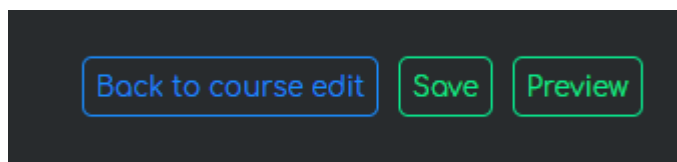
Rysunek 46 - Wygląd strony edycji treści kursu (źródło: opracowanie własne)

Aplikacja daje możliwość dodania przez autora obrazka oraz nadanie mu identyfikatora, pokazane na rysunku 47. Identyfikator może być dowolnym ciągiem znaków unikalnym w obrębie kursu. Id zostanie użyte przez niego do zamieszczenia obrazka w kursie.



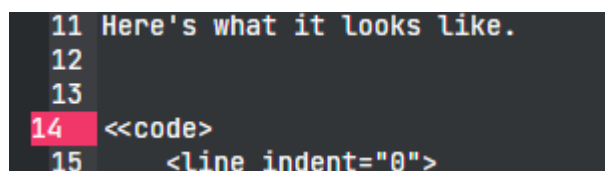
Rysunek 47 - Dodawanie obrazków do kursu (źródło: opracowanie własne)

Możliwe jest opuszczenie strony edycji kursu i jego zapis przy użyciu odpowiednich przycisków znajdujących się w górnej części strony, zob. rys. 48. Przycisk Preview umożliwia sprawdzenie poprawności kodu przed jego zapisem.



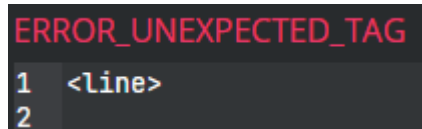
Rysunek 48 - Panel zarządzania zapisem treści kursu (źródło: opracowanie własne)

Edytor kodu kursu pozwala na szybsze znalezienie błędów wykrytych przez parser poprzez zaznaczanie pozycji, na której wystąpił błąd. Sposób zaznaczenia pokazany jest na rysunku 49.



Rysunek 49 - Zaznaczenie błędnej linii kodu (źródło: opracowanie własne)

Jeżeli błąd będzie dotyczył logiki kodu, a nie poprawności składniowej parser zwróci bardziej przejrzysty błąd, taki jak nieoczekiwany tag, czy brak danych. Na rysunku 50 pokazano błąd `ERROR_UNEXPECTED_TAG`, gdyż znacznik `<line>` musi być zawarty w tagu `<code>`, nie może występować samodzielnie.



```
ERROR_UNEXPECTED_TAG
1 <line>
2
```

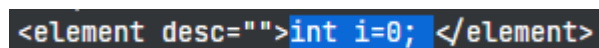
Rysunek 50 - Wyświetlenie treści błędu w kodzie (źródło: opracowanie własne)

Edytor zawiera szereg przycisków ułatwiających pisanie kodu. Pozwalają one na automatyczne dodanie tagu w miejscu kursora (zobacz rysunek 51).



Rysunek 51 - Panel dodawania znaczników do kodu (źródło: opracowanie własne)

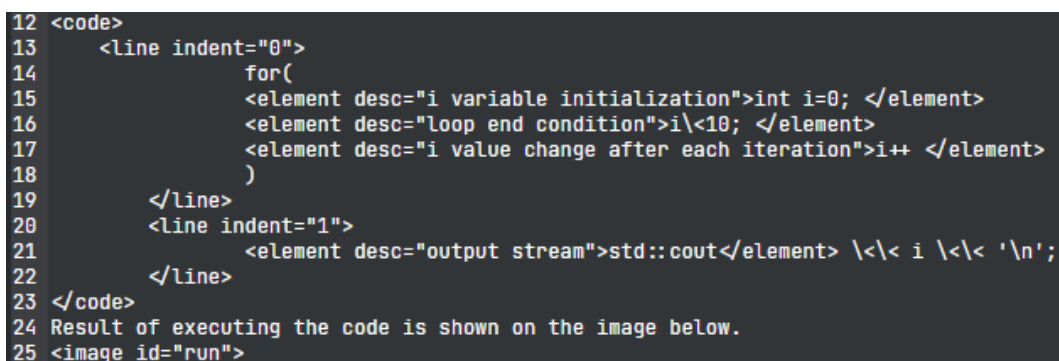
Jeśli położenie karetki było zaznaczeniem element zostanie dodany zagnieżdżając zaznaczenie wewnątrz, tak jak to pokazano na rysunku 52.



```
<element desc="">int i=0; </element>
```

Rysunek 52 - Wstawianie znaczników podczas zaznaczenia (źródło: opracowanie własne)

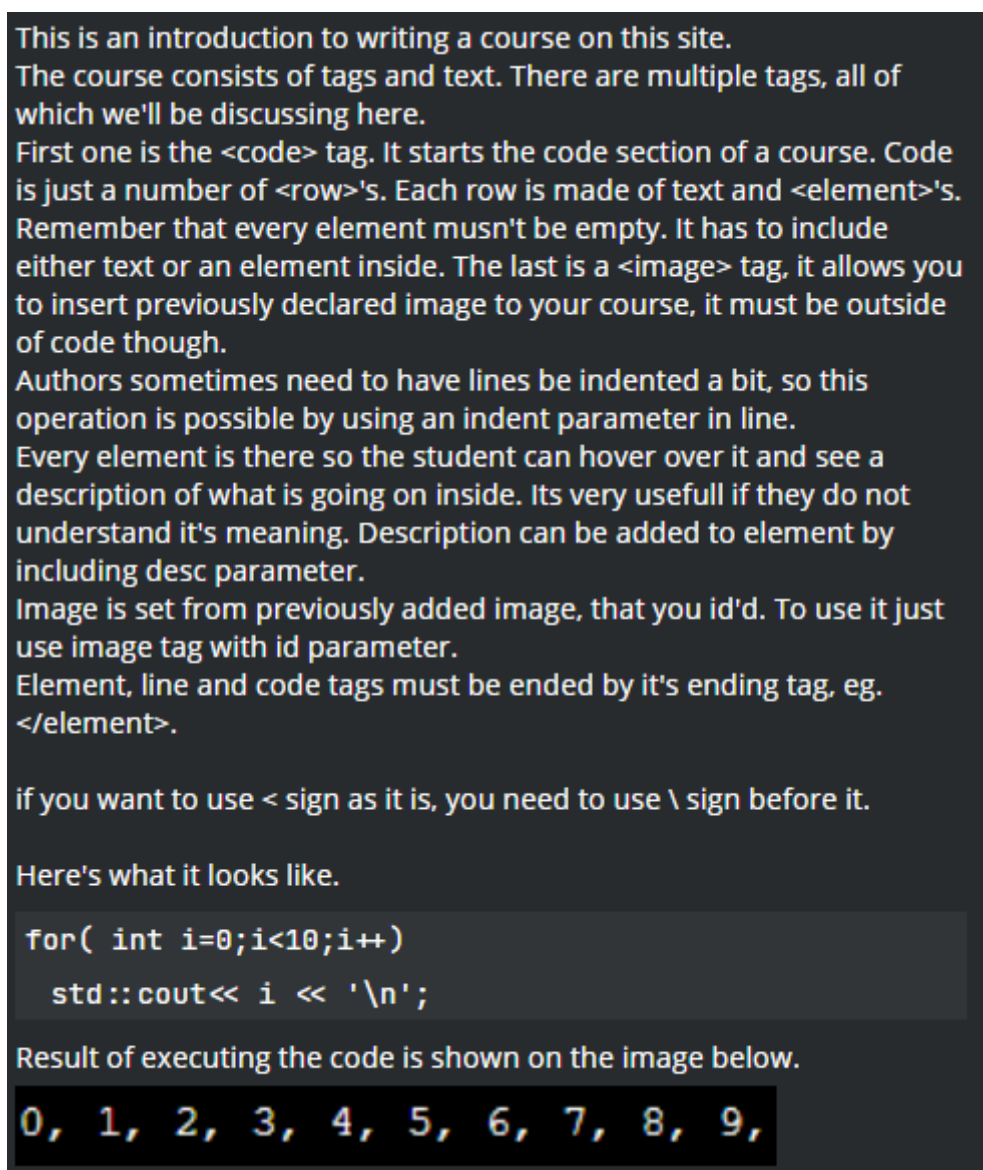
Poprawnie napisany kod kursu zawierać powinien treści tłumaczące, obrazki przedstawiające diagramy czy wyniki działania aplikacji, ale przede wszystkim odpowiednio napisany kod wyjaśniający (zob. rys. 53). Sekcja kodu powinna składać się z rozpoczęcia sekcji kodu oraz co najmniej jednej linii, gdzie każda linia zawierać może tekst oraz elementy wyjaśniające.



```
12 <code>
13   <line indent="0">
14       for(
15           <element desc="i variable initialization">int i=0; </element>
16           <element desc="loop end condition">i<10; </element>
17           <element desc="i value change after each iteration">i++ </element>
18       )
19   </line>
20   <line indent="1">
21       <element desc="output stream">std::cout</element> \<\< i \<\< '\n';
22   </line>
23 </code>
24 Result of executing the code is shown on the image below.
25 <image id="run">
```

Rysunek 53 - Przykładowy kod kursu (źródło: opracowanie własne)

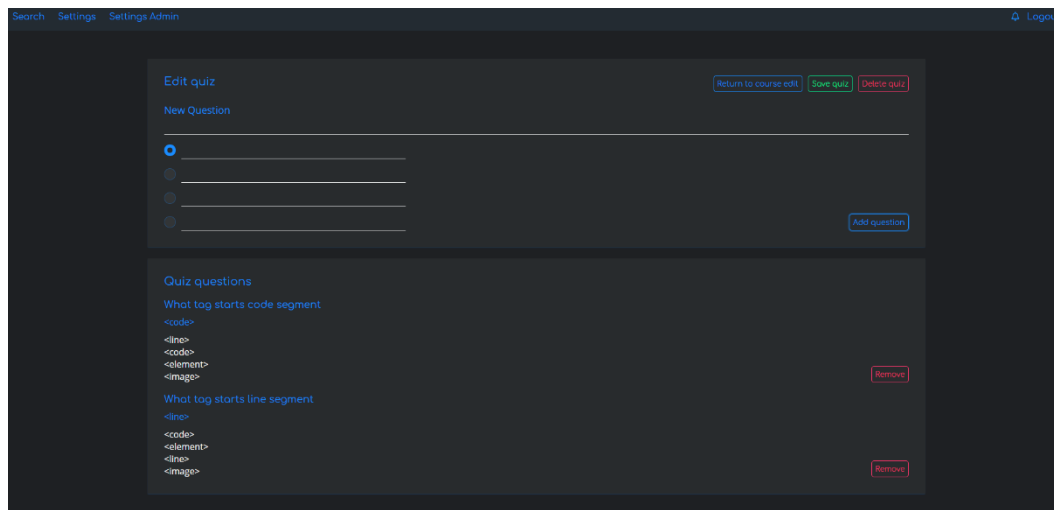
Działanie kodu, którego część przedstawiona jest na rysunku 53 pokazano na rysunku 54. W kodzie użyto różnych typów elementów składowych kursu, na początku znajduje się opis zagadnienia dodany w formie zwykłego tekstu, pod nim zdefiniowano sekcję kodu z opisem pętli "for". Po najechaniu myszką na poszczególne składowe pętli powyżej wyświetlają się odpowiednie wyjaśnienia w kolorze niebieskim. Ostatnią sekcją wstawioną do kodu jest obrazek wyświetlany na podstawie odpowiedniego id ustawionego wcześniej przy dodawaniu obrazka do kursu. Wstawiony obrazek pokazuje wynik działania omawianego fragmentu kodu.



Rysunek 54 - Wygląd podglądu przykładowego kursu (źródło: opracowanie własne)

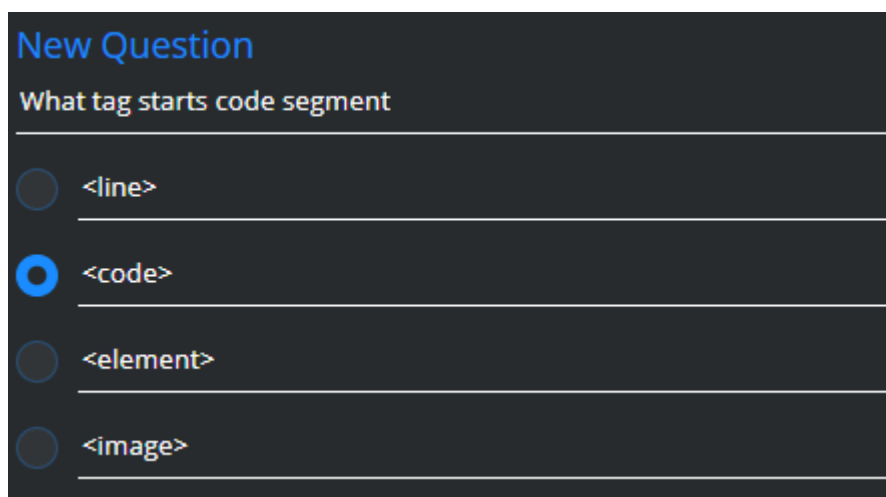
## Tworzenie quizu

Kolejnym etapem tworzenia kursu może być stworzenie do niego quizu. Kreator quizu pokazano na rysunku 55.



Rysunek 55 - Wygląd kreatora quizu (źródło: opracowanie własne)

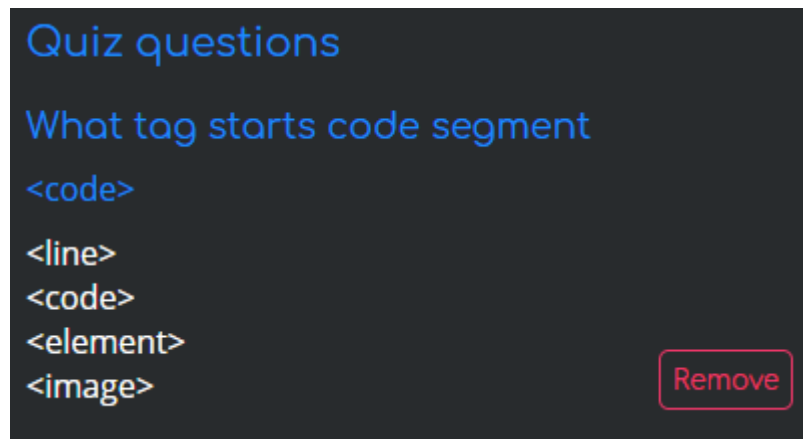
Zadania są tworzone poprzez podanie pytania oraz czterech możliwych odpowiedzi. Poprawna odpowiedź jest zaznaczana przez wybranie właściwej linijki, zobacz rysunek 56.



Rysunek 56 - Tworzenie nowego pytania quizu (źródło: opracowanie własne)



Poniżej wyświetlane są wszystkie dodane pytania wraz możliwościami odpowiedzi. Możliwe jest także ich usuwanie, zob. rys. 57.



*Rysunek 57 - Wygląd listy pytań quizu (źródło: opracowanie własne)*

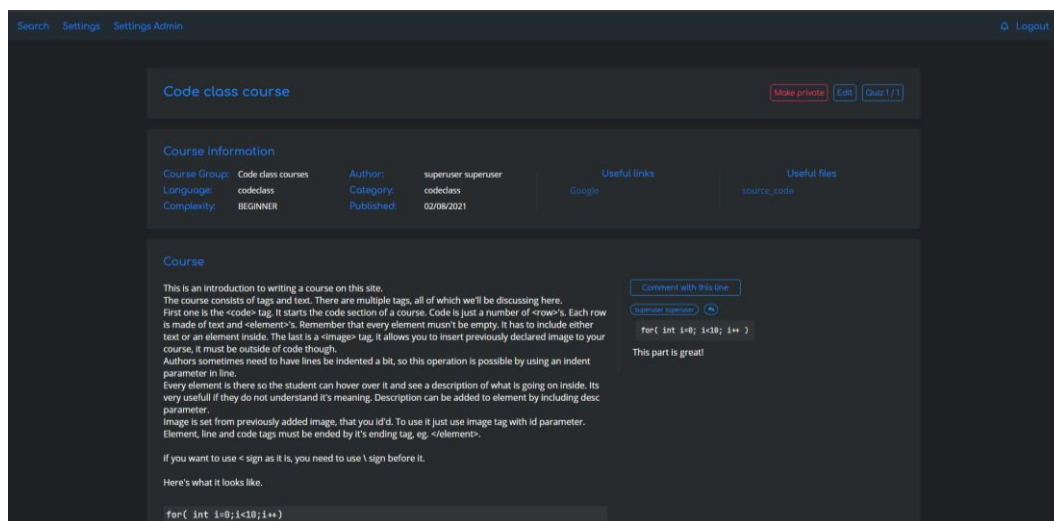
Strona zawiera możliwość powrotu do edycji kursu, zapisu, a także usunięcia quizu (zobacz rysunek 58). Możliwość usunięcia odblokowana jest, jeżeli quiz już istnieje.



*Rysunek 58 - Panel zarządzania zapisem quizu (źródło: opracowanie własne)*

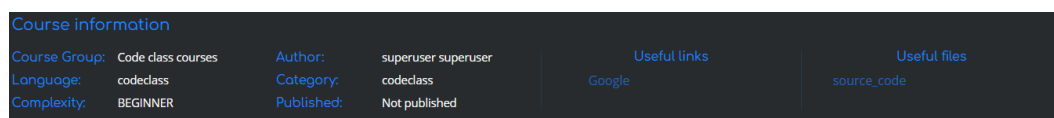
### **Strona główna kursu**

Po ukończeniu edycji kursu można do niego wejść. Strona kursu (zobacz rysunek 59) składa się z wielu elementów opisanych poniżej.



Rysunek 59 - Wygląd strony głównej kursu (źródło: opracowanie własne)

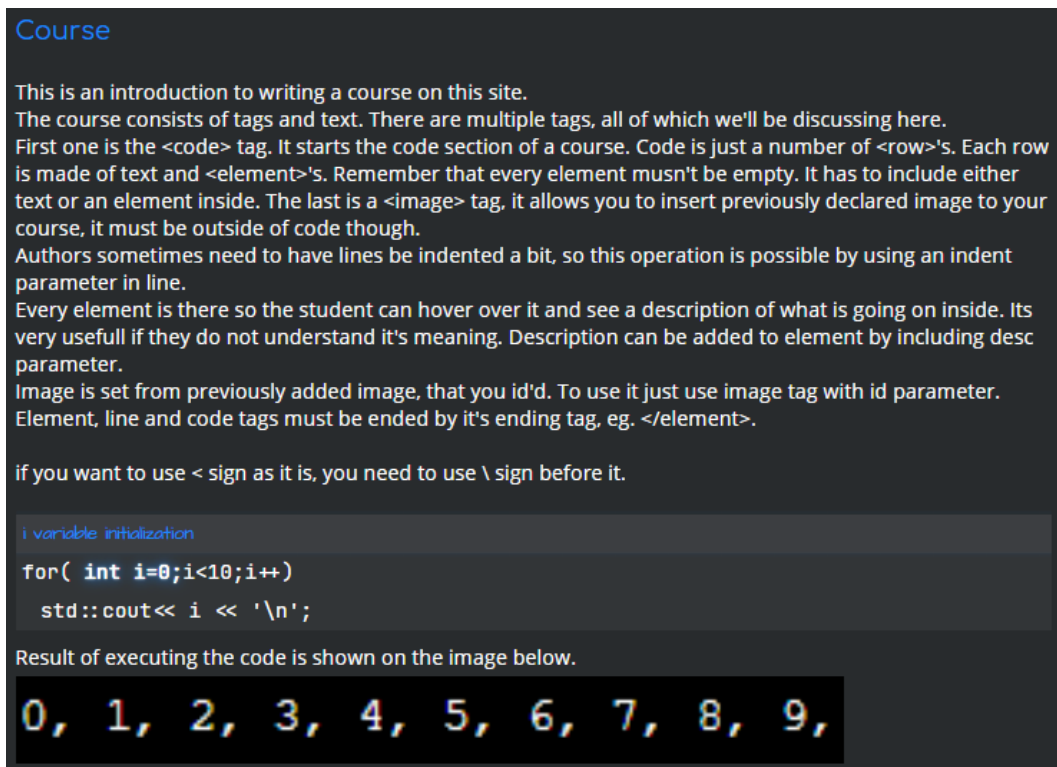
Główna strona kursu zawiera wprowadzone przez autora podstawowe dane oraz przesłane pliki i linki (zobacz rysunek 60).



Rysunek 60 - Wygląd podstawowych danych kursu (źródło: opracowanie własne)

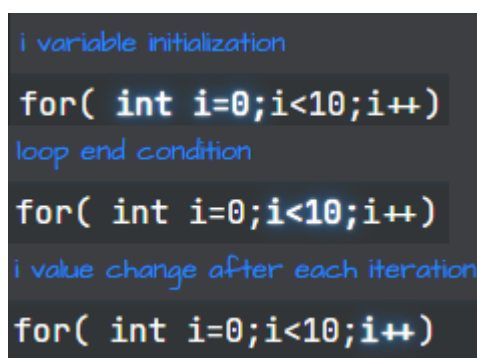
## Sekcja treści kursu

Pod sekcją informacji znajduje się sekcja właściwa kursu. Zawiera ona dane wprowadzane w panelu edycji i wygląda tak samo jak przedstawiony autorowi podgląd, zob. rys. 61. Obrazek dodany do kursu przez autora można nacisnąć. Pozwoli to na zobaczenie go w postaci oryginalnej (kurs posiada wersję zminiaturyzowaną). Aplikacja umożliwia przesuwanie oraz przybliżanie obrazka.



Rysunek 61 - Wygląd przykładowej treści kursu (źródło: opracowanie własne)

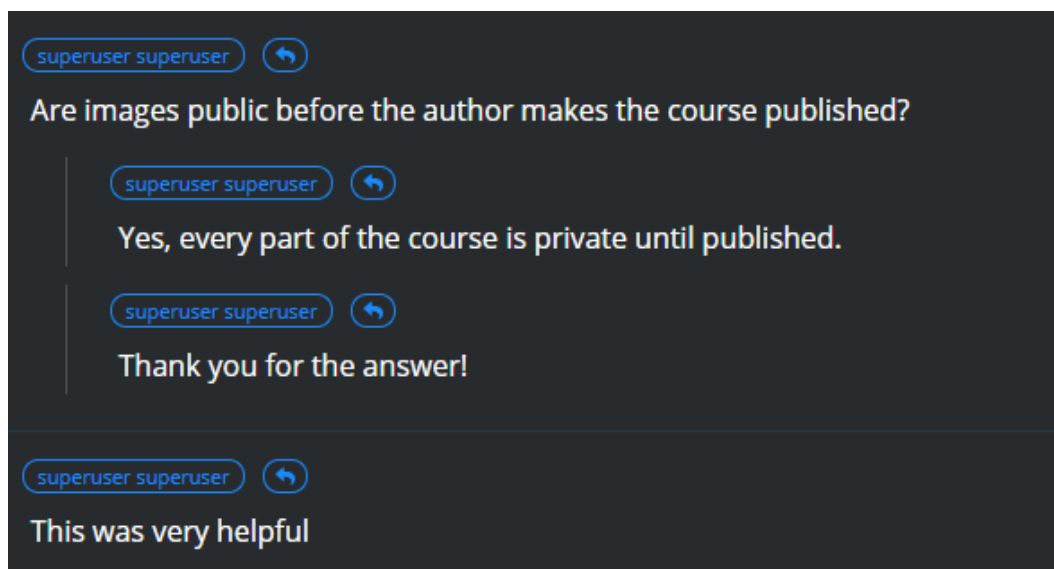
Kod napisany przez autora pozwala na zaznaczenie konkretnej części linii i wyświetlenie jej wytłumaczenia powyżej, w momencie, gdy użytkownik najedzie na niego kursorem myszy. Zaznaczenie to może być dowolnie zagnieżdżone. Sposób wyświetlania opisu jest przedstawiony na rysunku 62 poprzez pokazanie trzech różnych wytłumaczeń kodu, por. rys. 61, powyżej.



Rysunek 62 - Wygląd opisu (wytłumaczenia kodu) po najechaniu kursorem myszy (źródło: opracowanie własne)

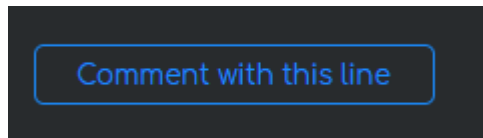
### Sekcja komentarzy użytkowników

Sekcją poniżej treści kursu są komentarze użytkowników. Zalogowany użytkownik może dodać komentarz do kursu oraz odpowiedzieć na komentarz już istniejący. Sposób wyświetlania komentarzy przedstawiony jest na rysunku 63.

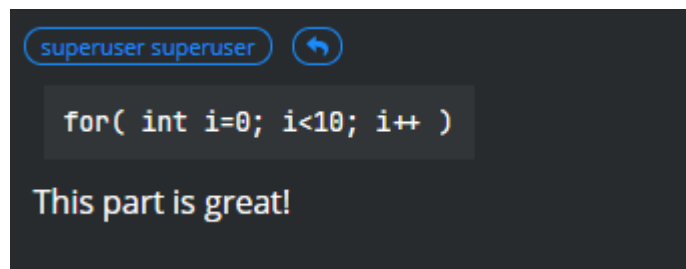


Rysunek 63 - Wygląd komentarzy (źródło: opracowanie własne)

Jeżeli użytkownik naciśnie na interaktywnej linii kodu w części kursu, obok zostanie wyświetlona opcja dodania komentarza z załączeniem kodu w postaci przycisku pokazanego na rysunku 64. Pod przyciskiem zostaną także wyświetlone komentarze dotyczące danej linii, przedstawione na rysunku 65. Dzięki temu użytkownik może skomentować dowolną linię w kodzie i na przykład zapytać autora o jej znaczenie lub poprosić go o dodatkowe wyjaśnienia. Autor może dodać takie wyjaśnienie odpowiadając komentarzem na komentarz lub poprawić wyjaśnienie w treści kursu. Korzyść jest obopólna, użytkownik zyskuje odpowiedź, a autor podpowiedź np. jak ulepszyć kurs.

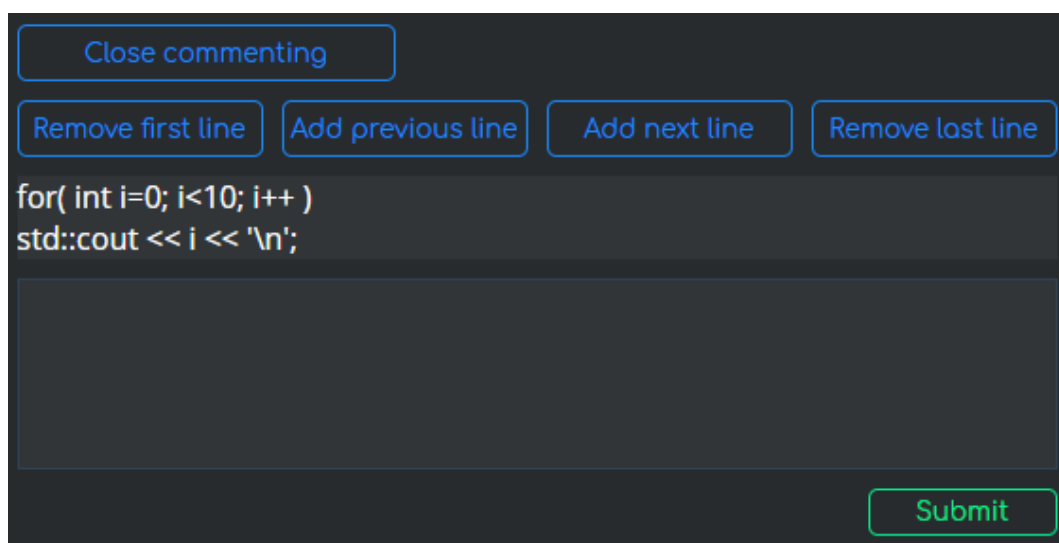


*Rysunek 64 - Przycisk wyświetlający formularz dodawania komentarza do danej linii kodu (źródło: opracowanie własne)*



*Rysunek 65 – Widok przykładowego komentarza dla pojedynczej linii kodu (źródło: opracowanie własne)*

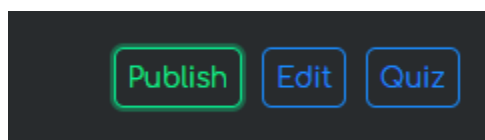
Komentarz może dotyczyć jednej lub wielu linii kodu programu. Załączanie kodu jest bardzo proste, użytkownik może naciskać odpowiednie przyciski zmieniając linie kodu, które chciał załączyć jako część komentarza, wpisać jego treść, a następnie nacisnąć przycisk *Submit*. Przesłanie komentarza automatycznie dodaje go na stronie bez konieczności odświeżenia strony. Na rysunku 66 przedstawiono dodawanie komentarza dotyczącego dwóch linii kodu.



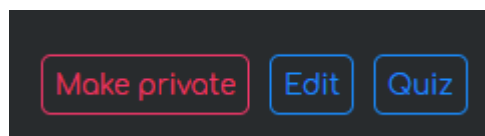
*Rysunek 66 - Edycja komentarza zawierającego kod (źródło: opracowanie własne)*

### Widok przycisków sterujących prywatnością kursu

Jeśli zalogowany użytkownik jest autorem kursu, może oznaczyć go jako publiczny lub prywatny. Służą do tego przyciski publish oraz make private, zobacz odpowiednio rysunek 67 i 68.



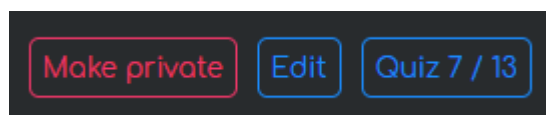
*Rysunek 67 - Wygląd przycisku zmiany stanu kursu na publiczny (źródło: opracowanie własne)*



*Rysunek 68 - Wygląd przycisku zmiany stanu kursu na prywatny (źródło: opracowanie własne)*

### Widok quizu

Wśród przycisków znajduje się również przycisk (Quiz) umożliwiający wzięcie udziału w quizie, utworzonym przez autora. Użytkownik, który brał udział w quizie oraz zdobył przynajmniej jeden punkt może zobaczyć swój wynik na przycisku otwierającym quiz (zobacz rysunek 69).



*Rysunek 69 - Wygląd przycisku quizu po rozwiązaniu (źródło: opracowanie własne)*

Strona quizu wyświetla użytkownikowi kolejne pytania, informując go jednocześnie o numerze oraz liczbie pytań (zobacz rysunek 70). Odpowiedzi są zapisywane po naciśnięciu przycisku *next*. Po udzieleniu odpowiedzi na wszystkie pytania, wynik jest zapisywany na serwerze oraz przedstawiony użytkownikowi, zob. rys. 71.



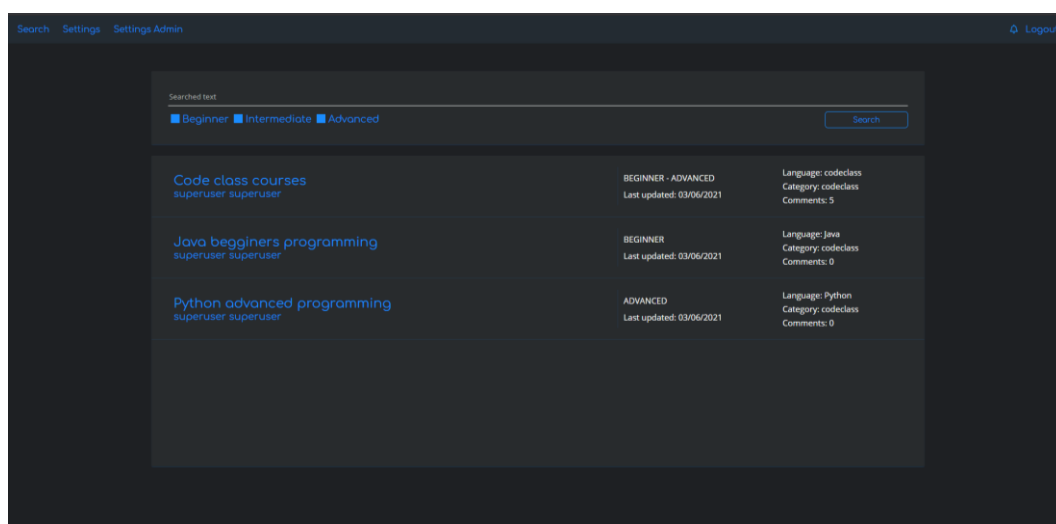
Rysunek 70 - Przykład pytania w quizie (źródło: opracowanie własne)



Rysunek 71 - Strona wynikowa po ukończeniu quizu (źródło: opracowanie własne)

## Wyszukiwanie kursów

Strona wyszukiwania kursów pozwala użytkownikowi wyszukać odpowiednie grupy kursów (zob. rys. 72).



Rysunek 72 - Wygląd strony wyszukiwarki kursów (źródło: opracowanie własne)

Użytkownik ma możliwość zaznaczyć jakie stopnie zaawansowania go interesują oraz wypełnić pole tekstowe wyszukiwarki (zobacz rysunek 73). Tekst wprowadzony przez użytkownika może zawierać szereg kryteriów. Możliwe jest wyszukiwanie języka, kategorii, autora, itp.

*Rysunek 73 - Wyszukiwarka kursów (źródło: opracowanie własne)*

Wyszukane elementy są podane w liście. Każdy rekord zawiera informacje o autorze, nazwie, zaawansowaniu, dacie aktualizacji czy liczbie komentarzy, zob. rys. 74.

|  |                                      |   |
|--|--------------------------------------|---|
| <b>Code class courses</b><br>superuser superuser | BEGINNER<br>Last updated: 02/07/2021 | Language: codeclass<br>Category: codeclass<br>Comments: 5 |
|--|--------------------------------------|---|

*Rysunek 74 - Przykładowy rekord zwrócony przez wyszukiwarkę (źródło: opracowanie własne)*

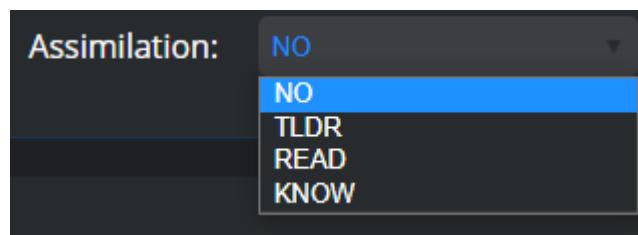
### **Funkcjonalności opcjonalne**

W ramach kursu użytkownik ma możliwość wyboru poziomu przyswojenia jego treści. Może to być dla niego istotną informacją, gdy zechce w przyszłości wrócić do kursu i pogłębić swoją wiedzę w tym zakresie. Funkcjonalność ta może również być przydatna użytkownikowi do odfiltrowywania artykułów, z którymi się zapoznał. Wybór poziomu przyswojenia artykułu dokonywany poprzez listę rozwijaną. Przewidziano w niej następujące opcje (rys. 75):

- NO - nie przeglądano artykułu,
- TLDR - artykuł przejrzany pobieżnie ze względu na obszerność,
- READ - artykuł przeczytany,
- KNOW - treść artykułu jest zrozumiała.

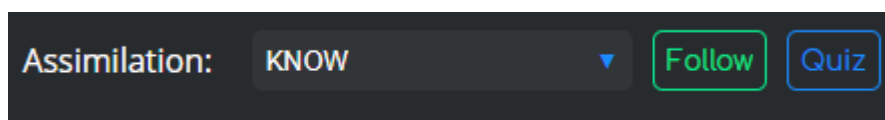
Wybrana wartość jest zapisana i wyświetlana przy kolejnych odwiedzinach na kursie.



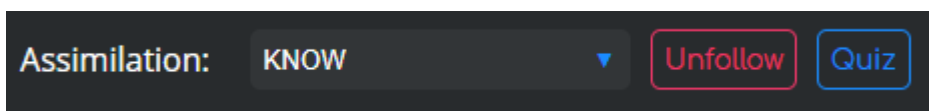


*Rysunek 75 - Określenie stanu przyswojenia wiedzy dla artykułu (źródło: opracowanie własne)*

Dostępna jest także możliwość obserwowania kursu oraz grupy kursów. Pozwala to na otrzymywanie powiadomień, jeżeli autor doda nowy kurs do śledzonej grupy kursów. Podana jest także możliwość rezygnacji z obserwowania. Zobacz kolejno rysunek 76 i rysunek 77.



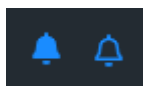
*Rysunek 76 - Śledzenie grupy kursu (źródło: opracowanie własne)*



*Rysunek 77 - Zakończenie śledzenia grupy kursu (źródło: opracowanie własne)*

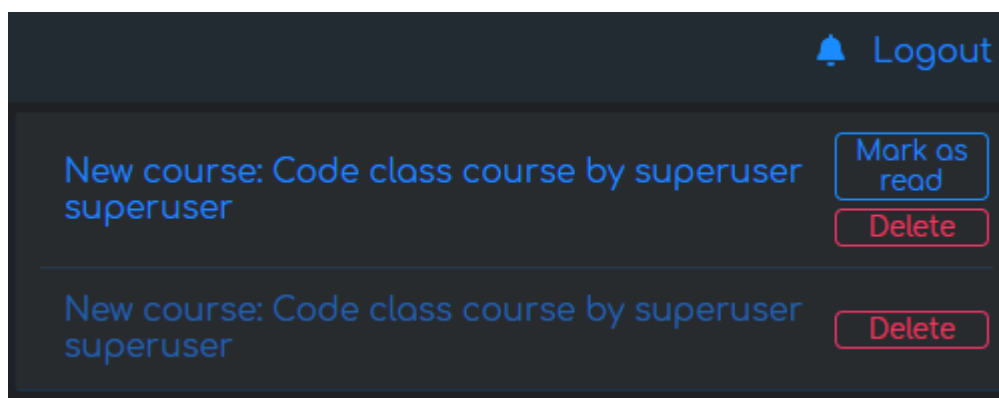
### **Widok powiadomień**

Jeśli użytkownik posiada nieprzeczytane powiadomienie, ikona powiadomień zostaje wypełniona (zobacz rysunek 78).



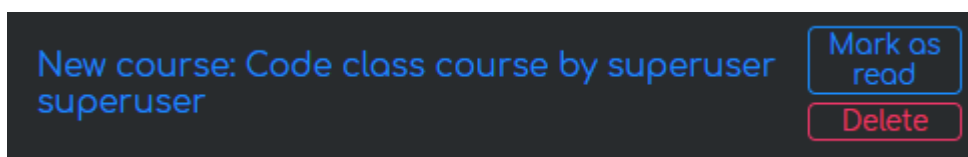
*Rysunek 78 – Widok ikony powiadomień, z lewej strony dzwonek informujący o nowych powiadomieniach, z prawej strony o braku powiadomień (źródło: opracowanie własne)*

Powiadomienia wyświetlane są w formie listy (zob. rys. 79).

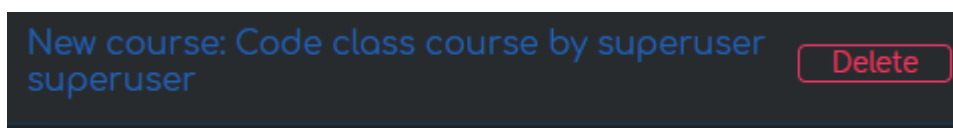


*Rysunek 79 - Wygląd listy powiadomień (źródło: opracowanie własne)*

Każdy jej element można przycisnąć, co spowoduje przekierowanie do obserwowanego kursu. Wygląd podstawowy powiadomienia przedstawiony jest na rysunku 80. Odwiedzone lub zaznaczone jako przeczytane powiadomienie zmienia kolor, aby było łatwe do odróżnienia od powiadomień aktywnych (zobacz rysunek 81). Wszystkie powiadomienia mogą zostać usunięte.



*Rysunek 80 - Wygląd powiadomienia (źródło: opracowanie własne)*



*Rysunek 81 - Wygląd powiadomienia po przeczytaniu (źródło: opracowanie własne)*

## 5. Zakończenie

Praca miała na celu stworzenie aplikacji, która umożliwi tworzenie kursów programistycznych będących przyjaznymi dla użytkownika, a proces ich tworzenia jest łatwy do zrozumienia i nie sprawia większych problemów. Cel ten został w pełni osiągnięty. Strona internetowa posiada przyjemną szatę graficzną oraz przyjazny interfejs. Kursy mogą być z łatwością tworzone w oparciu o odpowiednie narzędzie do edycji ze specjalnie w tym celu stworzonym językiem znaczników. Każdy kurs może zostać zaprogramowany tak, aby jego treść była jak najlepiej dopasowana do użytkowników. W przyszłości aplikacja może zostać wzbogacona o kolorowanie składni w treści kursu oraz w edytorze. Treść kursu mogłaby także otrzymać przyciski z linkami czy plikami, które obecnie są dostępne tylko w nagłówku kursu. Aplikacja mogłaby także zostać wzbogacona o interfejs kompilatora i środowiska uruchomieniowego, aby użytkownik mógł przetestować działanie podanego przez autora kodu.

## 6. Literatura

1. Implementing High Performance Parsers in Java -  
<https://www.infoq.com/articles/High-Performance-Parsers-in-Java-V2/>
2. Craig Walls, *Spring w akcji*, wydanie V, Helion, Gliwice 2019
3. Erik Hanchett, Benjamin Listwon, *Vue.js w akcji*, Helion, Gliwice 2020
4. <https://docs.spring.io/>
5. <https://hibernate.org/orm/documentation/5.4/>
6. <https://www.postgresql.org/docs/>
7. <https://vuejs.org/v2/guide/>
8. <https://docs.oracle.com/en/java/javase/13/>

## Spis tabel i rysunków

|   |    |
|---|----|
| Tabela 1. Typy elementów, z których składa się kod kursu .....  | 19 |
| Rysunek 1 – Ogólna architektura aplikacji .....   | 6  |
| Rysunek 2 – Przykład diagramu klas DAO .....  | 7  |
| Rysunek 3 – Diagram aktywności tokenizer .....  | 8  |
| Rysunek 4 – Diagram sekwencji zapisu treści kursu .....   | 9  |
| Rysunek 5 – Diagram sekwencji tworzenia elementów zagnieżdżonych .....                                      | 10 |
| Rysunek 6 - Diagram ERD.....  | 11 |
| Rysunek 7 – Struktura plików serwerowej części aplikacji.....   | 14 |
| Rysunek 8 - Struktura plików frontendowej części aplikacji.....   | 15 |
| Rysunek 9 – Klasa index .....   | 17 |
| Rysunek 10 – Wykrywanie znacznika code .....  | 18 |
| Rysunek 11 – Fragment kodu parsera sprawdzający istnienie parametru indent .....                            | 20 |
| Rysunek 12 – Przykład kodu kursu .....  | 21 |
| Rysunek 13 – Kod źródłowy elementów (tagów), z których składa się kod programu wyjaśnianego w kursie .....  | 22 |
| Rysunek 14 – Podział danych na listy względem zagnieżdżenia .....   | 23 |
| Rysunek 15 – Sprawdzenie głębokości następnego elementu .....   | 23 |
| Rysunek 16 – Aktualizacja składników systemu.....   | 24 |
| Rysunek 17 – Instalacja serwera bazy danych .....   | 24 |
| Rysunek 18 – Sprawdzenie działania usługi serwera baz danych.....   | 24 |
| Rysunek 19 – Utworzenie użytkownika i bazy danych .....   | 25 |
| Rysunek 20 – Instalacja Java JDK 13 .....   | 25 |
| Rysunek 21 – Nadanie uprawnień wykonywania dla pliku Maven oraz otwarcie pliku ustawień serwera danych..... | 25 |
| Rysunek 22 - Zmiana ustawień serwera danych .....   | 26 |
| Rysunek 23 - Kompilacja i uruchomienie serwera danych .....   | 26 |
| Rysunek 24 - Instalacja serwera www .....   | 26 |
| Rysunek 25 - Sprawdzenie działania serwera www.....   | 27 |
| Rysunek 26 - Instalacja node.js oraz yarn .....   | 27 |
| Rysunek 27 - Naprawa problemu dostępu podczas instalacji pakietów .....                                     | 27 |

|  |           |
|--|-----------|
| <i>Rysunek 28 - Instalacja zależności projektu oraz kompilacja aplikacji strony internetowej .....</i> | <i>28</i> |
| <i>Rysunek 29 - Konfiguracja udostępniania strony www .....</i>  | <i>28</i> |
| <i>Rysunek 30 - Zmiana uprawnień folderu strony .....</i>  | <i>28</i> |
| <i>Rysunek 31 – Formularz logowania .....</i>  | <i>29</i> |
| <i>Rysunek 32 - Dane zwrotne logowania .....</i>   | <i>30</i> |
| <i>Rysunek 33 - Wygląd strony ustawień administratora .....</i>  | <i>30</i> |
| <i>Rysunek 34 - Zarządzanie językami programowania i kategoriami .....</i>                             | <i>31</i> |
| <i>Rysunek 35 - Zarządzanie rolami .....</i>   | <i>31</i> |
| <i>Rysunek 36 - Edycja roli .....</i>  | <i>32</i> |
| <i>Rysunek 37 - Wygląd strony ustawień użytkownika .....</i>   | <i>32</i> |
| <i>Rysunek 38 - Zmiana danych użytkownika .....</i>  | <i>33</i> |
| <i>Rysunek 39 - Lista własnych kursów .....</i>  | <i>33</i> |
| <i>Rysunek 40 - Tworzenie grupy kursów .....</i>   | <i>33</i> |
| <i>Rysunek 41 - Widok grupy kursów .....</i>   | <i>34</i> |
| <i>Rysunek 42 - Tworzenie kursu .....</i>  | <i>34</i> |
| <i>Rysunek 43 - Wygląd strony edycji kursu .....</i>   | <i>35</i> |
| <i>Rysunek 44 - Edycja podstawowych danych kursu .....</i>   | <i>35</i> |
| <i>Rysunek 45 - Edycja linków i plików kursu .....</i>   | <i>36</i> |
| <i>Rysunek 46 - Wygląd strony edycji treści kursu .....</i>  | <i>36</i> |
| <i>Rysunek 47 - Dodawanie obrazków do kursu .....</i>  | <i>37</i> |
| <i>Rysunek 48 - Panel zarządzania zapisem treści kursu .....</i>                                       | <i>37</i> |
| <i>Rysunek 49 - Zaznaczenie błędnej linii kodu .....</i>   | <i>37</i> |
| <i>Rysunek 50 - Wyświetlenie treści błędu w kodzie .....</i>   | <i>38</i> |
| <i>Rysunek 51 - Panel dodawania znaczników do kodu .....</i>   | <i>38</i> |
| <i>Rysunek 52 - Wstawianie znaczników podczas zaznaczenia .....</i>                                    | <i>38</i> |
| <i>Rysunek 53 - Przykładowy kod kursu .....</i>  | <i>38</i> |
| <i>Rysunek 54 - Wygląd podglądu przykładowego kursu .....</i>  | <i>39</i> |
| <i>Rysunek 55 - Wygląd kreatora quizu .....</i>  | <i>40</i> |
| <i>Rysunek 56 - Tworzenie nowego pytania quizu .....</i>   | <i>40</i> |
| <i>Rysunek 57 - Wygląd listy pytań quizu .....</i>   | <i>41</i> |
| <i>Rysunek 58 - Panel zarządzania zapisem quizu .....</i>  | <i>41</i> |

|   |    |
|---|----|
| Rysunek 59 - Wygląd strony głównej kursu .....  | 42 |
| Rysunek 60 - Wygląd podstawowych danych kursu .....   | 42 |
| Rysunek 61 - Wygląd przykładowej treści kursu .....   | 43 |
| Rysunek 62 - Wygląd opisu (wy tłumaczenia kodu) po najechaniu kursorem myszy .....  | 43 |
| Rysunek 63 - Wygląd komentarzy .....  | 44 |
| Rysunek 64 - Przycisk wyświetlający formularz dodawania komentarza do danej linii kodu .....  | 45 |
| Rysunek 65 – Widok przykładowego komentarza dla pojedynczej linii kodu.....   | 45 |
| Rysunek 66 - Edycja komentarza zawierającego kod .....  | 45 |
| Rysunek 67 - Wygląd przycisku zmiany stanu kursu na publiczny .....   | 46 |
| Rysunek 68 - Wygląd przycisku zmiany stanu kursu na prywatny .....  | 46 |
| Rysunek 69 - Wygląd przycisku quizu po rozwiązaniu .....  | 46 |
| Rysunek 70 - Przykład pytania w quizie .....  | 47 |
| Rysunek 71 - Strona wynikowa po ukończeniu quizu .....  | 47 |
| Rysunek 72 - Wygląd strony wyszukiwarki kursów .....  | 47 |
| Rysunek 73 - Wyszukiwarka kursów.....   | 48 |
| Rysunek 74 - Przykładowy rekord zwrócony przez wyszukiwarkę.....  | 48 |
| Rysunek 75 - Określenie stanu przyswojenia wiedzy dla artykułu .....  | 49 |
| Rysunek 76 - Śledzenie grupy kursu .....  | 49 |
| Rysunek 77 - Zakończenie śledzenia grupy kursu .....  | 49 |
| Rysunek 78 – Widok ikony powiadomień, z lewej strony dzwonek informujący o nowych powiadomieniach, z prawej strony o braku powiadomień..... | 49 |
| Rysunek 79 - Wygląd listy powiadomień .....   | 50 |
| Rysunek 80 - Wygląd powiadomienia .....   | 50 |
| Rysunek 81 - Wygląd powiadomienia po przeczytaniu .....   | 50 |

## Streszczenie

Tytuł pracy w języku polskim: **Aplikacja internetowa ułatwiająca naukę programowania**

Tytuł pracy w języku angielskim: **Web application facilitating learning of programming**

Praca omawia oraz przedstawia aplikację internetową wspomagającą nauczanie programowania. Udostępnia ona interfejs umożliwiający tworzenie kursów z dołączeniem interaktywnego kodu w postaci komentarzy przy odpowiednio zaprogramowanych częściach. Aplikacja implementuje tworzenie sekcji programistycznej w stylu podobnym do xml oraz odpowiedni dla niego analizator składni. Udostępniony jest szereg funkcjonalności pozwalający autorowi kursu na odpowiednie zaprezentowanie treści zawartych w kursie. Możliwe jest dodawanie linków, plików czy obrazków.