

**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ**

Mykyta Vovk

Serwis internetowy
do zarządzania szkołą

Praca dyplomowa inżynierska

Opiekun pracy:
dr inż. Antoni Szczepański

Rzeszów, 2020

Spis treści

SPIS TREŚCI.....	3
1. WPROWADZENIE.....	5
2. NARZĘDZIA INFORMATYCZNE UŻYTE DO REALIZACJI SERWISU	9
2.1. Język programowania Java	9
2.1.1. Java 1.8 jako język obiektowy.....	9
2.1.2. Java - zalety i wady	10
2.2. Spring Framework	12
2.2.1. Spring MVC	13
2.2.2. Technologia Spring Security	15
2.2.3. Technologia Spring Boot.....	15
2.3. Relacyjna baza danych.....	16
2.4. System zarządzania bazą danych MySQL	19
2.5. Hibernate framework	20
2.6. Lokalny serwer TomCat	21
2.7. Technologia JSP	22
2.8. Technologia Maven Builder	23
2.9. Cascading Style Sheets (CSS).....	24
2.10. Środowisko programistyczne JetBrains IntelliJ IDEA	25
2.11. Wizualny edytor baz danych MySQL Workbench.....	25
2.12. System kontroli wersji Git i GitHub.....	26
3. ETAPY PRACY, PROWADZĄCE DO POWSTANIA SERWISU.....	28
3.1. Utworzenie nowego projektu w IntelliJ Idea.....	28
3.2. Zaprojektowanie i utworzenie relacyjnej bazy danych	30
3.3. Organizacja i podział funkcjonalności	37
3.4. Realizacja serwisu	40
3.4.1. Uruchomienie projektu na serwerze lokalnym TomCat	40
3.4.2. Dodawanie zależności Maven	41
3.4.3. Użycie technologii Git i GitHub	42

3.4.4.	Ustawienie konfiguracji projektu.....	45
3.4.5.	Bezpieczeństwo serwisu	47
3.4.6.	Komunikacja z bazą danych	48
3.4.7.	Tworzenie logiki biznesowej serwisu.....	52
3.4.8.	Tworzenia kontrolerów	53
3.4.9.	Tworzenie widoku serwisu	56
4.	DOKUMENTACJA MOŻLIWOŚCI ZIMPLEMENTOWANEGO SERWISU	59
4.1.	Widok serwisu od strony administratora	59
4.2.	Widok serwisu od strony nauczyciela.....	63
4.3.	Widok serwisu od strony ucznia.....	68
5.	PODSUMOWANIE I WNIOSKI.....	71
	LITERATURA	73

1. Wprowadzenie

Obecne tempo rozwoju społeczeństwa jest niezwykle wysokie. Aby nie doświadczyć tzw. wykluczenia cyfrowego, człowiek musi nadążyć za zmianami, przetwarzać ogromne ilości informacji, pochodzących ze wszystkich zakątków globu. Po pojawienniu się pierwszych komputerów, różne firmy zaczęły korzystać z technologii komputerowej do automatyzacji procesów, które wcześniej były wykonywane ręcznie przez ludzi. Komputery i nowe technologie pomogły przyspieszyć pracę, automatyzując wiele procesów i automatyzując przetwarzanie dużych zbiorów informacji. Z tego więc powodu technologia informacyjna jest obecnie priorytetem w wielu obszarach działalności i determinuje przyszły rozwój społeczeństwa.

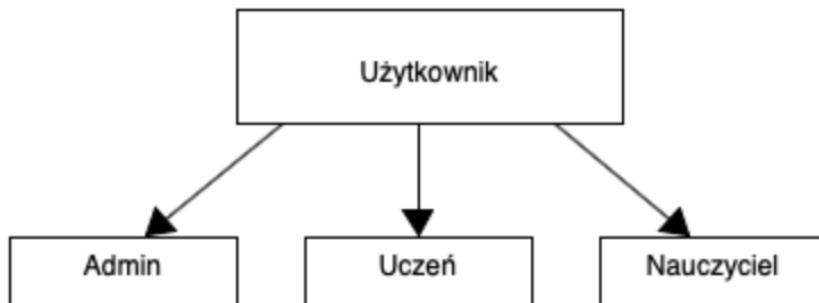
Żyjemy w skomputeryzowanym świecie, w którym każda szkoła, instytut, uniwersytet i inna instytucja edukacyjna musi zapewnić komputeryzację procesu edukacyjnego. Ułatwia to i przyspiesza zarządzanie procesem uczenia.

Automatyzacja kontroli procesu uczenia się za pomocą sprzętu i oprogramowania komputerowego jest opłacalnym i wydajnym narzędziem zarządzania, dlatego utworzenie serwisu online do zarządzania szkołą jest dobrym rozwiązaniem tego problemu. Taka usługa może pomóc przejść od papierowej do cyfrowej formy procesu uczenia się, co pomaga administracji szkoły, nauczycielom i uczniom. Oznacza to, że uczniowie mogą przeglądać swoje oceny, zadania domowe, frekwencję bezpośrednio za pomocą takiej usługi, a nauczyciele mogą wpisywać oceny, sprawdzać frekwencję uczniów i dodawać zadania domowe. Administracja szkoły może odejść od papierowego archiwum i prowadzić elektroniczną dokumentację swoich uczniów i nauczycieli.

Obecnie istnieje wiele usług zarządzania procesem uczenia się, takich jak: Alma, Schoolbic, Classter, OpenEduCat, Gradelink. Wszystkie te usługi mają swoje wady i zalety, ale główna idea i cel są takie same. Większość z nich to aplikacje komputerowe, co jest wadą takiego rozwiązania, ponieważ dostęp do danych jest możliwy tylko za pośrednictwem urządzeń, na których zainstalowane jest określone oprogramowanie. Na przykład Schoolbic ma lepsze rozwiązanie tego problemu, ponieważ jest serwisem internetowym, do którego można uzyskać dostęp i zalogować się do niego z dowolnego urządzenia.

Celem tej pracy jest stworzenie usługi online, która pomogłaby w zarządzaniu tokiem szkolenia. W przypadku tworzenia tego serwisu użytkownikom należałooby

przypisać role, co oznacza, że uczniowie, nauczyciele i administracja mieliby swoje oddzielne role i funkcje. To rozwiązanie wymaga utworzenia różnych praw dostępu do danych dla każdego użytkownika. Na rysunku 1.1 przedstawiono przykładowy schemat podziału użytkowników.



Rys. 1.1. Schemat podziału użytkowników na grupy

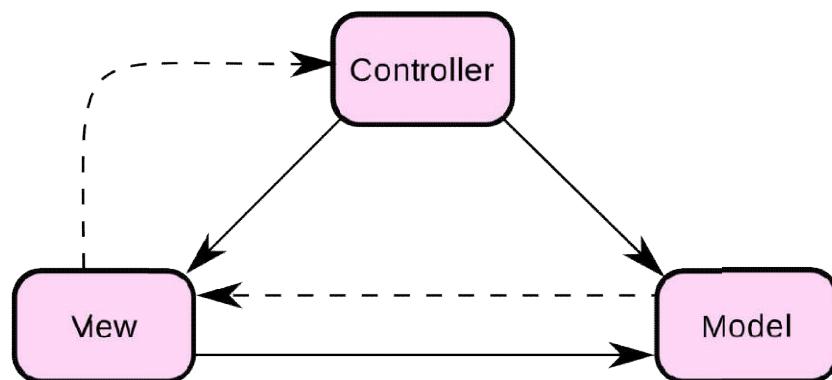
Implementacja takiego serwisu polega na stworzeniu strony internetowej z własnym interfejsem i funkcjonalnością do pracy z danymi. Taki serwis można utworzyć na podstawie programów architektury MVC (Model-View-Controller). Architektura MVC pozwala na podział kodu aplikacji na 3 części: Model (Model), Widok (View) i Kontroler (Controller). Po raz pierwszy ta architektura została opisana w 1978 roku i była przeznaczona dla aplikacji z interfejsem graficznym (okna i przyciski), ale później została przystosowana do aplikacji internetowych. Podział na części pozwala uprościć duży kod. Jeśli napisać kod w jednym i długim skrypcie, to on staje się trudny do zrozumienia i trudno jest wprowadzić zmiany bez popełnienia błędu. MVC nie jest powiązany z żadnym konkretnym językiem programowania i nie wymaga użycia programowania obiektowego ani żadnego innego paradygmatu.

Model zawiera całą logikę aplikacji, przechowuje i przetwarza dane, bez bezpośredniej interakcji z użytkownikiem (dostęp do modelu można uzyskać tylko z kodu, wywołując jego funkcje). Na przykład, przechowywanie informacji w bazie danych czy sprawdzanie poprawności danych wprowadzonych w formularzu jest zadaniem Modelu, ale odbieranie tych danych od użytkownika lub wyświetlanie informacji na ekranie lub przetwarzanie kliknięcia przycisku już nie jest.

Widok wyświetla dane, które zostały do niego przekazane. W aplikacji internetowej widok zazwyczaj składa się z szablonów stron HTML, natomiast w aplikacjach stacjonarnych lub mobilnych Prezentacja to kod odpowiedzialny za wyświetlanie informacji na ekranie, renderowanie przycisków i innych elementów interfejsu.

Kontroler jest odpowiedzialny za wykonywanie żądań otrzymywanych od użytkownika. W aplikacji internetowej kontroler zazwyczaj analizuje parametry żądania HTTP z metod POST i GET, wywołuje model, aby uzyskać lub zmienić niektóre dane, a na koniec wywołuje widok, aby wyświetlić wynik żądania. Liczba kontrolerów zależy od liczby sekcji lub stron witryny. W aplikacjach komputerowych kontroler jest odpowiedzialny za obsługę kliknięć przycisków i innych działań użytkownika.

Na poniższym rysunku przedstawiono przykładowy schemat interakcji między składnikami szablonu architektury MVC.



Rys. 1.2. Schemat interakcji między składnikami szablonu architektury MVC

Ten serwis internetowy i ta architektura została zrealizowana za pomocą języka programowania Java w wersji 1.8. Java jest bardzo prosta z punktu widzenia programisty, ponieważ ma ona dość zrozumiałą składnię. Jest bardzo wygodna, ma obszerną bibliotekę klas do zrealizowania różnych zadań i ma dobrą dokumentację, za pomocą której można łatwo jej nauczyć się.

Realizując serwis internetowy, na podstawie architektury MVC, najlepiej jest użyć biblioteki Spring MVC, za pomocą której można łatwo stworzyć taki program. Ta biblioteka jest najbardziej przyjazna do zrealizowania takiego zadania, ponieważ zawiera ona wszystkie elementy do stworzenia programu na podstawie architektury MVC. Także ma ona możliwość konfigurowania bezpośrednio w kodzie programu, za pomocą adnotacji i ma ona dużą i pełną dokumentację.

Dla realizacji modelu serwisu została użyta biblioteka technologii Hibernate, która jest biblioteką języka Java. Za pomocą niej można tworzyć encje bezpośrednio w kodzie programu. Hibernate pozwala na pracę z różnymi bazami danych, ponieważ zapytania do bazy danych tworzy się w języku HQL. W tym przypadku użyto bazę MySQL.

Dla realizacji widoku serwisu została wybrana technologia JSP (Java Server Pages), która pozwala programistom dynamicznie generować HTML i XML.

Żeby uruchomić taki program, potrzebny jest serwer. W tym serwisie został wybrany serwer TomCat w wersji 9.0.24, ponieważ jest on wieloplatformowy. Chociaż jego funkcje są podstawowe, zawiera wszystkie potrzebne dla uruchomienia tego serwisu.

Aplikacja webowa została stworzona za pomocą narzędzia programistycznego IntelliJ IDEA, ponieważ jest ona łatwe w wykorzystaniu, zawiera dużo szablonów dla różnych programów i ma ona możliwość dodawania różnych wtyczek, które upraszczają tryb pisania kodu.

Praca będzie składać się z trzech części. W pierwszej będą opisane wszystkie użyte technologie oraz ich zastosowania w tym projekcie. Druga część pracy polegać będzie na opisie charakterystyki tworzenia projektu i dokonaniu analizy funkcjonalności systemu. W trzeciej części będzie przedstawiony widok zrealizowanego serwisu, wraz z opisem jego funkcjonalności i ilustracją ich na zrzutach ekranu komputera.

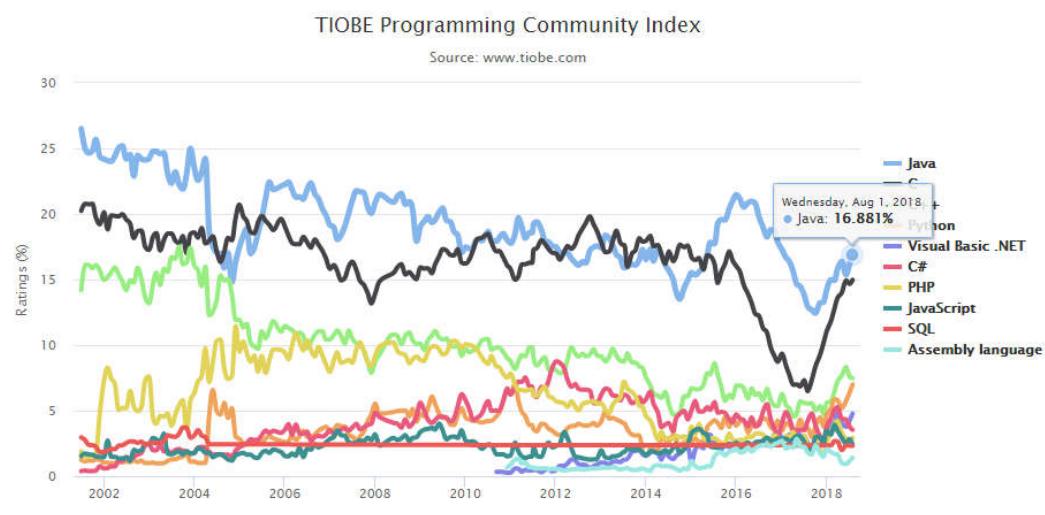
2. Narzędzia informatyczne użyte do realizacji serwisu

2.1. Język programowania Java

2.1.1. Java 1.8 jako język obiektowy

Java jest od wielu lat jednym z najważniejszych i najczęściej używanych języków programowania na świecie. W przeciwieństwie do innych języków programowania, wpływ Javy nie tylko nie zmniejszył się z czasem, ale wręcz wzrósł. Od pierwszego wydania, przeszedł na czoło programowania aplikacji internetowych. Każda kolejna wersja tylko umacniała tę pozycję. Dzisiaj Java jest nadal pierwszym i najlepszym językiem do tworzenia aplikacji internetowych. Mówiąc najprościej, większość współczesnego kodu jest napisana w języku Java. Świadczy to o szczególnym znaczeniu języka Java dla programowania [1].

Dzięki swojej długiej historii, Java zdobyła swoje miejsce w programowaniu. Indeks TIOBE, jeden z najbardziej autorytatywnych indeksów popularności programów na świecie, wykorzystuje wyniki wyszukiwania do celach rankingowych. Pomimo rosnącej popularności Go i Pythona, Java pozostaje na szczycie listy od ponad dekady (rys. 2.1).



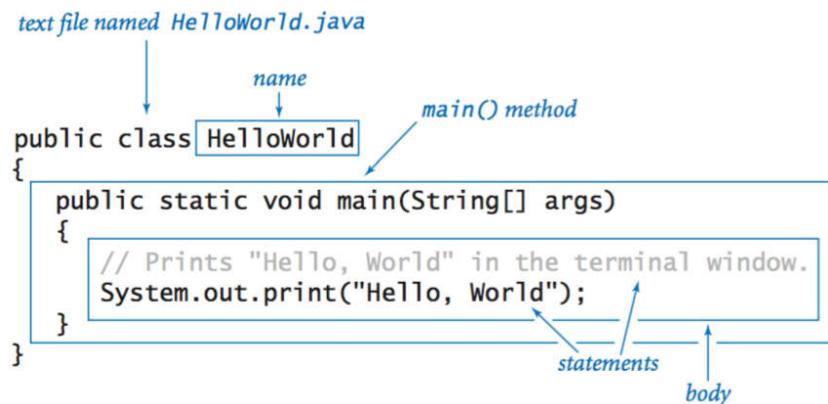
Rys. 2.1. Indeks popularności języka Java, na tle innych języków programowania [2]

Głównym powodem sukcesu Java jest jego elastyczność. Począwszy od pierwszej wersji 1.0, ten język stale dostosowuje się do zmian w środowisku programistycznym i podejść do pisania programów. A co najważniejsze, nie tylko podąża za trendami w programowaniu, ale także pomaga je tworzyć. Zdolność Javy do przystosowywania się do szybkich zmian w informatyce jest głównym powodem sukcesu tego języka [1].

2.1.2. Java - zalety i wady

Wybór języka programowania Java polegał na rozważeniu jego zalet i wad. Zalety tego języka programowania są następujące:

- **Programowanie obiektowe:** Java obejmuje programowanie obiektowe - koncepcję, w której nie tylko jest definiowany typ danych i jego struktura, ale także zestaw zastosowanych do niego funkcji. W ten sposób struktura danych staje się obiektem, który można kontrolować, w celu tworzenia relacji między różnymi obiektami [1].
- **Java to język wysokiego poziomu z prostą składnią:** Java jest językiem wysokiego poziomu, to znaczy że jest on podobny do języka ludzkiego, w przeciwieństwie do języków niskiego poziomu, które przypominają kod maszynowy. Języki wysokiego poziomu są konwertowane za pomocą kompilatorów lub tłumaczy. Upraszczają to rozwój, dzięki czemu język jest łatwiejszy do pisania, czytania i utrzymywania [1].

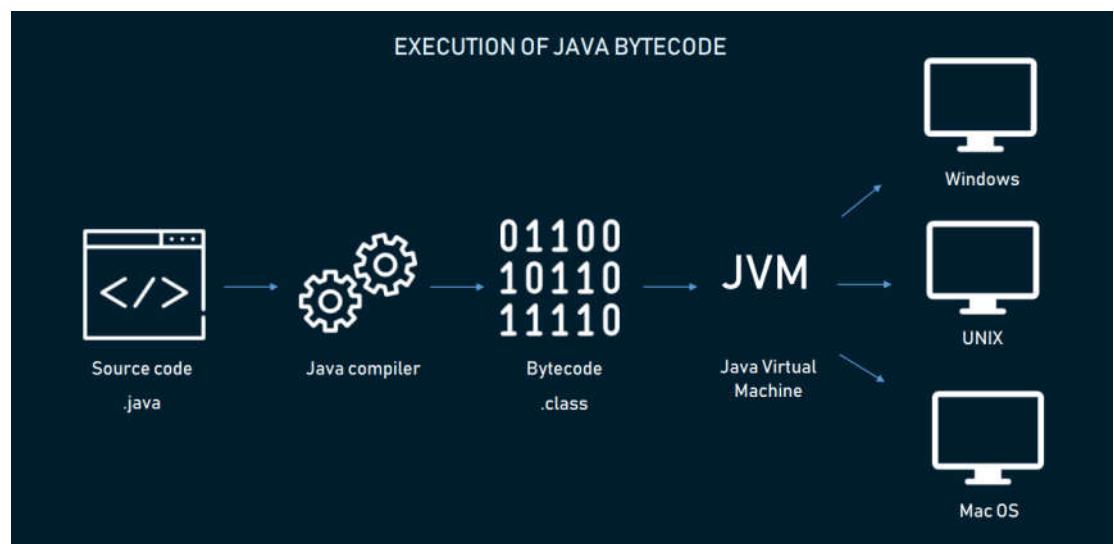


Rys. 2.2. Przykładowa składnia głównej metody Java [2]

- **Standard obliczeniowy dla przedsiębiorstw.** Aplikacje korporacyjne to główna zaleta Javy od lat 90., kiedy organizacje zaczęły szukać niezawodnych narzędzi programistycznych, innych niż C. Java obsługuje wiele bibliotek - elementów składowych każdego systemu korporacyjnego. Biblioteki pomagają programistom tworzyć wszelkie funkcje, które mogą być potrzebne firmie [1].
- **Bezpieczeństwo:** uważa się, że Java jest bezpiecznym językiem, ale nie jest to do końca prawda. Sam język nie chroni przed lukami, ale niektóre jego funkcje eliminują typowe luki. Po pierwsze, w przeciwieństwie do C, Java nie ma wskaźników. Wskaźnik to obiekt, który przechowuje adres komórki pamięci o innej wartości, co może powodować nieautoryzowany dostęp do pamięci. Po

drugie, w Javie jest Security Manager, polityka bezpieczeństwa tworzona dla każdej aplikacji, w której można określić reguły dostępu [1].

- **Niezależność od platformy:** można utworzyć aplikację Java w systemie Windows, skompilować ją do kodu bajtowego i uruchomić na dowolnej platformie obsługującej wirtualną maszynę Java (JVM). Zatem JVM służy jako warstwa abstrakcji między kodem a sprzętem. Wszystkie główne systemy operacyjne, w tym Windows, Mac OS i Linux, obsługują JVM [2].



Rys. 2.3. Przykładowy schemat działania programów stworzonych w języku Java [2]

- **Automatyczne zarządzanie pamięcią:** programiści Java nie muszą ręcznie pisać kodu do zarządzania pamięcią, dzięki automatycznemu zarządzaniu pamięcią (AMM). Skuteczność programu jest bezpośrednio związana z pamięcią. Ilość pamięci jest jednak ograniczona. Pisząc aplikację w językach z ręcznym zarządzaniem pamięcią, programiści ryzykują, że zapomną przydzielić pamięć, co doprowadzi do zwiększenia pamięci zajmowanej przez aplikację i problemy z wydajnością [2].
- **Wielowątkowość:** wątek jest najmniejszą jednostką przetwarzania danych w programowaniu. Aby zmaksymalizować wykorzystanie czasu procesora, Java umożliwia jednoczesne uruchamianie wątków, co nazywa się wielowątkowością [2].
- **Stabilność i społeczność:** przez wiele lat Java była promowana przez społeczność, wsparcie Oracle oraz mnóstwo aplikacji i języków w JVM. Ponadto ciągle pojawiają się nowe wersje Java, z interesującymi nowymi funkcjami [2].

Język programowania Java, mimo swoich niekwestionowanych zalet, nie jest pozbawiony wad. Niektóre z nich zostały wymienione poniżej:

- **Płatne wykorzystanie komercyjne:** firma Oracle niedawno ogłosiła, że od 2019 r. firma zacznie pobierać opłaty za korzystanie z oprogramowania Java Standard Edition 8 do „celów komercyjnych”. Trzeba będzie płacić za wszystkie nowe aktualizacje i poprawki błędów. Opłata będzie zależeć od liczby użytkowników lub komputerów [2].
- **Niska wydajność:** każdy język wysokiego poziomu ma raczej niską wydajność z powodu komplikacji i abstrakcji przy użyciu maszyny wirtualnej. Nie jest to jednak jedyny powód niskiej prędkości Javy. Na przykład aplikacja do czyszczenia pamięci jest przydatną funkcją, która niestety prowadzi do poważnych problemów z wydajnością, jeśli wymaga więcej niż 20 procent czasu procesora. Złe ustawienia buforowania mogą powodować nadmierne zużycie pamięci [2].
- **Pełny i złożony kod:** pełny kod może wydawać się zaletą, która pomaga nauczyć się języka. Jednak długie, zbyt złożone zdania utrudniają czytanie i przeglądanie kodu. Podobnie jak języki naturalne, wiele języków programowania wysokiego poziomu zawiera zbędną informację [2].

2.2. Spring Framework

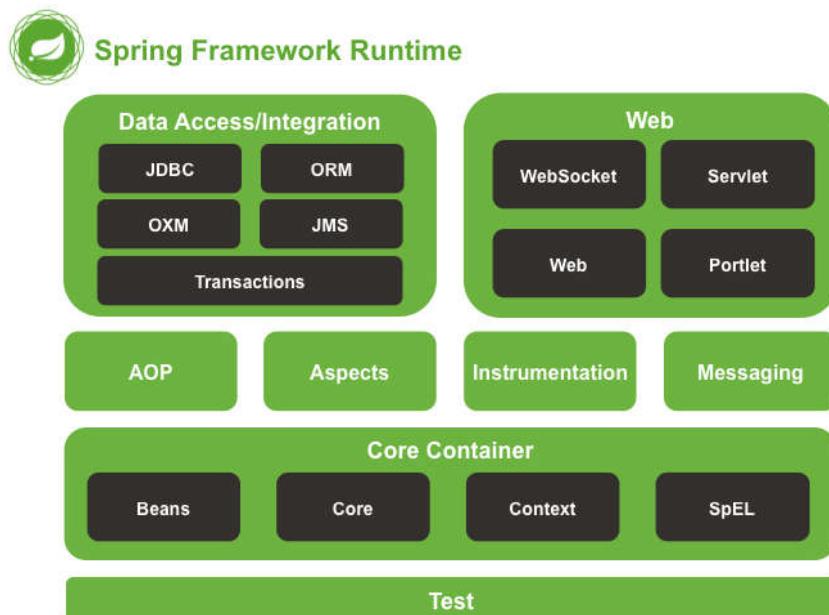
Spring jest architektonicznym frameworkiem, ponieważ nie był tak bardzo zaprojektowany do wykonywania pewnego rodzaju zadań aplikacyjnych, ale zapewnia lepszą skalowalność, możliwość łatwiejszego testowania i łatwiejszą integrację z innymi frameworkami (np. Struts lub Hibernate). Dzięki temu łatwiej jest pisać duże aplikacje. Programiści po prostu unikają wielu problemów związanych z tworzeniem aplikacji korporacyjnych, zamiast je rozwiązywać [3].

Spring jest dość dużym frameworkem, ponieważ jego twórcom udało się objąć prawie wszystkie aspekty programowania przemysłowych aplikacji Java. Głównymi aspektami frameworka Spring są:

- **IoC (Inversion of Control) kontener:** inwersja kontroli jest zasadą w inżynierii oprogramowania, dzięki której sterowanie obiektem lub częściami programu jest przenoszone do kontenera lub frameworka. Najczęściej kontener jest używany w kontekście programowania obiektowego [3].

- **AOP framework:** Aspect-Oriented Programming (AOP) uzupełnia Object-Oriented Programming (OOP), zapewniając inny sposób myślenia o strukturze programu. Kluczową jednostką modułowości w OOP jest klasa, podczas gdy w AOP jednostką modułowości jest aspekt. Aspekty umożliwiają modularyzację problemów, takich jak zarządzanie transakcjami, które obejmują wiele typów i obiektów [3].

Jak widać na poniższym rysunku, Spring ma budowę modułową. Dzięki temu można podłączyć tylko te moduły, które są potrzebne do aplikacji, i nie podłączać tych, które nie będą używane (rys. 2.4).



Rys. 2.4. Modułowa budowa frameworka Spring [4]

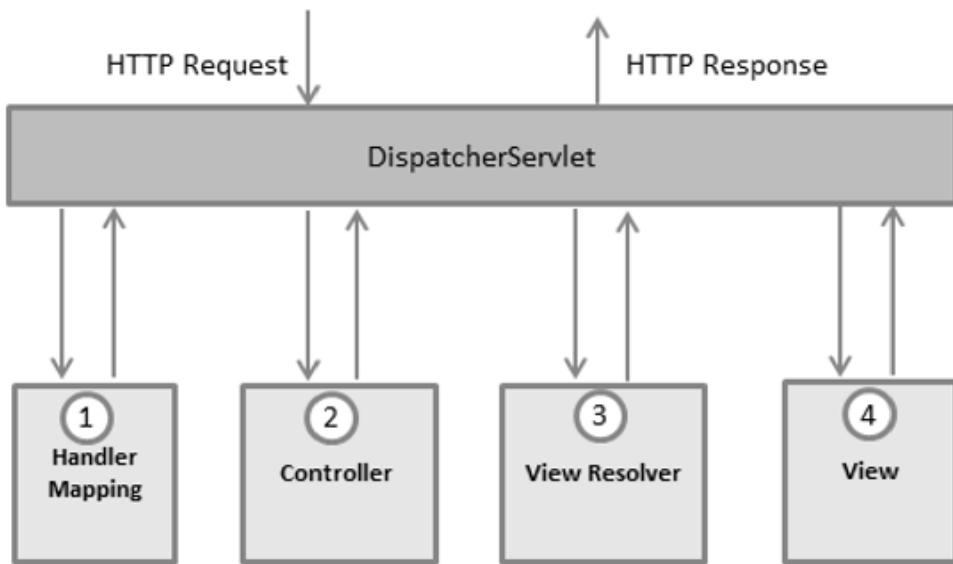
2.2.1. Spring MVC

Framework Spring MVC zapewnia architekturę Model-View-Controller, przy użyciu luźno powiązanych gotowych komponentów. Wzorzec MVC oddziela aspekty aplikacji (logikę wejściową, logikę biznesową i logikę interfejsu użytkownika), zapewniając jednocześnie swobodną komunikację między nimi. Poniżej jest opisany każdy z komponentów tej architektury:

- **Model** inkapsuluje (łączy) dane aplikacji, to znaczy że dane składają się z POJO [5].
- **View** (wyświetlanie, widok) jest odpowiedzialny za wyświetlanie danych modelu, za pomocą generowania kodu HTML, który widzimy w naszej przeglądarce [5].

- **Controller** (kontroler) przetwarza żądanie użytkownika, tworzy odpowiedni model i przekazuje go do wyświetlenia [5].

Cała logika Spring MVC opiera się na DispatcherServlet, który odbiera i przetwarza wszystkie żądania HTTP (z interfejsu użytkownika) i odpowiada na nie. Przepływ pracy przetwarzania żądania DispatcherServlet jest zilustrowany na rysunku 2.5.



Rys. 2.5. Schemat przetwarzania żądania DispatcherServlet [5]

Poniżej znajduje się sekwencja zdarzeń odpowiadająca przychodzącom żądaniom HTTP:

- Po otrzymaniu żądania HTTP, DispatcherServlet uzyskuje dostęp do interfejsu HandlerMapping, który określa, który kontroler powinien zostać wywołany, a następnie wysyła żądanie do potrzebnego kontrolera.
- Kontroler akceptuje żądanie i wywołuje odpowiednią metodę narzędzia na podstawie GET lub POST metody. Wywołana metoda określa dane modelu, na podstawie logiki biznesowej i zwraca nazwę widoku do DispatcherServlet.
- Korzystając z interfejsu ViewResolver, DispatcherServlet decyduje, którego widoku należy użyć na podstawie otrzymanej nazwy.
- Po utworzeniu widoku DispatcherServlet wysyła dane modelu jako atrybuty do widoku, który ostatecznie jest wyświetlany w przeglądarce.

2.2.2. Technologia Spring Security

Spring Security to platforma Java/JavaEE, która zapewnia mechanizmy budowania systemów uwierzytelniania i autoryzacji, a także inne funkcje bezpieczeństwa dla aplikacji korporacyjnych, utworzonych za pomocą Spring Frameworka [6].

Proces uwierzytelniania w Spring Security odbywa się według kroków:

- Nazwa użytkownika i hasło są uzyskiwane i łączone w instancję klasy UsernamePasswordAuthenticationToken.
- Token jest przekazywany do instancji AuthenticationManager w celu weryfikacji.
- AuthenticationManager zwraca zupełnie wypełnioną instancję uwierzytelnienia, jeśli uwierzytelnianie się powiedzie.
- Kontekst bezpieczeństwa jest ustalany przez wywołanie SecurityContextHolder.getContext().SetAuthentication(...), do którego przekazywana jest zwrócona instancja uwierzytelniania.

2.2.3. Technologia Spring Boot

Autorzy Spring postanowili udostępnić programistom niektóre narzędzia, które automatyzują proces konfiguracji i przyspieszają proces tworzenia i wdrażania aplikacji Spring, zwanych łącznie Spring Boot.

Spring Boot to przydatny projekt, którego celem jest uproszczenie tworzenia aplikacji opartych na Spring. Pozwala na stworzenie aplikacji internetowej w najprostszym sposobie, wymagający minimalnego wysiłku ze strony programistów do skonfigurowania i napisania kodu. Spring Boot ma świetną funkcjonalność, a jego najważniejsze cechy to:

- Łatwe zarządzanie zależnościami: aby przyspieszyć proces zarządzania zależnościami, Spring Boot domyślnie pakuje niezbędne zależności stron dla każdego rodzaju aplikacji opartych na Spring i dostarcza je deweloperowi za pośrednictwem tak zwanych pakietów startowych (spring-boot-starter-web, spring-boot-starter-data-jpa itp.) [7].
- Automatyczna konfiguracja: po wybraniu odpowiedniego pakietu startowego, Spring Boot próbuje automatycznie skonfigurować aplikację Spring, na podstawie dodanych zależności jar. Na przykład, jeśli dodać Spring-boot-starter-web, Spring Boot automatycznie skonfiguruje zarejestrowane komponenty bean, takie jak DispatcherServlet, ResourceHandlers, MessageSource. Jeśli użyć Spring-boot-starter-jdbc, Spring Boot automatycznie rejestruje komponent DataSource,

EntityManagerFactory, TransactionManager bean i odczytuje informacje o łączaniu się z bazą danych z pliku application.properties [7].

- Obsługa wbudowanego serwera aplikacji: każda aplikacja internetowa Spring Boot zawiera wbudowany serwer internetowy. Programiści nie muszą się już martwić o konfigurację kontenera serwletu i wdrożenie na nim aplikacji. Teraz aplikacja może uruchomić się sama, jako plik wykonywalny jar, przy użyciu wbudowanego serwera [7].

2.3. Relacyjna baza danych

Relacyjna baza danych to zestaw połączonych tabel, z których każda zawiera informacje o obiektach określonego typu. Każdy wiersz tabeli zawiera dane o jednym obiekcie (na przykład o samochodzie, komputerze, kliencie), a kolumny tabeli zawierają różne cechy tych obiektów, tzw. atrybuty (na przykład numer silnika, markę procesora, telefony firm lub klientów) [8].

Wiersze tabeli nazywane są rekordami. Wszystkie wpisy tabeli mają taką samą strukturę - składają się z pól (elementów danych), w których przechowywane są atrybuty obiektu (rys. 2.6). Każde pole rekordu zawiera jedną cechę obiektu i reprezentuje określony typ danych (na przykład ciąg tekstowy, liczbę, datę). Klucz podstawowy służy do identyfikacji rekordów. Klucz podstawowy to zestaw pól tabeli, których kombinacja wartości jednoznacznie identyfikuje każdy rekord w tabeli [8].

idteacher	name	surname	account_id	dateOfBirth
1	Jakub	Czachor	2	1988-07-14
2	Andrzej	Stec	3	1986-08-09
30	Łukasz	Wierzbicki	NULL	1970-06-15
35	Aneta	Dudak	NULL	1980-06-13
167	Małgorzata	Koszorz	NULL	2000-06-17
NULL	NULL	NULL	NULL	NULL

Rys. 2.6. Wiersze z przykładowej tabeli

W tym przykładzie identyfikacja rekordu odbywa się za pomocą jednego pola w wierszu (klucz jednopolowy). Możliwa jest także identyfikacja rekordu za pomocą więcej niż jednej kolumny (klucze wielopolowe). Taki sposób jest bardziej złożony, i identyfikacja tym sposobem odbywa się za pomocą unikalnego zestawu komórek tworzących klucz w wierszu.

Tabele w relacyjnych bazach danych mają wiele właściwości. Najważniejsze z nich to:

- Tabela nie może mieć dwóch identycznych wierszy. W matematyce tabele z tą właściwością nazywane są relacjami, stąd nazwa - relacyjna.
- Kolumny są ułożone w określonej kolejności, która jest definiowana podczas tworzenia tabeli. Tabela może nie zawierać ani jednego wiersza, ale musi mieć co najmniej jedną kolumnę.
- Każda kolumna ma unikalną nazwę (w obrębie tabeli), a wszystkie wartości w jednej kolumnie są tego samego typu (liczba, tekst, data itp.).

Relacje między tabelami bazy danych mogą istnieć między dwiema lub więcej tabelami. Relacje określają, że dla każdego rekordu w głównej tabeli może być jeden lub więcej rekordów w podtablicy. Takie relacje są zrealizowane za pomocą klucza obcego (foreign key) i klucza podstawowego (primary key). Klucz obcy to jedno lub kilka pól (kolumn) tabeli, które odwołują się do pola lub pól klucza podstawowego w innej tabeli. Istnieją trzy typy relacji między tabelami bazy danych:

- jeden do wielu (one to many),
- wiele do wielu (many to many),
- jeden do jednego (one to one) [8].

Na poniższej tabeli pokazany jest opis i przykład każdego typu relacji.

Tabela 2.1. Opis relacji między tabelami relacyjnej bazy danych

Typ relacji	Opis	Przykład
Jeden do wielu	Rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B, ale rekord w tabeli B ma tylko jeden dopasowany rekord w tabeli A	Jeden autor może mieć wiele książek, a książka może mieć jednego autora.
Wiele do wielu	Rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B i tak samo rekord w tabeli B może mieć wiele dopasowanych do niego rekordów z tabeli A.	Jeden autor może mieć wiele książek, a książka może mieć wielu autorów.

Jeden do jednego	Rekord w tabeli A może mieć tylko jeden dopasowany rekord z tabeli B, i tak samo każdy rekord w tabeli B może mieć tylko jeden dopasowany rekord z tabeli A.	Jeden autor może mieć jedną książkę, i książka może mieć jednego autora.
-------------------------	--	--

Do pracy z danymi stosuje się systemy zarządzania bazami danych (SZBD). Główne funkcje SZBD:

- definicja danych (opis struktury bazy danych),
- przetwarzanie danych,
- zarządzanie danymi.

Opracowanie struktury bazy danych jest najważniejszym zadaniem do rozwiązania podczas projektowania bazy danych. Struktura bazy danych (zestaw, forma i relacje jej tabel) jest jedną z głównych cech projektowych podczas tworzenia aplikacji przy użyciu bazy danych. Struktura bazy danych utworzona przez programistę jest opisana w języku definicji danych SZBD.

Dowolny system SZBD umożliwia wykonywanie następujących operacji na danych:

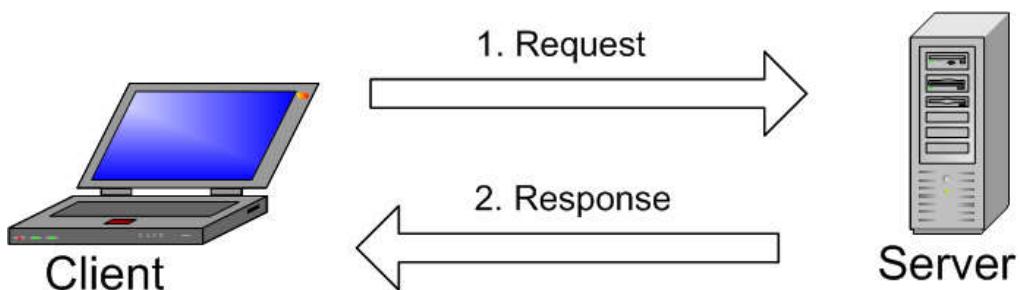
- dodawanie rekordów do tabel,
- usuwanie rekordów z tabeli,
- aktualizowanie wartości niektórych pól w jednym lub kilku rekordach w tabelach bazy danych,
- wyszukiwanie jednego lub więcej rekordów, które spełniają dany warunek.

Aby wykonać te operacje, używany jest mechanizm zapytania. Wynikiem wykonania zapytania jest albo zestaw rekordów wybranych zgodnie z określonymi kryteriami, albo zmiany w tabelach. Zapytania do bazy danych są tworzone w specjalnie do tego stworzonym języku, który nazywa się „ustrukturyzowany język zapytań” (SQL - Structured Query Language) [9].

2.4. System zarządzania bazą danych MySQL

MySQL jest jednym z najpopularniejszych otwartych systemów zarządzania bazami danych, z modelem klient-serwer. Chociaż jest stary, ale jest wykorzystywany w wielu projektach na dzień dzisiejszy. Otwarte źródło systemy oznacza, że można go używać i modyfikować. Każdy może zainstalować oprogramowanie. Można także uczyć się i dostosowywać kod źródłowy, aby lepiej odpowiadał różnym potrzebom [10].

Komputery, które instalują i uruchamiają oprogramowanie SZBD, nazywane są klientami. Gdy potrzebują dostępu do danych, łączą się z serwerem SZBD. Taki system nazywa się klient-serwer, i jego działanie przykładowo pokazane na rysunku 2.7.



Rys. 2.7. Schemat działania systemu klient-serwer [10]

Obraz wyjaśnia podstawową strukturę klient-serwer. Jedno lub więcej urządzeń (klientów) jest połączonych do serwera za pośrednictwem określonej sieci. Każdy klient może złożyć żądanie z graficznego interfejsu użytkownika (GUI) na swoich ekranach, a serwer da pożądany wynik, jeśli obie strony zrozumieją instrukcję. Bez wchodzenia w aspekty techniczne główne procesy zachodzące w środowisku MySQL są takie same:

- MySQL tworzy bazę danych do przechowywania danych i zarządzania nimi, która określa relacje między poszczególnymi tabelami.
- Klienci mogą tworzyć zapytania, wprowadzając określone polecenia SQL w MySQL.
- Aplikacja serwera odpowie żądanymi informacjami i pojawi się po stronie klienta.

Na dzień dzisiejszy istnieją wiele SZBD. Chociaż inne systemy są nowsze, MySQL zostaje się jednym z najpopularniejszych systemów, ponieważ ma takie zalety:

- **Elastyczność i łatwość użytkowania:** jest możliwość zmiany koda źródłowego, aby spełnić własne oczekiwania, i nie jest potrzebnym płacić za ten poziom swobody, w tym opcje aktualizacji do rozszerzonej wersji komercyjnej [10].

- **Wysoka wydajność:** szeroka gama serwerów klastrowych obsługuje MySQL. Bez względu na to, czy przechowywana jest duża ilość danych w handlu elektronicznym, czy wykonywane są ciężkie analizy biznesowe, MySQL pomaga z optymalną prędkością [10].
- **Standard branżowy:** Branże używają MySQL od wielu lat, co oznacza, że istnieją bogate zasoby dla doświadczonych programistów. Użytkownicy MySQL mogą liczyć na szybki rozwój oprogramowania [10].
- **Bezpieczeństwo:** bezpieczeństwo danych powinno być głównym celem przy wyborze odpowiedniego oprogramowania SZBD. Korzystając z systemu zarządzania dostępem i kontami, MySQL zapewnia wysoki poziom bezpieczeństwa. Dostępna jest weryfikacja oparta na hoście i szyfrowanie hasła [10].

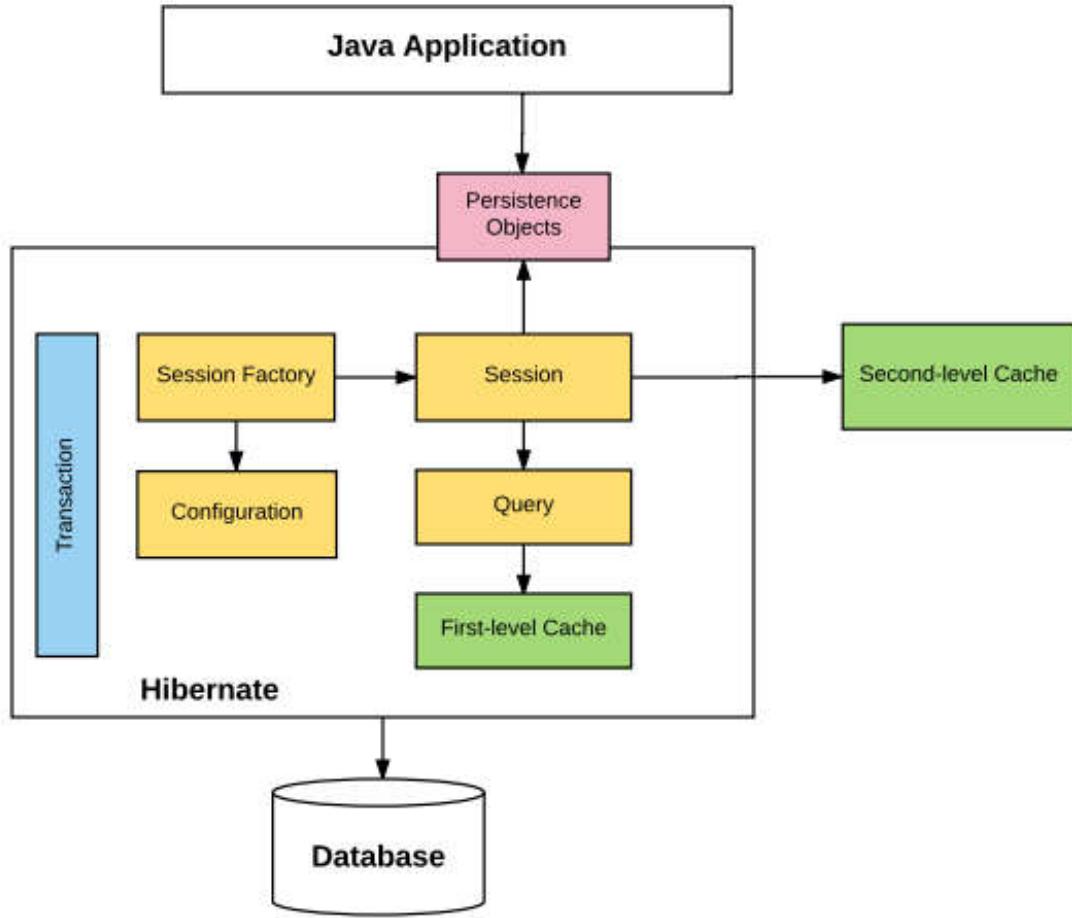
2.5. Hibernate framework

Hibernate - to biblioteka przeznaczona do zadań mapowania obiektowo-relacyjnego. Mówiąc najprościej, hibernate pozwala pracować z bazą danych nie bezpośrednio, jak to działa z biblioteką JDBC, a prezentuje tabele bazy danych jako klasy java. Hibernate jest jedną z najpopularniejszych implementacji JPA specyfikacji [11].

JPA (Java Persistence API) - to specyfikacja Java EE i Java SE, która opisuje system do zarządzania przechowywaniem obiektów Java w tabelach relacyjnych baz danych. Sama Java nie zawiera implementacji JPA, ale istnieje wiele implementacji tej specyfikacji od różnych firm (otwartych i nie). Nie jest to jedyny sposób zapisywania obiektów Java w bazach danych (systemy ORM), ale jeden z najpopularniejszych w świecie Java.

Hibernate pozwala na tworzenia encję klasy bazy danych, oznaczyć ją specjalnymi adnotacjami i framework zrobi wszystko sam. Nazywa się to ORM - Object Relational Mapping. Potrzebne tylko utworzyć klasy pasujące do tabel w bazie danych i napisać metody, które potrzebne zrobić z danymi (pobierz, usuń, utwórz, zaktualizuj). Klasa z adnotacją „@Entity” jest zwykłą POJO klasą, z prywatnymi polami, getterami i seterami dla nich. Klasa musi mieć nieprywatnego konstruktora bez parametrów (lub domyślnego konstruktora) i musi mieć klucz podstawowy, tj. coś, co jednoznacznie zidentyfikuje każdy rekord tej klasy w bazie danych.

Hibernate sam generuje zapytanie SQL i wykonuje je. I nie ma znaczenia, jakiej bazy danych używamy. Pozwala to przełączać bazy danych i nie martwić się o to, że kod nie działa. Poniższy diagram podsumowuje główne elementy składowe w architekturze Hibernate.



Rys. 2.8. Diagram ze składowymi elementami Hibernate [12]

2.6. Lokalny serwer TomCat

Tomcat to kontener serwletów typu „open source”, który działa również jako serwer WWW. Tomcat implementuje technologie Java Servlet i JavaServer Pages (JSP).

Większość aplikacji Java jest uruchamiana za pomocą wiersza polecenia i wykonuje pewne działania. Takie aplikacje implementują jedną predefiniowaną funkcję, po której nie są już wykonywane. Takie programy z reguły mają główną metodę, dzięki której można je uruchomić [13].

Aplikacja internetowa została zaprojektowana do interakcji z klientem. Jeśli istnieje żądanie od klienta, jest ono przetwarzane, a odpowiedź jest wysyłana do użytkownika. Jeśli nie, aplikacja jest bezczynna.

W rzeczywistości TomCat jest to aplikacja Java, która zajmuje się otwieraniem portu do interakcji z klientem, konfigurowaniem sesji, liczbą żądań, długością nagłówka i wieloma innymi operacjami.

Tomcat ma komponenty, które wykonują określone funkcje i te komponenty są opisane poniżej:

- **Catalina:** dzięki temu komponentowi programiści mają możliwość wdrażania swoich programów w kontenerze. Catalina implementuje specyfikację Servlet API, podstawową technologię internetową w programowaniu Java. W rzeczywistości Catalina jest pojemnikiem serwletu w Tomcat [13].
- **Jasper:** dzięki temu komponentowi programista wykorzystuje technologię JSP. To jest jak pliki HTML, tylko że mają osadzony kod Java, który można wykonać, gdy strona jest wysyłana do użytkownika. Umożliwia to dynamiczne osadzanie dowolnych danych na stronie. Jasper przekształca kod Java w HTML, a także śledzi zmiany i automatycznie je aktualizuje [13].
- **Coyote:** jest to ważny komponent, który nasłuchuje żądań HTTP od klienta na określonym porcie, udostępnia te dane do przetworzenia w aplikacji, a także zwraca odpowiedzi użytkownikom. Oznacza to, że Coyote implementuje funkcjonalność serwera HTTP [13].

2.7. Technologia JSP

Java Server Pages wprowadza technologię, która pozwala tworzyć dynamiczne strony internetowe. Początkowo JSP (wraz z serwletami) na początku Java EE było dominującym podejściem do tworzenia stron internetowych Java. Chociaż teraz ustąpiły miejsca innej technologii - JSF, niemniej jednak strony JSP są nadal szeroko stosowane [14].

Zasadniczo strona Java Server Page lub JSP to kod HTML przeplatany z kodem Java. Jednocześnie strony JSP nie są standardowymi stronami HTML. Gdy żądanie dotrze do określonej strony JSP, serwer przetwarza ją, generuje z niej kod HTML i wysyła do klienta. W rezultacie użytkownik po wejściu na stronę JSP widzi w swojej przeglądarce zwykłą stronę HTML.

Podobnie jak zwykłe statyczne strony internetowe, pliki JSP muszą być umieszczone na serwerze internetowym, do którego zwykli użytkownicy mogą uzyskać dostęp za pośrednictwem protokołu http, na przykład wpisując żądany adres w pasku

adresu przeglądarki. Jednak aby serwer mógł przetwarzać pliki JSP, musi używać silnika JSP (silnika JSP), zwanego również kontenerem JSP. Istnieje wiele silników JSP i wszystkie implementują tę samą specyfikację i ogólnie działają tak samo [14].

2.8. Technologia Maven Builder

Maven jest narzędziem do budowania projektu Java: kompilowania, tworzenia jar, tworzenia pakietu dystrybucyjnego dla programu i generowania dokumentacji. Proste projekty można budować za pomocą wiersza poleceń. Jeśli budować duże projekty z wiersza poleceń, polecenie budowania będzie bardzo długie, więc czasami ono jest zapisywane w skrypcie bat / sh. Ale takie skrypty zależą od platformy. Aby pozbyć się tej zależności i uprościć pisanie skryptu, do budowy projektu wykorzystywane są takie narzędzia jak Maven [15].

Maven zapewnia deklaratywne, ale nie bezwzględne zebranie projektu. Oznacza to, że pliki projektu pom.xml zawierają opis deklaratywny, a nie poszczególne polecenia. Wszystkie zadania przetwarzania plików Maven są wykonywane przez wtyczki.

Plik pom.xml to XML zawierający informacje o szczegółach projektu i konfiguracjach użytych do utworzenia projektu w Maven. Zawsze znajduje się w katalogu podstawowym projektu. Ten plik zawiera także zadania i wtyczki. Podczas wykonywania zadań, Maven szuka pliku pom.xml w katalogu podstawowym projektu. Czyta ją i otrzymuje niezbędne informacje, po których wykonuje zadania [15].

Element główny „<project>” to jest schemat ułatwiający edycję i sprawdzanie poprawności. W środku taga „<projekt>” są podstawowe i wymagane informacje o projekcie. Plik pom.xml zawiera następującą informację:

zależności projektu (project dependencies),

- Wtyczki (plugins),
- Zadania (goals),
- Wersję projektu (project version),
- Programiści (developers),
- Profil tworzenia (build profiles).

Na rysunku 2.9 przedstawiono przykładową zawartość pliku pom.xml:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>2.1.8.RELEASE</version>
9          <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>com.example</groupId>
12     <artifactId>demo</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <packaging>war</packaging>
15     <name>demo</name>
16     <description>Demo project for Spring Boot</description>
17
18     <properties>
19         <java.version>1.8</java.version>
20     </properties>
21
22     <dependencies>
23         <dependency>
24             <groupId>org.springframework.boot</groupId>
25             <artifactId>spring-boot-starter-web</artifactId>
26         </dependency>

```

Rys. 2.9. Zawartość pliku "pom.xml"

2.9. Cascading Style Sheets (CSS)

CSS to formalny język używany do opisywania wyglądu dokumentu utworzonego przy użyciu języka znaczników (HTML, XHTML, XML).

Celem CSS jest oddzielenie tego, co decyduje o wyglądzie strony od jej zawartości. Jeśli dokument został utworzony tylko przy użyciu HTML, to on musi określać nie tylko każdy element, ale także sposób jego wyświetlania (kolor, czcionka, położenie bloku itp.). Jeśli połączone są kaskadowe arkusze stylów, wówczas HTML opisuje tylko kolejność obiektów. A CSS odpowiada za wszystkie ich właściwości. W HTML wystarczy napisać klasę bez wymieniania wszystkich stylów za każdym razem [16].

Taka technologia:

- zapewnia stosunkowo prosty i szybki rozwój, ponieważ raz utworzony dokument CSS można zastosować na wielu stronach;
- zwiększa elastyczność i łatwość edycji - po prostu można edytować CSS, aby widok zmieniał się wszędzie;
- upraszcza kod, zmniejszając powtarzalność elementów. Łatwiej jest czytać programistom i wyszukiwarkom;
- przyspiesza czas ładowania, ponieważ CSS może być buforowany przy pierwszym otwarciu, a kolejne struktury i dane są odczytywane w kolejnych;
- zwiększa liczbę wizualnych rozwiązań do prezentacji treści;

- zapewnia możliwość łatwego zastosowania różnych stylów do tego samego dokumentu (na przykład, stworzenie dostosowanej wersji dla urządzeń mobilnych lub specjalnych stylów dla osób niedowidzących).

2.10. Środowisko programistyczne JetBrains IntelliJ IDEA

Oprogramowanie JetBrains IntelliJ IDEA to wiodące środowisko szybkiego programowania Java. IntelliJ IDEA to zaawansowany technologicznie kompleks ściśle zintegrowanych narzędzi programistycznych, w tym inteligentny edytor źródeł z zaawansowanymi narzędziami automatyzacji, potężne narzędzia refaktoryzacji kodu, wbudowane wsparcie dla technologii J2EE, mechanizmy integracji ze środowiskiem testowym Ant / JUnit i systemami kontroli wersji, unikalne narzędzie do optymalizacji i weryfikacji kodu Code Inspection, a także innowacyjny projektant wizualny interfejsów graficznych [17].

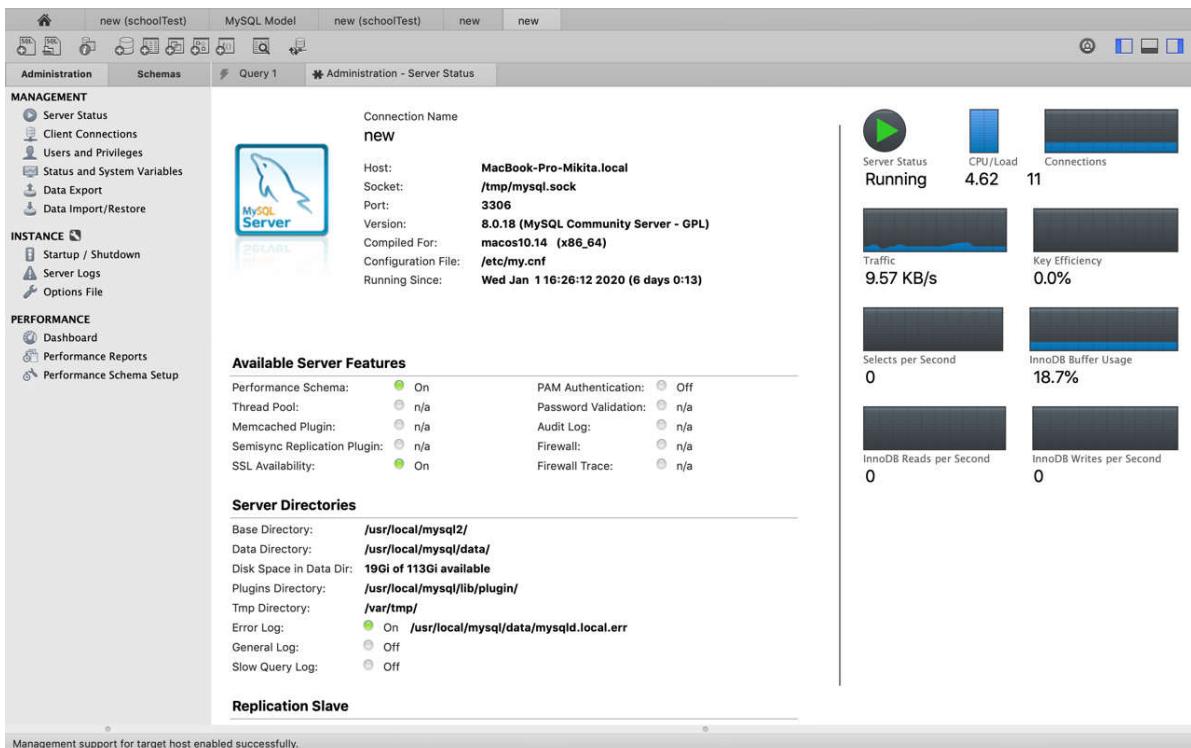
Unikalne funkcje JetBrains IntelliJ IDEA odciążają programistę od rutynowej pracy, pomagają w odpowiednim czasie eliminować błędy i poprawiają jakość kodu, podnosząc produktywność programisty na nowy poziom.

2.11. Wizualny edytor baz danych MySQL Workbench

MySQL Workbench – jest programem służącym do wizualnego projektowania, modelowania i tworzenia baz danych dla systemu baz danych MySQL. Możliwości tego programu są następujące:

- Pozwala na przedstawienia modeli bazy danych w formie graficznej.
- Wizualny i funkcjonalny mechanizm ustalania relacji między tabelami, w tym wiele do wielu, dzięki tworzeniu tabeli połączeń.
- Wygodny edytor zapytań SQL, który pozwala natychmiast wysłać je na serwer i otrzymać odpowiedź w postaci tabeli.
- Możliwość edycji danych w tabeli w trybie wizualnym.

Na rysunku 2.10 przedstawiono przykładowy zrzut programu MySQL Workbench:



Rys. 2.10. Widok programu MySQL Workbench

2.12. System kontroli wersji Git i GitHub

Systemy kontroli wersji (VCS, Version Control Systems) pozwalają programistom zapisywać wszystkie zmiany wprowadzone w kodzie. Dlatego w przypadku opisany powyżej mogą po prostu przywrócić kod do stanu roboczego, zamiast spędzać godziny na szukaniu małego błędu lub błędów, które psują cały kod.

SKW pozwalają również wielu programistom pracować nad jednym projektem i zapisują zmiany, aby wszyscy mogli śledzić to, nad czym pracują.

Git to rozproszony system kontroli wersji, który umożliwia programistom śledzenie zmian w plikach i współpracę z innymi programistami. Został stworzony aby umożliwić innym programistom możliwość wnosić zmiany do jądra Linuksa. Git jest znany z szybkości, prostoty projektowania, obsługi nieliniowego rozwoju, całkowitej decentralizacji i zdolności do wydajnej pracy z dużymi projektami [18].

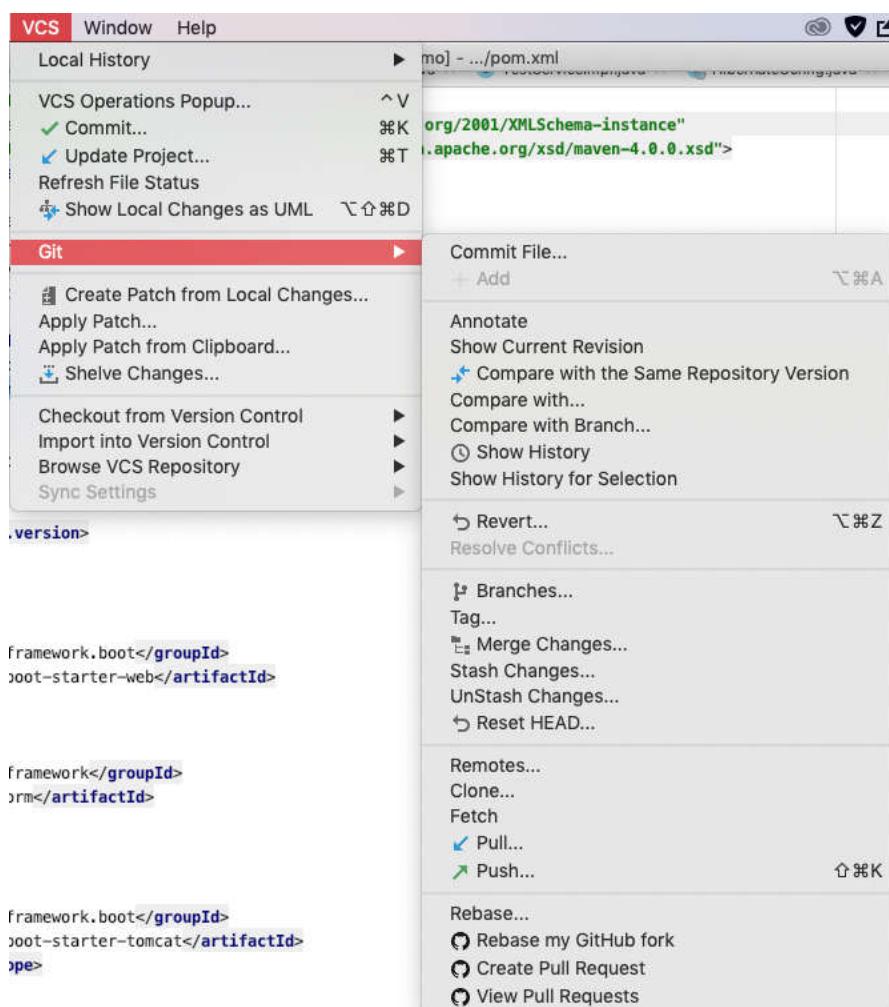
Git wyróżnia się spośród innych SKW ze względu na podejście do pracy z danymi. Większość innych systemów przechowuje informacje jako listę zmian w plikach. Zamiast tego podejście Gita do przechowywania danych przypomina bardziej migawkę małej systemy plików. Za każdym razem, gdy jest zapisywany status projektu w Git, system zapamiętuje wygląd każdego pliku w tym momencie i zapisuje link do tej migawki.

GitHub to usługa hostingowa repozytoriów, która ma wszystkie funkcje rozproszonej kontroli wersji i funkcje zarządzania źródłami. Jest zwykle używany w połączeniu z Git i umożliwia programistom zapisywanie kodu online, a następnie interakcję z innymi programistami w różnych projektach [18].

GitHub oferuje także kontrolę dostępu, śledzenie błędów, zarządzanie zadaniami i strony wiki dla każdego projektu. Celem GitHub jest zwiększenie zaangażowania programistów.

Dostęp do projektu przesłanego do GitHub można uzyskać za pomocą interfejsu, wiersza poleceń Git i poleceń Git. Istnieją również inne funkcje, takie jak dokumentacja, żądania ściągania zmian (pull request), historia zatwierdzania, integracja z wieloma popularnymi usługami, powiadomienia e-mail itp.

Środowisko programistyczne IntelliJ Idea zawiera funkcje wspomagające współpracę z różnymi systemami kontroli wersji. Za pomocą opcji VCS na pasku menu programu Idea można ustawić konfiguracje potrzebną do współpracy Git i IntelliJ Idea (rys. 2.11).



Rys. 2.11. Opcja VCS z menu programu IntelliJ Idea, z rozwiniętą listą funkcji Git

3. Etapy pracy, prowadzące do powstania serwisu

Proces realizacji programu polega na stworzeniu serwisu na podstawie programów z architekturą MVC, za pomocą odpowiednich narzędzi, które są opisane wcześniej. Najpierw dla takiego serwisu należy stworzyć projekt w środowisku programistycznym IntelliJ Idea, który zapewnia wszystkie potrzebne narzędzia do pracy.

Następnym etapem było zaprojektowanie i stworzenie relacyjnej bazy danych w MySQL oraz zapewnienie komunikacji tej bazy z projektem Java. Baza danych, dla serwisu do zarządzania szkołą, musi przechowywać wszystkie dane dotyczące toku szkolenia: dane o uczniach i nauczycielach, o ocenach i obecnościach, o sprawdzianach i prowadzonych lekcjach itp.

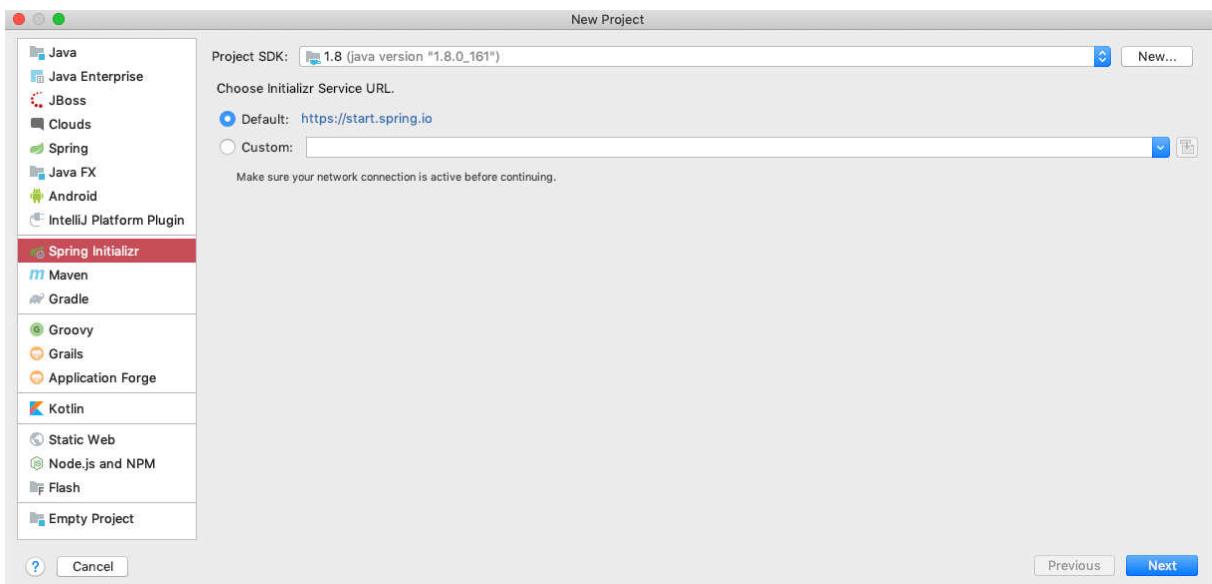
Kolejnymi etapami są zaprojektowania funkcjonalności potrzebnych dla serwisu oraz realizacja tej funkcjonalności. Polega to na tworzeniu potrzebnych klas Java oraz dodawania do projektu plików, za pomocą których można byłoby ustalić niezbędną konfigurację, stworzyć modele bazy danych oraz zapewnić komunikację z nią, stworzyć logikę działania serwisu, skonfigurować kontrolery. W tym etapie została rozbudowana hierarchia drzewa projektu, zostały stworzone pakiety Java, które przechowują odpowiednie klasy. W tym momencie kod aplikacji został zorganizowany i poprawnie przydzielony do różnych klas.

Jednym z ostatnich etapów tworzenia serwisu była część klienta – widok dla użytkowników systemu. Zostały dodane pliki JSP, które generują stronę HTML z podanymi danymi. Zostały dodane pliki stylów CSS, które opisują właściwości obiektów HTML.

W następnych podrozdziałach będzie szczegółowo opisany każdy z etapów.

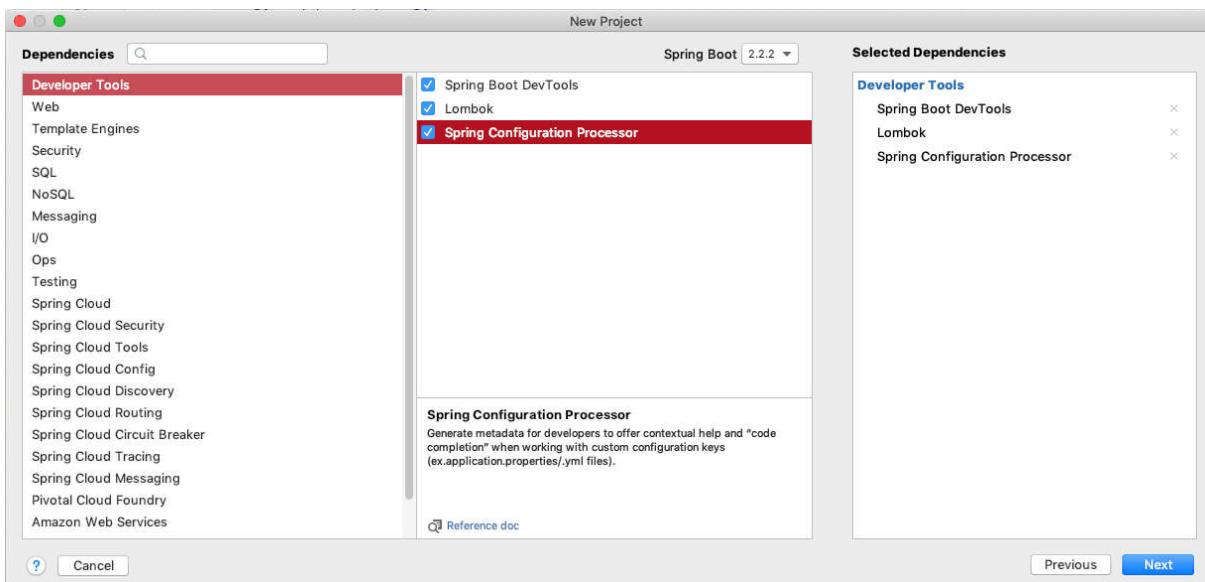
3.1. Utworzenie nowego projektu w IntelliJ Idea

Środowisko programistyczne IntelliJ Idea pozwala na tworzenie projektów różnego typu. Dla serwisu zarządzania szkołą potrzebnym było wybrać odpowiednie ustawienia podczas zaczynania projektu. W oknie „New Project” programu został wybrany punkt „Spring Initializr”, który pomaga utworzyć Spring Boot aplikację, wraz z dodatkowymi zależnościami i charakterystykami. Na rysunku 3.1 widać przykładowo pokazany proces tworzenia projektu.



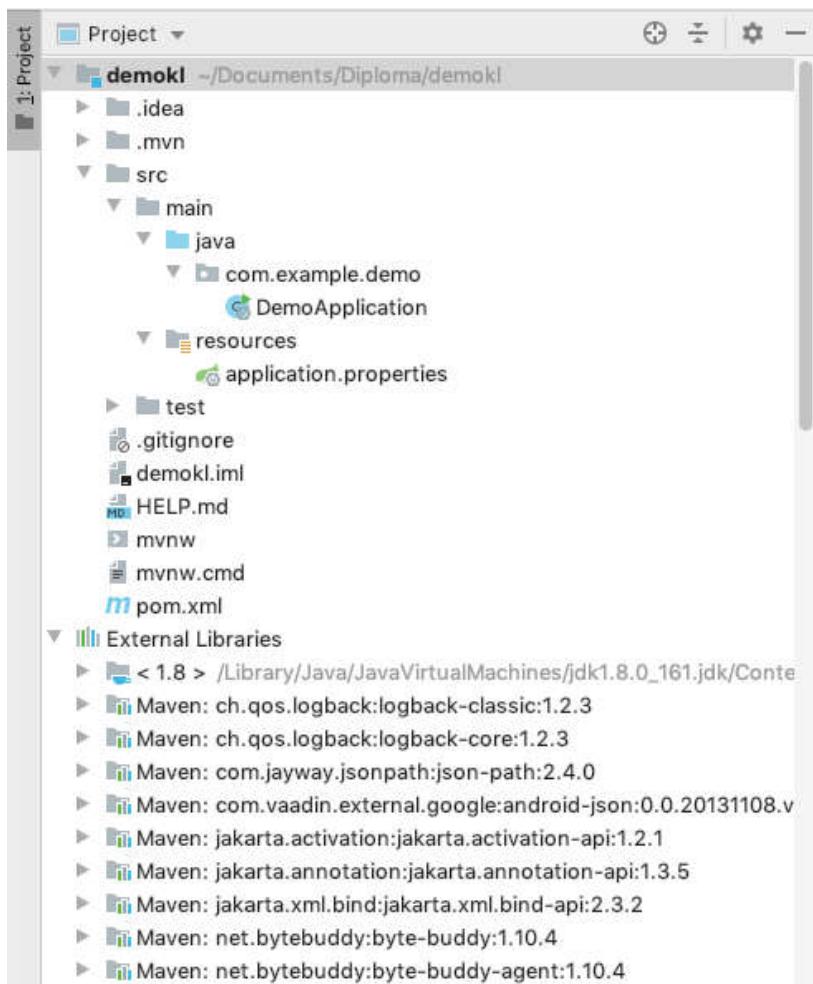
Rys. 3.1. Okno tworzenia projektu w programie IntelliJ Idea

W kolejnych krokach tworzenia projektu potrzebnym jest podać jego nazwę i wybrać katalog, gdzie będzie on znajdować się. Później można wybrać wszystkie zależności potrzebne dla tego programu. W tym przypadku zostały dodane zależności, które przedstawione są na rysunku 3.2.



Rys. 3.2. Wybór potrzebnych zależności w oknie tworzenia projektu programu IntelliJ Idea

Po zakończeniu tworzenia nowego projektu powstaje drzewo projektu, które jest pokazane na rysunku 3.3.



Rys. 3.3. Wygląd drzewa projektu po jego stworzeniu

3.2. Zaprojektowanie i utworzenie relacyjnej bazy danych

Do przechowywania różnych danych związanych z serwisem, potrzebne jest zaprojektowanie i stworzenie bazy danych. Celem zaprojektowania BD jest organizacja wszystkich tabel, które powinni być oraz związki między nimi. Wszystkie potrzebne tabele są wymienione i opisane niżej:

- **Account** – przechowuje informacje o użytkowniku, jego login i hasło oraz rolę tego użytkownika.
- **Ac_Roles** – przechowuje wszystkie role wraz z ich kluczami publicznymi.
- **Class** – przechowuje informacje o wszystkich klasach w szkole, ich nazwę oraz klucz publiczny.
- **DaysOfWeek** – zawiera wszystkie robocze dni tygodnia wraz z ID.

- **Frequency** – przechowuje wszystkie zapisane przez nauczyciela nieobecności, wraz z ich datami, z uczniami którzy ich uzyskali i na którym zajęciu.
- **HomeWork** – przechowuje wszystkie dodane przez nauczyciela zadanie domowe, wraz z datą ich dodania i datą wykonania tego zadania, także zawiera kolumnę pokazującą zajęcie, na którym zostało dodane zadanie.
- **LessonTheme** - przechowuje wszystkie dodane przez nauczyciela tematy zajęć, z ich tytułem, datą i z identyfikatorom zajęcia.
- **LessonTime** – zawiera godziny rozpoczęcia i zakończenia każdego zajęcia.
- **Marks** - przechowuje wszystkie dodane przez nauczyciela oceny, wraz z datą ich dodania, z uczniami które ich uzyskali i na którym zajęciu.
- **Student** – zawiera wszystkie informacje o uczniu: imię, nazwisko, datę urodzenia, adres, email, numer telefonu, ID klasy, w której znajduje się uczeń.
- **Subject** – przechowuje informacje o wszystkich przedmiotach w szkole, wraz z ich ID.
- **Teacher** - zawiera wszystkie informacje o uczniach: imię, nazwisko, data urodzenia, adres, email, numer telefonu, ID klasy której jest wychowawcą.
- **Teacher_Subject_Class** – ta tabela przechowuje informację o zajęciach, nauczycielach które ich prowadzą, dla której klasy i jaki przedmiot.
- **Test** – ta tabela zawiera informacje o sprawdzianach, wraz z ich tytułem, datą kiedy one odbędą się i z identyfikatorom zajęcia.
- **TimeTable** – przechowuje plan zajęć każdej klasy: ID zajęcia, ID dnia roboczego, ID pory lekcji.

Każda tabela została dodana do DB wraz ze swoimi kolumnami i właściwościami. Na następnej tabeli można zobaczyć, jakie kolumny i właściwości zawiera każda tabela bazy danych.

Tabela 3.1. Opis oraz szczegóły każdej tabeli bazy danych

Nazwa tabeli	Kolumny	Właściwości	Typy danych
Account	idAccount	PRIMARY KEY NOT NULL UNIQUE	BIGINT
	Username	UNIQUE NOT NULL	VARCHAR
	Password	NOT NULL	VARCHAR
	Enabled	-	TINYINT
	accountRole_idac_Roles	FOREIGN KEY	BIGINT
Ac_Roles	idac_Roles	PRIMARY KEY	BIGINT
	role	NOT NULL	VARCHAR
Class	idclass	PRIMARY KEY	BIGINT
	name	NOT NULL	VARCHAR
DaysOfWeek	id	PRIMARY KEY	BIGINT
	name	NOT NULL	VARCHAR
Frequency	idfrequency	PRIMARY KEY	BIGINT
	Date	NOT NULL	DATE
	frequency	NOT NULL	VARCHAR
	student_idstudent	FOREIGN KEY	BIGINT
	subject_idsubject	FOREIGN KEY	BIGINT
HomeWork	idHomeWork	PRIMARY KEY	BIGINT
	homeWork	NOT NULL	VARCHAR
	tsc_id	FOREIGN KEY	BIGINT
	dateAdded	NOT NULL	DATE
	dateDue	NOT NULL	DATE
LessonTheme	idLessonTheme	PRIMARY KEY	BIGINT
	date	NOT NULL	DATE
	theme	NOT NULL	VARCHAR
	tsc_id	FOREIGN KEY	BIGINT
LessonTime	idLessonTime	PRIMARY KEY	BIGINT
	Count	NOT NULL	INT
	timeStart	NOT NULL	TIME
	timeEnd	NOT NULL	TIME

Marks	idMark	PRIMARY KEY	BIGINT
	mark	NOT NULL	INT
	student_idstudent	FOREIGN KEY	BIGINT
	subject_idssubject	FOREIGN KEY	BIGINT
	date	NOT NULL	DATE
Student	idStudent	PRIMARY KEY	BIGINT
	Name	NOT NULL	VARCHAR
	Surname	NOT NULL	VARCHAR
	account_idaccount	FOREIGN KEY	BIGINT
	aClass_idclass	FOREIGN KEY	BIGINT
	address	-	VARCHAR
	dateOfBirth	-	VARCHAR
	email	-	VARCHAR
	phone	-	VARCHAR
Subject	idSubject	PRIMARY KEY	BIGINT
	name	NOT NULL	VARCHAR
Teacher	idTeacher	PRIMARY KEY	BIGINT
	Name	NOT NULL	VARCHAR
	Surname	NOT NULL	VARCHAR
	account_idaccount	FOREIGN KEY	BIGINT
	dateOfBirth	-	DATE
	aClass_idclass	FOREIGN KEY	BIGINT
	address	-	VARCHAR
	email	-	VARCHAR
	phone	-	VARCHAR
Teacher Subject Class	id	PRIMARY KEY	BIGINT
	aClass_idclass	FOREIGN KEY	BIGINT
	subject_idssubject	FOREIGN KEY	BIGINT
	teacher_idteacher	FOREIGN KEY	BIGINT
Test	idTest	PRIMARY KEY	BIGINT
	date	NOT NULL	DATE
	test	NOT NULL	VARCHAR
	Tsc_id	FOREIGN KEY	BIGINT
TimeTable	ID	PRIMARY KEY	BIGINT
	daysOfWeek_id	FOREIGN KEY	BIGINT
	teacherSubjectClass_id	FOREIGN KEY	BIGINT
	lessonTime_idLessonTime	FOREIGN KEY	BIGINT

Za pomocą frameworka Hibernate, który jest implementacją specyfikacji JPA wraz z EntityManager, tworzenie bazy danych ręcznie czy za pomocą MySQL Workbench nie jest konieczne. Bazę danych można stworzyć za pomocą klas Java, do których są przypisane odpowiednie adnotacje. Za pomocą adnotacji „@Entity” oraz „@Table”, można stworzyć klasę i dodać do niej wszystkie potrzebne zmienne z różnymi właściwościami, które muszą być dodane do tabeli. Przy pierwszym uruchomieniu serwera tabele zostaną dodane do bazy danych, wraz z ich polami i właściwościami. Poniżej, na rysunku 3.4, znajduje się przykładowa klasa, która jednocześnie jest encją bazy danych.

```

1 package com.example.demo.domain;
2
3 import ...
4
5
6 @Entity
7 @Table(name = "Student")
8 public class Student {
9
10    @Id
11    @GeneratedValue(strategy = GenerationType.AUTO)
12    @Column(name = "idstudent")
13    private Long idstudent;
14
15    @NotNull
16    private String name;
17
18    @NotNull
19    private String surname;
20
21    @OneToOne
22    private Account account;
23
24    @ManyToOne
25    private Class aClass;
26
27    private String address;
28    private String email;
29    private String phone;
30    private Date dateOfBirth;
31
32    @Override
33    public Student() {
34    }
35
36    public String getAddress() { return address; }
37
38    public void setAddress(String address) { this.address = address; }
39
40    public String getEmail() { return email; }
41
42    public void setEmail(String email) { this.email = email; }
43
44    public String getPhone() { return phone; }
45
46    public void setPhone(String phone) { this.phone = phone; }
47
48    public Date getDateOfBirth() { return dateOfBirth; }
49
50    public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }
51
52    public Long getIdstudent() { return idstudent; }
53
54    public void setIdstudent(Long idstudent) { this.idstudent = idstudent; }

```

Rys. 3.4. Encja bazy danych tabeli „Student”, przedstawiona w wyglądzie klasy Java

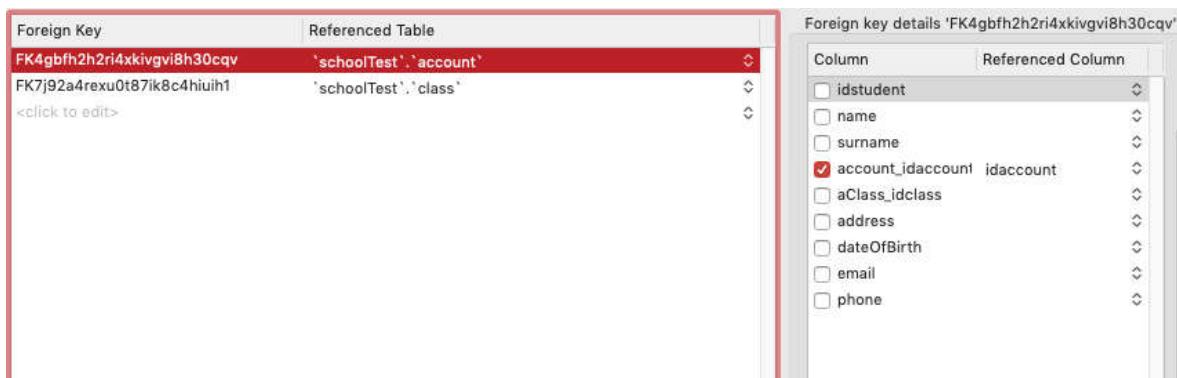
Adnotacja „@Entity” oznacza, że dana klasa jest encją. Adnotacja „@Table” jest opcjonalna i wskazuje na nazwę tablicy, która będzie wyświetlana dla tej encji. Jak można zobaczyć na rys. 2, klasa zawiera jeszcze publiczne gettery i setery oraz konstruktor bez parametrów. Jest to potrzebne dla prawidłowego działania biblioteki. Także klasa mieści zmienne, które są polami w tej encji, i te zmienne są opisane różnymi adnotacjami. Adnotacja „@Id” wskazuje na to, że ta zmienna będzie kluczem publicznym dla tej encji. Adnotacja „@GeneratedValue” wskazuje, że ta właściwość zostanie utworzona zgodnie z określona strategią. Adnotacja „@Column” wskazuje nazwę kolumny, która jest odwzorowana w encji. Adnotacja „@NotNull” oznacza zmienną jako pole, które nie może być pustym. Jeszcze w tym przykładzie są adnotacje „@OneToOne” i „@ManyToOne”. Służą one do utworzenia relacji pomiędzy encjami i utworzenia kluczy obcych. Jak można zobaczyć, te zmienne muszą być obiektami klas, które są encjami i do których trzeba stworzyć relację.

Po uruchomieniu serwera wszystkie tablice i wszystkie zmiany zostały dodane do bazy danych. Jak widać na rysunku 3.5 wszystkie pola i właściwości, opisane za pomocą adnotacji, zostały dodane do bazy danych.

Name:	Student									
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
idstudent	BIGINT(20)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
surname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
account_idaccount	BIGINT(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
aClass_idclass	BIGINT(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
address	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
dateOfBirth	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
email	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
phone	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

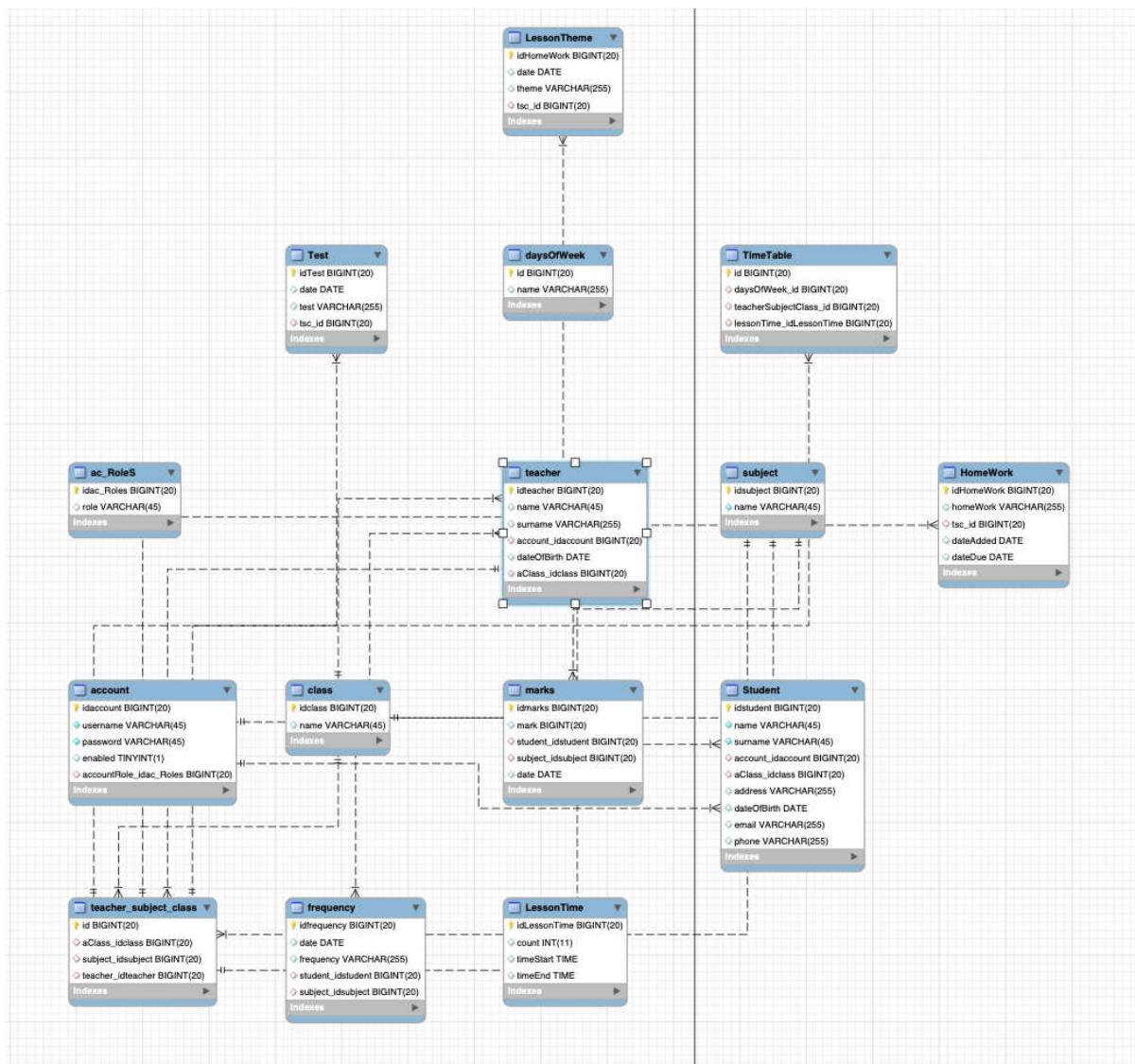
Rys. 3.5. Wygląd stworzonej tabeli „Student” w programie MySQL Workbench

Na rysunku 3.6 przedstawiono klucze obce, implementacja których odbyła się za pomocą adnotacji.



Rys. 3.6. Klucze obce tabeli „Student”

Jak można zobaczyć, wszystkie konfiguracje, zrobione za pomocą klas Java, zostały poprawnie dodane do bazy danych. W celu bardziej szczegółowego sprawdzenia poprawności bazy danych, został zrobiony diagram ERD, który powstał w programie MySQL Workbench (rys. 3.7).



Rys. 3.7. Diagram ERD bazy danych serwisu, wygenerowany w programie MySQL Workbench

Na tym diagramie dobrze widać wszystkie encje bazy danych oraz wszystkie związki pomiędzy nimi. Analizując ten diagram, można powiedzieć, że baza danych została stworzona poprawnie, i zawiera wszystkie tabele potrzebne dla serwisu.

3.3. Organizacja i podział funkcjonalności

Przed rozpoczęciem realizacji projektu, niezbędnym jest przemyślenie, organizacja i opracowanie funkcjonalności dla prawidłowego działania serwisu. Ponieważ w danym serwisie użytkownicy są podzieleni na role (rola ucznia, rola nauczyciela, rola administratora), to dla każdego musi być odpowiedni funkcjonał. Oznacza to że na przykład uczeń mógłby oglądać swoje oceny, które są wprowadzone do bazy danych przez nauczyciela. Dlatego niezbędnym jest organizacja funkcjonalności dla każdego typu użytkownika. W kolejnych tabelach znajdują się opisane wszystkie funkcje serwisu, dla każdego typu użytkownika.

Funkcjonalności ucznia:

Tabela 3.2. Opis funkcjonalności uczniów

Nº	Funkcja	Opisanie
1	Logowanie	Możliwość zalogować się do serwisu z prawami ucznia.
2	Przegląd ocen	Możliwość oglądania wszystkich ocen otrzymanych przez ucznia, ze wszystkich przedmiotów
3	Przegląd frekwencji	Możliwość oglądania uzyskanej frekwencji przez ucznia
4	Przegląd dziennika	Możliwość oglądania planu zajęć na każdy tydzień oraz możliwość poglądu ocen i frekwencji uzyskanych przez ucznia w ciągu wybranego tygodnia
5	Przegląd zadań domowych	Możliwość oglądania wszystkich zadań domowych w ciągu wybranego tygodnia
6	Przegląd sprawdzianów	Możliwość przeglądania wszystkich sprawdzianów na najbliższe cztery tygodnia
7	Przegląd danych o szkole	Możliwość przeglądania informacji o szkole oraz o nauczycielach prowadzących zajęcia u zalogowanego ucznia
8	Przegląd danych osobistych i ich edytowanie	Możliwość przeglądu danych osobistych oraz możliwość ich zmiany

Funkcjonalności nauczyciela:

Tabela 3.3. Opis funkcjonalności nauczycieli

Nº	Funkcja	Opisanie
1	Logowanie	Możliwość zalogować się do serwisu z prawami nauczyciela
2	Wystawienie i pogląd ocen	Możliwość wpisywania i przeglądania ocen dla uczniów wybranej klasy
3	Wystawienie i pogląd frekwencji	Możliwość wpisywania i przeglądania frekwencji dla uczniów wybranej klasy
4	Dodawania sprawdzianów	Możliwość dodawania tematu oraz nazwę sprawdzianów które odbędą się dla wybranej klasy
5	Przegląd planu zajęć	Możliwość oglądania planu zajęć na każdy tydzień
6	Przegląd danych o szkole	Możliwość przeglądania informacji o szkole oraz o uczniach, w których nauczyciel jest wychowawcą
7	Przegląd danych osobistych i ich edytowanie	Możliwość przeglądu danych osobistych oraz możliwość ich zmiany
8	Dodawania tematów zajęcia	Możliwość wpisywania tematu zajęć dla wybranego zajęcia
9	Dodawania zadań domowych	Możliwość wpisywania zadania domowego dla wybranego zajęcia

Jak widać, funkcjonalności ucznia i nauczyciela są ściśle ze sobą powiązane, ponieważ dla tych grup użytkowników potrzebne są te same dane z bazy danych dla korzystania z serwisu. Użytkownik z prawami administratora ma natomiast możliwość zarządzania bardziej globalnymi danymi, takimi jak: szczegółowe informacje o uczniach i nauczycielach, o przedmiotach, o klasach itp.

W poniższej tabeli znajdują się opisane funkcje serwisu, dla użytkownika z prawami administratora:

Tabela 3.4. Opis funkcjonalności administratora

Nº	Funkcja	Opisanie
1	Logowanie	Możliwość zalogowania się do serwisu z prawami administratora
2	Dodawanie, edytowanie i usuwanie uczniów	Możliwość dodawania uczniów do bazy danych oraz możliwość ich edycji oraz możliwość ich usuwania
3	Poszukiwanie uczniów	Możliwość wyszukiwania uczniów za pomocą wybranych filtrów, takich jak: klasa w której znajduje się uczeń, imię lub nazwisko itp.
4	Dodawanie, edytowanie i usuwanie nauczycieli	Możliwość dodawania nauczycieli do bazy danych oraz możliwość ich edycji oraz możliwość ich usuwania
5	Poszukiwanie nauczycieli	Możliwość wyszukiwania nauczycieli za pomocą wybranych filtrów.
6	Dodawania i usuwania przedmiotów	Możliwość dodawania nazwy przedmiotów do bazy danych oraz ich usuwania
7	Dodawania i usuwania zajęć	Możliwość dodawania i usuwania zajęć dla wybranej klasy.
8	Tworzenia planu zajęć	Możliwość tworzenia planu zajęć dla wybranej klasy

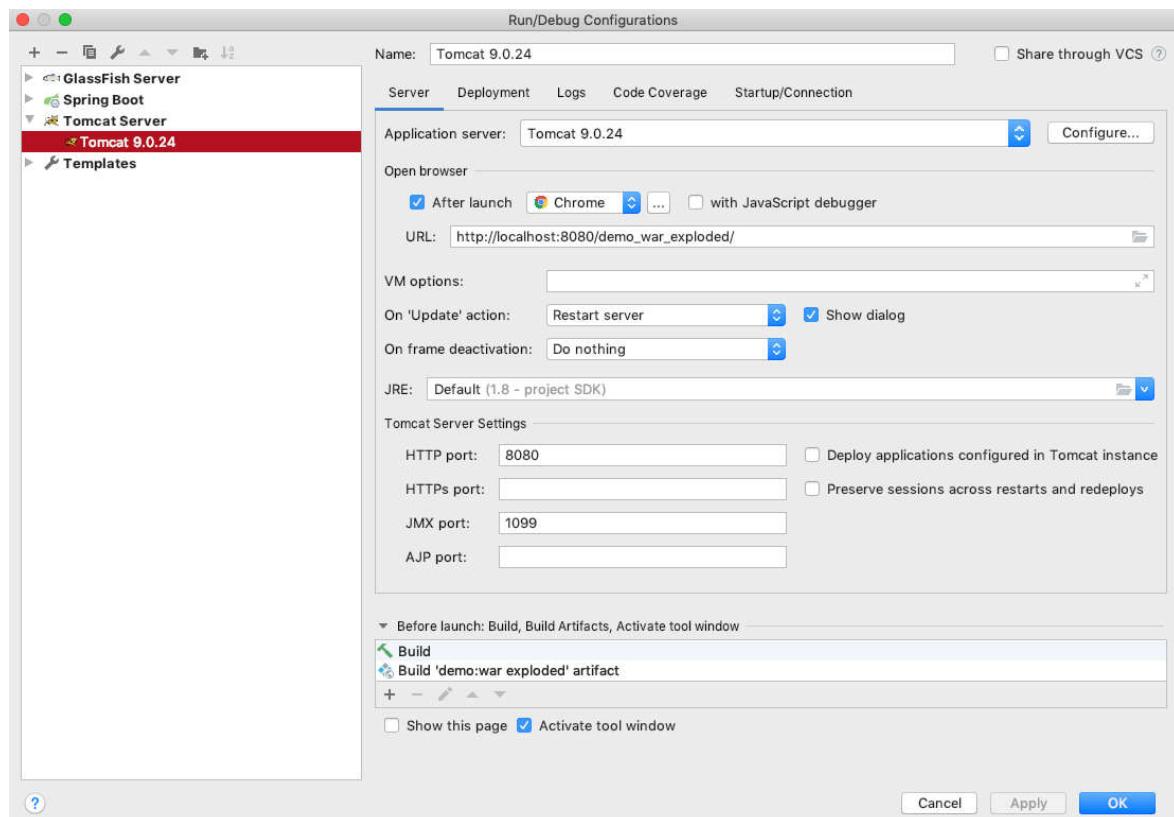
Wszystkie funkcjonalności opisane powyżej są potrzebnymi dla prawidłowego działania serwisu, dlatego muszą być one zrealizowane. Po ich opracowaniu można przejść do następnego etapu tworzenia aplikacji, którym będzie wykonanie danego serwisu.

3.4. Realizacja serwisu

W tym etapie zostały stworzone wszystkie potrzebne klasy Java oraz została ustawiona konfiguracja dla prawidłowego działania i uruchomienia projektu. Ten etap można podzielić na oddzielne części, ponieważ w jednej części została ustawiona konfiguracja, w innej projekt został uruchomiony na serwerze lokalnym, w innej do projektu zostały podłączone zależności z różnymi bibliotekami itp. Dlatego ten etap został podzielony na różne części, które są opisane kroki po kroku w poniższych podpunktach.

3.4.1. Uruchomienie projektu na serwerze lokalnym TomCat

Uruchomienie projektu na serwerze lokalnym TomCat jest potrzebnym dlatego, żeby można było zobaczyć wygląd projektu w przeglądarce oraz sprawdzać poprawność działania tej czy innej stworzonej funkcji, za pomocą przeglądarki. Dla tego zadania najpierw potrzebnym było pobranie archiwum z serwerem z Internetu. Następnie, po rozpakowaniu tego archiwum, potrzebnym było skonfigurować środowisko IntelliJ Idea tak, aby po uruchomieniu projektu on był uruchomiony na potrzebnym serwerze. Z tego powodu, w ustawieniach uruchomienia został wybrany serwer TomCat i ścieżka do katalogu, gdzie on się znajdował. Na rysunku 3.8 dobrze widać wszystkie ustawienia uruchomienia tego projektu.



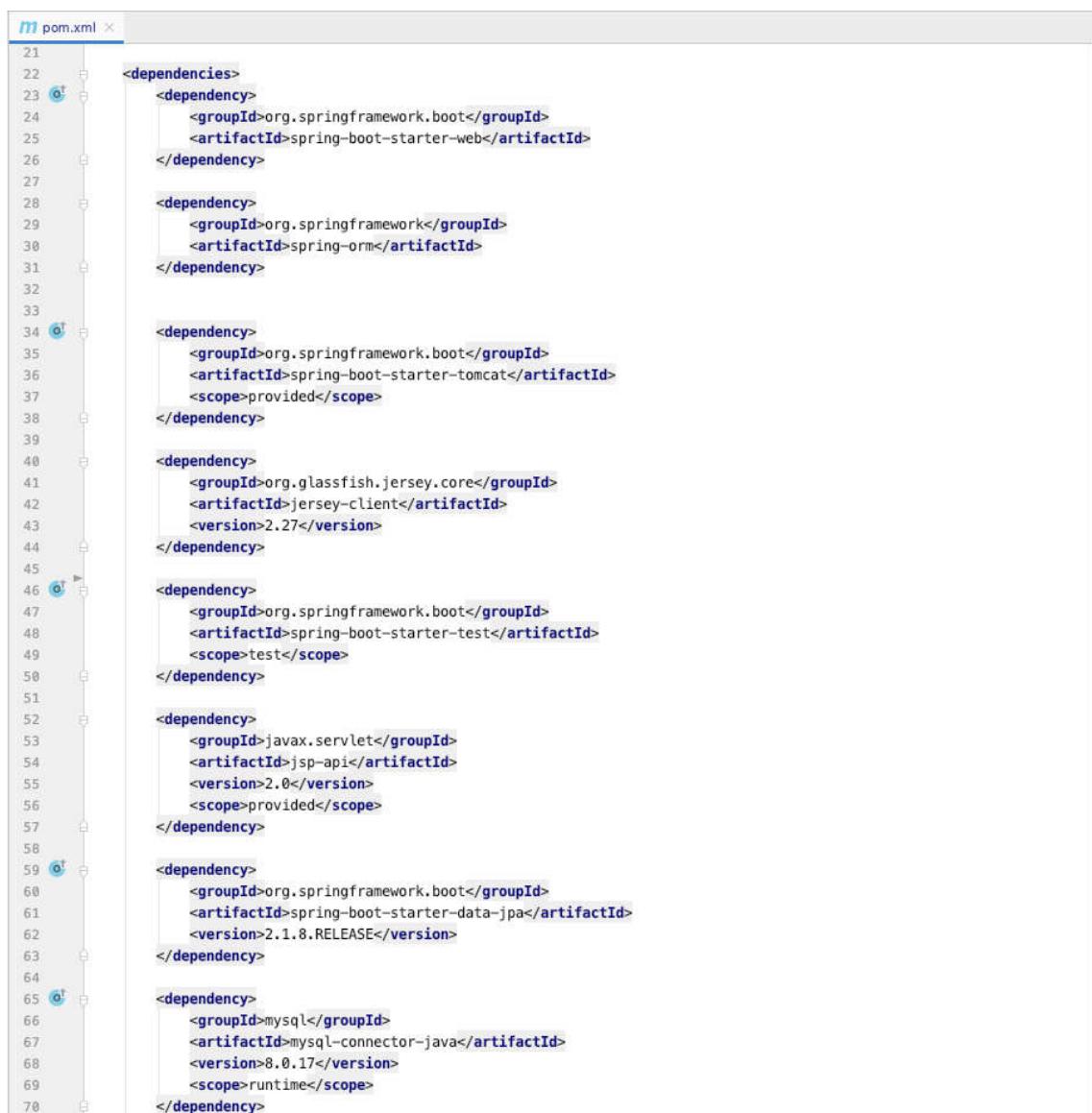
Rys. 3.8. Konfiguracje uruchomienia projektu w IntelliJ Idea

Jak można zobaczyć, w tych ustawieniach została wybrana przeglądarka Google Chrome, także można określić adres URL, na którym będzie widoczny projekt, oraz można było wybrać HTTP port, na którym będzie uruchomiony projekt. Dla konieczności konfiguracji, w ustawieniach w zakładce „Deployment”, potrzebnym było dodać artefakt projektu, który jest upakowany w archiwum „WAR” (Web Application Resource).

3.4.2. Dodawanie zależności Maven

Następnym etapem realizacji takiego serwisu było dodawanie do projektu zależności za pomocą technologii Maven. Jest to potrzebnym dla połączania wszystkich bibliotek, za pomocą których można korzystać funkcje potrzebujących na tą realizację.

Można to zrobić metodą dodawania zależności do pliku „pom.xml”. Pozwala na to tag „<dependecies>”. Na rysunku 3.9 jest to przykładowo pokazane.

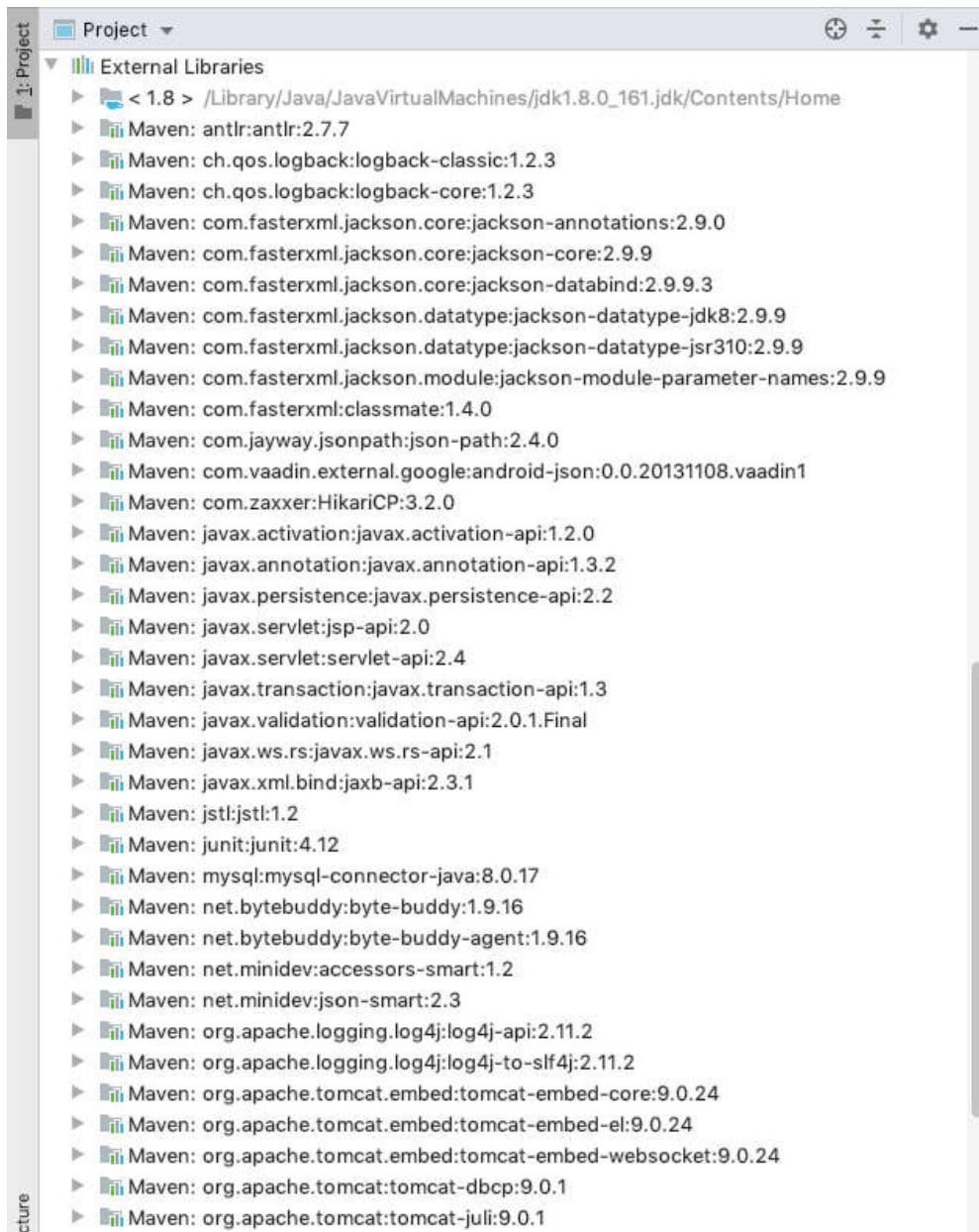


```
1 pom.xml <!--
2
3   <dependencies>
4     <dependency>
5       <groupId>org.springframework.boot</groupId>
6       <artifactId>spring-boot-starter-web</artifactId>
7     </dependency>
8
9     <dependency>
10      <groupId>org.springframework</groupId>
11      <artifactId>spring-orm</artifactId>
12    </dependency>
13
14    <dependency>
15      <groupId>org.springframework.boot</groupId>
16      <artifactId>spring-boot-starter-tomcat</artifactId>
17      <scope>provided</scope>
18    </dependency>
19
20    <dependency>
21      <groupId>org.glassfish.jersey.core</groupId>
22      <artifactId>jersey-client</artifactId>
23      <version>2.27</version>
24    </dependency>
25
26    <dependency>
27      <groupId>org.springframework.boot</groupId>
28      <artifactId>spring-boot-starter-test</artifactId>
29      <scope>test</scope>
30    </dependency>
31
32    <dependency>
33      <groupId>javax.servlet</groupId>
34      <artifactId>jsp-api</artifactId>
35      <version>2.0</version>
36      <scope>provided</scope>
37    </dependency>
38
39    <dependency>
40      <groupId>org.springframework.boot</groupId>
41      <artifactId>spring-boot-starter-data-jpa</artifactId>
42      <version>2.1.8.RELEASE</version>
43    </dependency>
44
45    <dependency>
46      <groupId>mysql</groupId>
47      <artifactId>mysql-connector-java</artifactId>
48      <version>8.0.17</version>
49      <scope>runtime</scope>
50    </dependency>
51
52  </dependencies>
53
54  <build>
55    <plugins>
56      <plugin>
57        <groupId>org.springframework.boot</groupId>
58        <artifactId>spring-boot-maven-plugin</artifactId>
59        <version>2.1.8.RELEASE</version>
60      </plugin>
61    </plugins>
62  </build>
63
64</project>
```

Rys. 3.9. Zawartość pliku „pom.xml”

Jak można zobaczyć na wcześniejszym rysunku, dla dodawania zależności „<dependency>” potrzebnym jest głównie „<groupId>” oraz „<artifactId>”, które można znaleźć w Internecie.

Po importie zmian w projekcie Maven, wszystkie biblioteki zostają podłączone do projektu. Można to sprawdzić i zobaczyć w drzewie projektu, gdzie znajdują się wszystkie podłączone biblioteki. Jest to pokazane na rysunku 3.10.

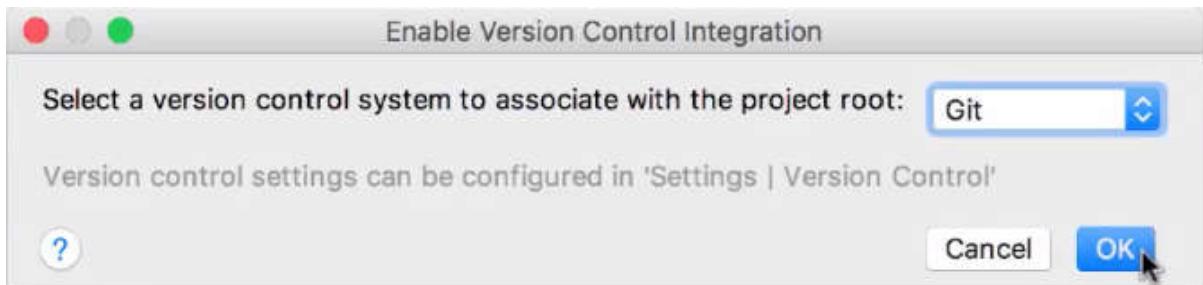


Rys. 3.10. Wygląd wszystkich podłączonych bibliotek w drzewie projektu

3.4.3. Użycie technologii Git i GitHub

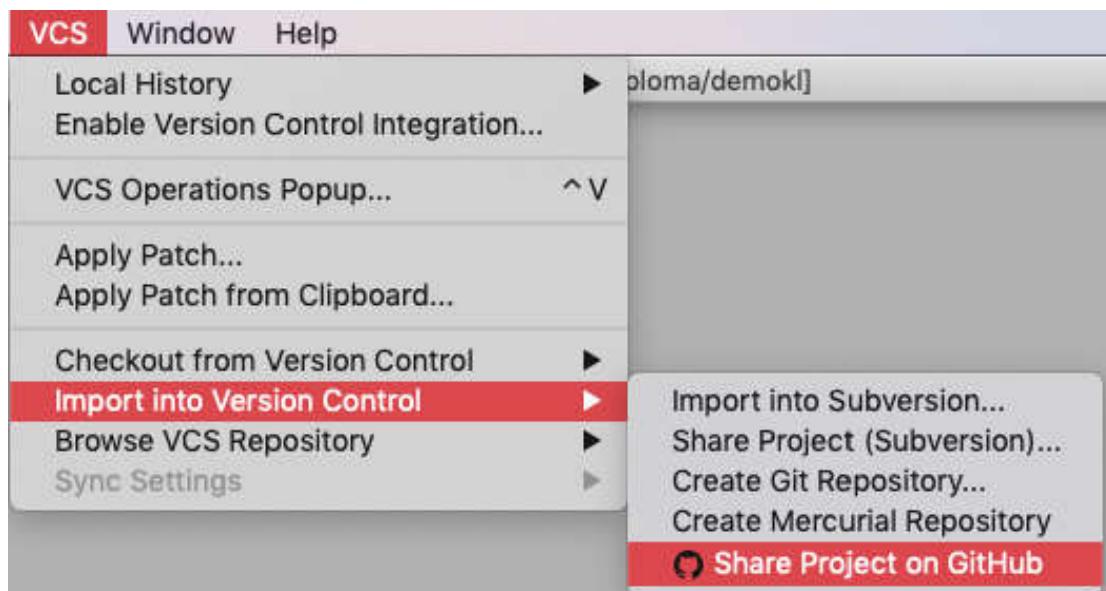
W celu bezpieczeństwa projektu przed zagrożeniem zniknięcia lub od napisania nieprawidłowego kodu, wygodnym jest używanie technologii Git i GitHub. Ta technologia pozwala na robienie kopii bezpieczeństwa bezpośrednio do serwisu internetowego GitHub.

Dla użycia tej technologii potrzebnym jest założenia konta na GitHub. Następnie za pomocą menu „VCS” programu IntelliJ Idea można włączyć wsparcie różnych systemów kontroli wersji, w tym przypadku została wybrany Git (rys. 3.11).



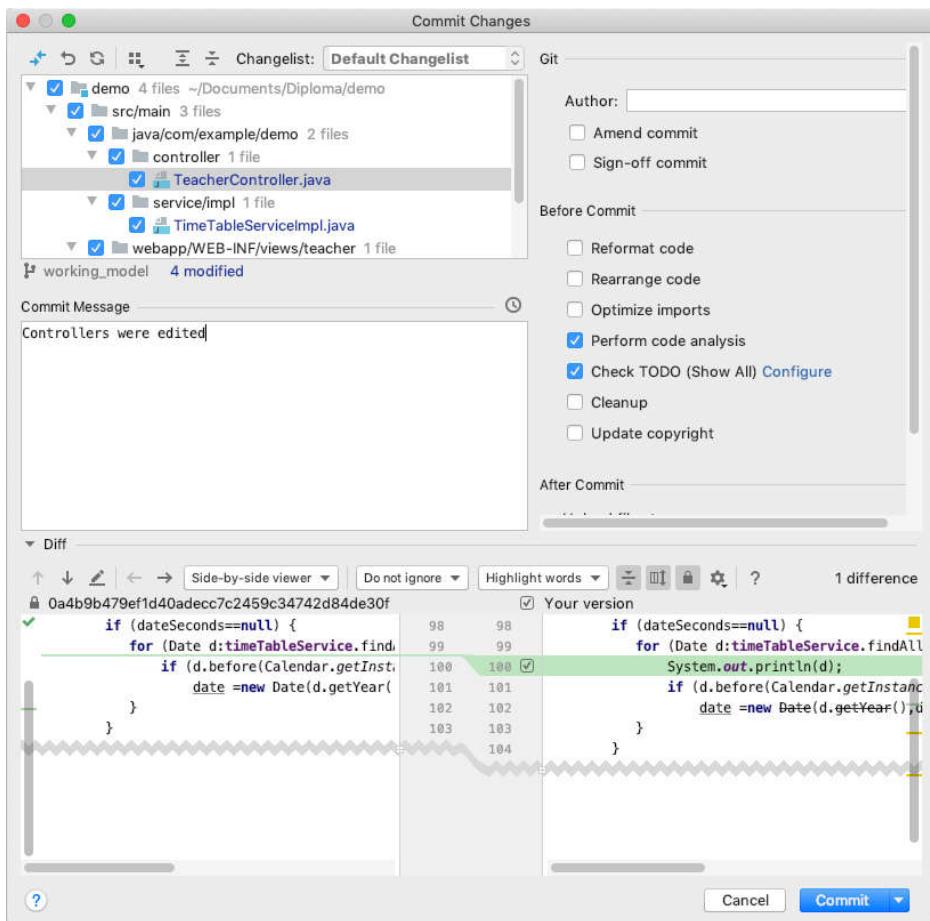
Rys. 3.11. Okno wyboru VCS w programie IntelliJ Idea

Następnie, po zalogowaniu się do swojego konta GitHub przez program IntelliJ Idea, można stworzyć repozytorium swojego projektu i można korzystać z tej technologii (rys. 3.12).



Rys. 3.12. Przykład tworzenia repozytorium w IntelliJ Idea

W tym momencie, gdy zostało utworzone repozytorium, można robić komity wszystkich plików, które zostały zmienione i wysyłać je do serwisu GitHub (rys. 3.13).



Rys. 3.13. Przykład komitu w IntelliJ Idea

Gdy zmiany zostały dodane do obecnej gałęzi, można je wprowadzić do gałęzi mastera, jeśli nie budzą wątpliwości. Można to zrobić bezpośrednio za pomocą przeglądarki, po zalogowaniu do systemu (rys. 5).

The screenshot shows a GitHub repository page for 'NikitaVovk / schoolDiploma'. At the top, there are buttons for 'Code', 'Issues 0', 'Pull requests 0', 'Actions', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. Below this, a message says 'No description, website, or topics provided.' with an 'Edit' button. A 'Manage topics' link is also present. Key statistics are shown: 17 commits, 2 branches, 0 packages, 0 releases, and 1 contributor. A section titled 'Your recently pushed branches:' lists 'working_model' (less than a minute ago) with a 'Compare & pull request' button. Below this, a summary says 'This branch is 7 commits ahead, 3 commits behind master.' with 'Pull request' and 'Compare' buttons. A detailed list of commits shows the following entries:

- Controllers were edited (latest commit cadad54, 3 minutes ago)
- .mvn/wrapper (All, 2 months ago)
- src (Controllers were edited, 3 minutes ago)
- .gitignore (All, 2 months ago)
- mvnw (All, 2 months ago)
- mvnw.cmd (All, 2 months ago)
- pom.xml (Controllers were edited, 3 minutes ago)

Rys. 3.14. Wygląd repozytorium w serwisie GitHub

3.4.4. Ustawienie konfiguracji projektu

Dla prawidłowego działania wszystkich bibliotek oraz wszystkich technologii potrzebnym było ustawić konfiguracje projektu. Te konfiguracje dotyczą różnych aspektów, takich jak: komunikacja z bazą danych, ustawienia Spring oraz Spring Security. Wszystkie główne ustawienia znajdują się w pliku „application.properties”, który został dodany podczas tworzenia projektu. Zawartość tego pliku jest przedstawiona na rys. 3.15.

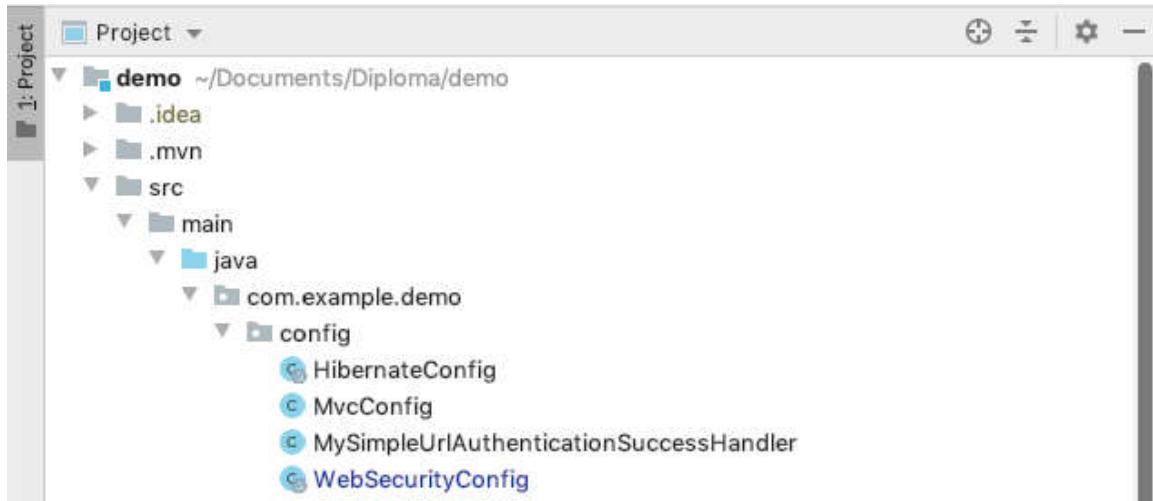


```
application.properties
1 spring.mvc.view.prefix=/WEB-INF/views/
2 spring.mvc.view.suffix=.jsp
3 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
4 spring.jpa.hibernate.ddl-auto=update
5
6 spring.datasource.url=jdbc:mysql://localhost:3306/schoolTest?serverTimezone=UTC
7 spring.datasource.username=root
8 spring.datasource.password=12qw34er
9 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
10 spring.jpa.show-sql=true
```

Rys. 3.15. Zawartość pliku „application.properties”

Jak widać, w pierwszych dwóch liniach znajduje się konfiguracja dla poprawnego umieszczenia plików „.jsp”, które odpowiadają za widok serwisu. W następnych liniach znajduje się konfiguracja dla poprawnej komunikacji z bazą danych.

Ponieważ w tym pliku nie da się ustawić wszystkie konfiguracji potrzebnych dla projektu, to został stworzony Java pakiet z nazwą „config”, który zawiera klasy konfiguracji. Zawartość tego pakietu można zobaczyć na rysunku 3.16.



Rys. 3.16. Wygląd pakietu „config” w drzewie projektu

Jak widać, do pakietu zostały dodane cztery klasy, które mają różne cele i są opisane niżej:

- HibernateConfig – odpowiada za konfigurację biblioteki Hibernate, która służy do komunikacji z bazą danych.
- MvcConfig – odpowiada za konfigurację aplikacji stworzonych na podstawie architektury MVC.
- MySimpleUrlAuthenticationSuccessHandler – jest potrzebna do przeniesienia użytkowników na odpowiednią stronę, w zależności od roli użytkownika.
- WebSecurityConfig – ta klasa odpowiada za bezpieczeństwo całej aplikacji i nie pozwala użytkownikom uzyskać dostęp do danych, do których oni nie mają praw.

Wszystkie te klasy są opisane adnotacją „@Configuration”, która informuje kontener Spring, że jest to klasa konfiguracji, która zawiera definicje i zależności komponentów bean. Na rys. 3.17 przykładowo pokazana została zawartość klasy WebSecurityConfig.

```
26  @Configuration
27  @EnableAutoConfiguration
28  @ComponentScan({ "com.example.demo" })
29  @EnableWebSecurity
30  public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
31
32      @Autowired
33      private UserDetailServiceImpl loginService;
34
35      @Autowired
36      public void configAuthBuilder(AuthenticationManagerBuilder auth) throws Exception {
37          auth.userDetailsService(loginService);
38      }
39
40      @Override
41      protected void configure(HttpSecurity http) throws Exception {
42
43          http
44              .authorizeRequests() ExpressionUrlAuthorizationConfigurer<Http>.ExpressionInterceptUrlRegistry
45              .antMatchers( ...antPatterns: "/" ).permitAll() ExpressionUrlAuthorizationConfigurer<Http>.ExpressionInterceptUrlRegistry
46              .antMatchers( ...antPatterns: "/mainTeacher/**" ).hasRole("TEACHER") ExpressionUrlAuthorizationConfigurer<Http>.ExpressionIn
47              .antMatchers( ...antPatterns: "/mainStudent/**" ).hasRole("STUDENT") ExpressionUrlAuthorizationConfigurer<Http>.ExpressionIn
48              .antMatchers( ...antPatterns: "/mainAdmin/**" ).hasRole("ADMIN") ExpressionUrlAuthorizationConfigurer<Http>.ExpressionIntercep
49              .antMatchers( ...antPatterns: "/resources/css/**" ).permitAll() ExpressionUrlAuthorizationConfigurer<Http>.ExpressionIntercept
50              .antMatchers( ...antPatterns: "/resources/images/**" ).permitAll() ExpressionUrlAuthorizationConfigurer<Http>.ExpressionIntercept
51              .anyRequest().authenticated() ExpressionUrlAuthorizationConfigurer<Http>.ExpressionInterceptUrlRegistry
52
53          .and() HttpSecurity
54
55              .csrf().disable()
56              .formLogin() FormLoginConfigurer<HttpSecurity>
57              .loginPage("/login") FormLoginConfigurer<HttpSecurity>
58              .loginProcessingUrl("/login") FormLoginConfigurer<HttpSecurity>
59              .successHandler(myAuthenticationSuccessHandler()) FormLoginConfigurer<HttpSecurity>
60              .passwordParameter("password") FormLoginConfigurer<HttpSecurity>
61              .usernameParameter("username") FormLoginConfigurer<HttpSecurity>
62              .permitAll() FormLoginConfigurer<HttpSecurity>
63
64          .and() HttpSecurity
65
66
67          .logout() LogoutConfigurer<HttpSecurity>
68          .permitAll();
69      }
70      @Bean
71      public AuthenticationSuccessHandler myAuthenticationSuccessHandler(){
72          return new MySimpleUrlAuthenticationSuccessHandler();
73      }
74
75      @Bean
76      public PasswordEncoder passwordEncoder() { return NoOpPasswordEncoder.getInstance(); }

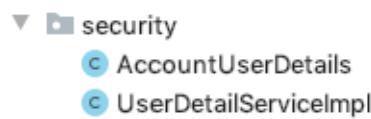
```

Rys. 3.17. Zawartość pliku WebSecurityConfig

Jak można zobaczyć, główną metodą tej klasy jest metoda o nazwie „configure”, która jest przedefiniowana i odpowiada za wszystkie konfiguracje bezpieczeństwa serwisu.

3.4.5. Bezpieczeństwo serwisu

Głównym celem bezpieczeństwa tego serwisu jest uniemożliwić dostęp do żądań HTTP użytkownikom, którzy nie mają odpowiednich praw. Dlatego został stworzony pakiet Java z nazwą „security”, który zawiera dwie klasy (rys. 3.18).



Rys. 3.18. Zawartość pakietu „security”

Te klasy mają podobne cele i obie są implementacjami interfejsów z biblioteki Spring. Te klasy dostarczają podstawowych informacji o użytkownikach. Implementacje nie są używane bezpośrednio przez Spring Security do celów bezpieczeństwa. Po prostu przechowują informacje o użytkowniku, które są następnie umieszczane w obiektach uwierzytelniania. Umożliwia to przechowywanie informacji o użytkownikach niezwiązanych z bezpieczeństwem (takich jak adresy e-mail, numery telefonów itp.).

Obiekty tych klas są wykorzystywane w konfiguracjach projektu oraz kontrolerach. Klasę „AccountUserDetails” można używać do zwrócenia obiektu użytkownika, który w tej chwili ma rozpoczętą sesję.

3.4.6. Komunikacja z bazą danych

Komunikacja z bazą danych odbywa się za pomocą obiektu zapytań „Querry”. Ponieważ zostały stworzone klasy Java, które są jednocześnie encjami bazy danych, to jest możliwość wykorzystać obiekty tych klas, w celu zwrócenia odpowiedzi na zapytania SQL. Dla realizacji takiej komunikacji zostaw stworzony pakiet Java z nazwą „dao”, a jego zawartość jest przedstawiona na rysunku 3.19.



Rys. 3.19. Zawartość pakietu „dao”

W tym pakiecie znajdują się interfejsy Java oraz pakiet „impl”, który zawiera ich implementację. Także w tym pakiecie znajduje się abstrakcyjna klasa „AbstractDao”, która zawiera pole prywatne obiektu „SessionFactory”, co pozwala na pracę z bazą danych. Zawartość tej klasy jest przykładowo pokazana na rysunku 3.20.

```
1 package com.example.demo.dao;
2
3 import ...
4
5 @Transactional
6 public abstract class AbstractDao<PK extends Serializable, T> {
7     private final Class<T> persistentClass;
8
9     @Autowired
10    private SessionFactory sessionFactory;
11
12    /unchecked/
13    public AbstractDao() {
14        this.persistentClass = (Class<T>) ((ParameterizedType) this.getClass().getGenericSuperclass()).
15            getActualTypeArguments()[1];
16    }
17
18
19    protected Session getSession() { return sessionFactory.getCurrentSession(); }
20
21    protected T getByKey(PK key) {
22        return getSession().get(persistentClass, key);
23    }
24
25
26    protected void persist(T entity) { getSession().persist(entity); }
27
28    protected void merge(T entity) { getSession().merge(entity); }
29
30    protected void update(T entity) { getSession().update(entity); }
31
32    protected void delete(T entity) { getSession().delete(entity); }
33
34    protected Query createQuery(String hql) { return getSession().createQuery(hql); }
```

Rys. 3.20. Zawartość abstrakcyjnej klasy „AbstractDao”

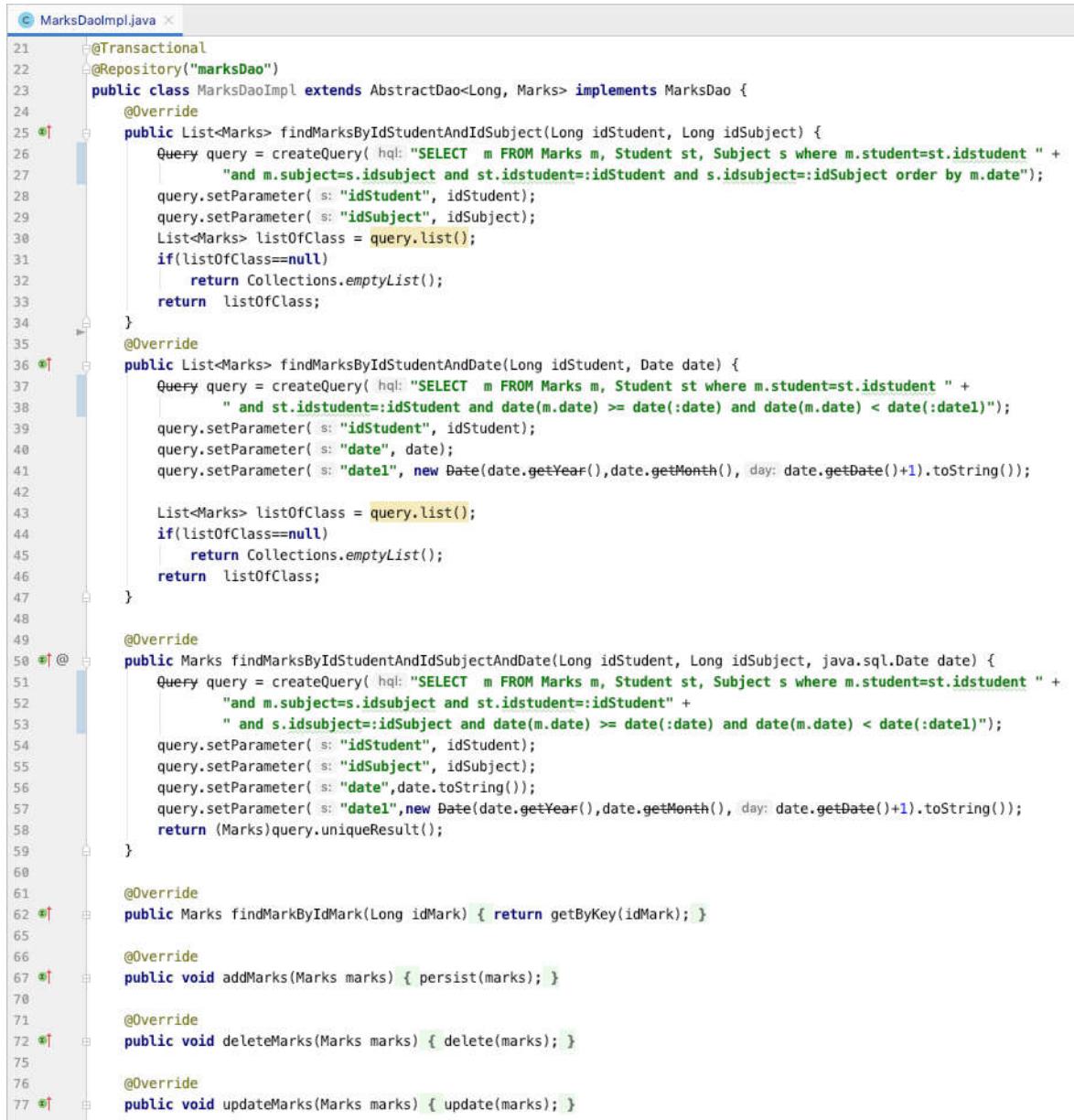
Jak można zobaczyć, ta klasa jest oznaczona adnotacją „@Transactional”, co oznacza, że Spring dynamicznie tworzy proxy dla tej klasy. Także ta klasa zawiera typy generyczne, co pozwala jej znaleźć obiekt dowolnego typu, za pomocą klucza, który jest implementacją interfejsu „Serializable”. Ta klasa została stworzona po to, żeby można było łatwo skorzystać z podstawowych zapytań do bazy danych, takich jak: znaleźć rekord za pomocą klucza podstawowego, dodać rekord do tabeli, usunąć rekord itp.

Interfejsy z pakietu „dao” są natomiast stworzone po to, żeby opisać wszystkie potrzebne funkcje i operacje, które będą wykonywane z bazą danych. Na rysunku 3.21 przykładowo pokazana jest zawartość interfejsu „MarksDao”.

```
1 package com.example.demo.dao;
2
3 import ...
4
5 public interface MarksDao {
6     List<Marks> findMarksByIdStudentAndIdSubject(Long idStudent, Long idSubject);
7     List<Marks> findMarksByIdStudentAndDate(Long idStudent, Date date);
8     Marks findMarksByIdStudentAndIdSubjectAndDate(Long idStudent, Long idSubject, java.sql.Date date);
9     Marks findMarkByMarkId(Long idMark);
10    void addMarks(Marks marks);
11    void deleteMarks(Marks marks);
12    void updateMarks(Marks marks);
13}
```

Rys. 3.21. Zawartość interfejsu „MarksDao”

Patrząc na te funkcje można zobaczyć, że one muszą zwracać jeden obiekt typu „Marks” lub listę tych obiektów, które są znalezione za pomocą kluczy podstawowych lub obcych. Implementacja tych interfejsów przedstawia realizację tych funkcji i jest przykładowo pokazana na rysunku 3.22.



```

1  package pl.edu.wszib;
2
3  import org.springframework.stereotype.Repository;
4  import org.springframework.transaction.annotation.Transactional;
5  import pl.edu.wszib.model.Marks;
6  import pl.edu.wszib.model.Student;
7  import pl.edu.wszib.model.Subject;
8
9  import javax.persistence.EntityManager;
10 import javax.persistence.PersistenceContext;
11 import javax.persistence.Query;
12 import java.util.List;
13
14 /**
15  * Implementation of MarksDao interface
16  */
17 @Repository("marksDao")
18 @Transactional
19 public class MarksDaoImpl extends AbstractDao<Long, Marks> implements MarksDao {
20
21     @Override
22     public List<Marks> findMarksByIdStudentAndIdSubject(Long idStudent, Long idSubject) {
23         Query query = createQuery( hql: "SELECT m FROM Marks m, Student st, Subject s where m.student=st.idstudent " +
24             "and m.subject=s.idsubject and st.idstudent=:idStudent and s.idsubject=:idSubject order by m.date");
25         query.setParameter( s: "idStudent", idStudent);
26         query.setParameter( s: "idSubject", idSubject);
27         List<Marks> listOfClass = query.list();
28         if(listOfClass==null)
29             return Collections.emptyList();
30         return listOfClass;
31     }
32
33     @Override
34     public List<Marks> findMarksByIdStudentAndDate(Long idStudent, Date date) {
35         Query query = createQuery( hql: "SELECT m FROM Marks m, Student st where m.student=st.idstudent " +
36             " and st.idstudent=:idStudent and date(m.date) >= date(:date) and date(m.date) < date(:date1)");
37         query.setParameter( s: "idStudent", idStudent);
38         query.setParameter( s: "date", date);
39         query.setParameter( s: "date1", new Date(date.getYear(),date.getMonth(), day: date.getDate()+1).toString());
40
41         List<Marks> listOfClass = query.list();
42         if(listOfClass==null)
43             return Collections.emptyList();
44         return listOfClass;
45     }
46
47     @Override
48     public Marks findMarksByIdStudentAndIdSubjectAndDate(Long idStudent, Long idSubject, java.sql.Date date) {
49         Query query = createQuery( hql: "SELECT m FROM Marks m, Student st, Subject s where m.student=st.idstudent " +
50             "and m.subject=s.idsubject and st.idstudent=:idStudent " +
51             " and s.idsubject=:idSubject and date(m.date) >= date(:date) and date(m.date) < date(:date1)");
52         query.setParameter( s: "idStudent", idStudent);
53         query.setParameter( s: "idSubject", idSubject);
54         query.setParameter( s: "date", date.toString());
55         query.setParameter( s: "date1", new Date(date.getYear(),date.getMonth(), day: date.getDate()+1).toString());
56         return (Marks)query.uniqueResult();
57     }
58
59     @Override
60     public Marks findMarkByIdMark(Long idMark) { return getByKey(idMark); }
61
62     @Override
63     public void addMarks(Marks marks) { persist(marks); }
64
65     @Override
66     public void deleteMarks(Marks marks) { delete(marks); }
67
68     @Override
69     public void updateMarks(Marks marks) { update(marks); }
70
71 }

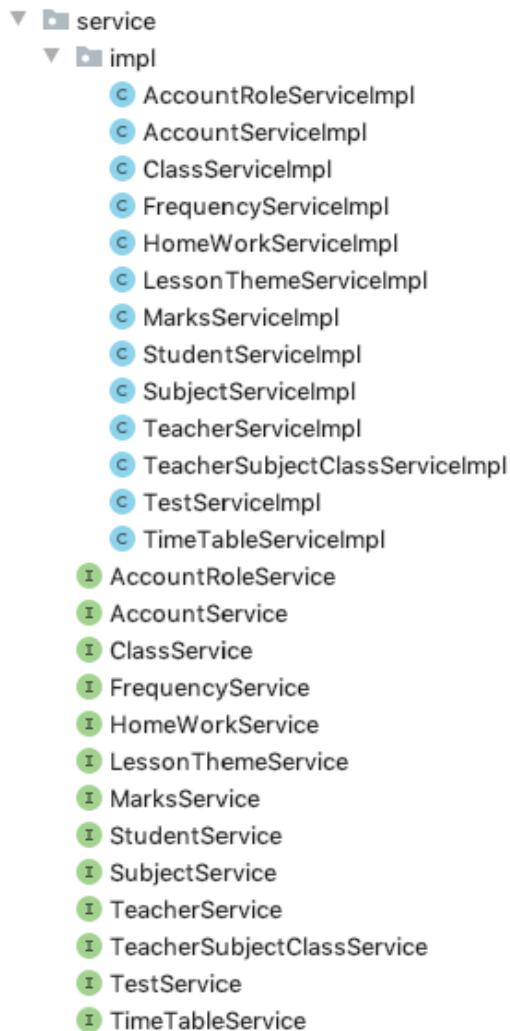
```

Rys. 3.22. Zawartość klasy „MarksDaoImpl”, która jest implementacją interfejsu „MarksDao”

Jak widać, ta klasa jest implementacją interfejsu „MarksDao” oraz ona dziedziczy z abstrakcyjnej klasy „AbstractDao”. Metody tej klasy tworzą zapytania HQL, za pomocą obiektu „Querry” i metody „createQuerry”. Parametry, które potrzeba wstawić do zapytania są oznaczone dwukropkiem, i później są wstawione do zapytania za pomocą metody „setParameter”. Po tych krokach jest wykonywane zapytania do bazy danych za pomocą metod „list” lub „uniqueResult”. Te metody stosują się w zależności od tego, czy zapytanie musi zwrócić pojedynczy rekord z tabeli czy listę rekordów.

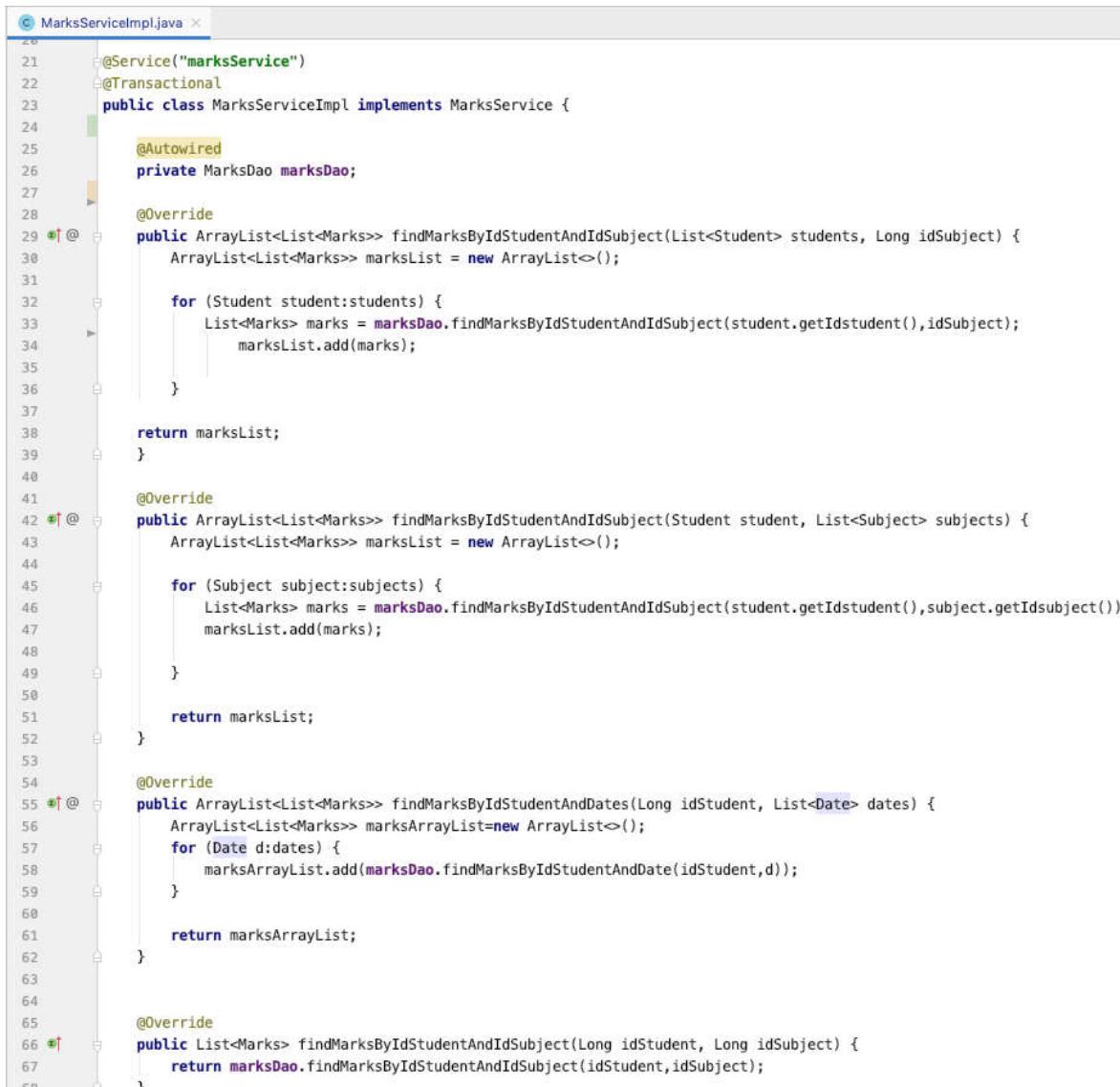
3.4.7. Tworzenie logiki biznesowej serwisu

Tworzenie biznes logiki serwisu polega na przedstawieniu danych w potrzebnym wyglądzie. To oznacza że dane zwrócone z bazy danych muszą być jakoś zmienione lub posortowane, czyli muszą one dostarczyć dane do kontrolerów w potrzebnym wyglądzie. Dla tego zadania został stworzony pakiet Java z nazwą „service”. Ten pakiet zawiera interfejsy oraz pakiet z ich implementacją, i zawartość tych pakietów jest przedstawiona na rysunku 3.23.



Rys. 3.23. Zawartość pakietu „service”

Jak było wcześniej, interfejsy zawierają tylko metody potrzebne do realizacji, a klasy są ich implementacjami i realizują te metody. Na rysunku 3.24 przykładowo przedstawiona zawartość klasy „MarksServiceImpl”.



```
20
21     @Service("marksService")
22     @Transactional
23     public class MarksServiceImpl implements MarksService {
24
25         @Autowired
26         private MarksDao marksDao;
27
28         @Override
29         public ArrayList<List<Marks>> findMarksByIdStudentAndIdSubject(List<Student> students, Long idSubject) {
30             ArrayList<List<Marks>> marksList = new ArrayList<>();
31
32             for (Student student:students) {
33                 List<Marks> marks = marksDao.findMarksByIdStudentAndIdSubject(student.getIdstudent(),idSubject);
34                 marksList.add(marks);
35             }
36
37             return marksList;
38         }
39
40         @Override
41         public ArrayList<List<Marks>> findMarksByIdStudentAndIdSubject(Student student, List<Subject> subjects) {
42             ArrayList<List<Marks>> marksList = new ArrayList<>();
43
44             for (Subject subject:subjects) {
45                 List<Marks> marks = marksDao.findMarksByIdStudentAndIdSubject(student.getIdstudent(),subject.getIdsubject());
46                 marksList.add(marks);
47             }
48
49             return marksList;
50         }
51
52         @Override
53         public ArrayList<List<Marks>> findMarksByIdStudentAndDates(Long idStudent, List<Date> dates) {
54             ArrayList<List<Marks>> marksArrayList=new ArrayList<>();
55             for (Date d:dates) {
56                 marksArrayList.add(marksDao.findMarksByIdStudentAndDate(idStudent,d));
57             }
58
59             return marksArrayList;
60         }
61
62         @Override
63         public List<Marks> findMarksByIdStudentAndIdSubject(Long idStudent, Long idSubject) {
64             return marksDao.findMarksByIdStudentAndIdSubject(idStudent,idSubject);
65         }
66     }
67 }
```

Rys. 3.24. Zawartość klasy „MarksServiceImpl”, która jest implementacją interfejsu „MarksService”

Na tym przykładzie widać, że ta klasa implementuje interfejs „MarksService”. Także ta klasa zawiera pole prywatne obiektu typu „MarksDao”, za pomocą którego można wykorzystywać metody do pobrania danych z bazy danych. Jak można zobaczyć, nie wszystkie dane potrzebują zmian i są dostarczone do kontrolerów bezpośrednio za pomocą pola „marksDao”. Natomiast metoda „findMarksByIdStudentAndIdSubject” tworzy listę z listy ocen zwróconej za pomocą pola „marksDao”, aby przedstawić oceny wszystkich uczniów z wybranego przedmiotu, ID którego jest parametrem tej funkcji.

3.4.8. Tworzenia kontrolerów

Tworzenia kontrolerów polega na tworzeniu zwykłych klas Java z prywatnymi polami oraz metodami, które przetwarzają żądanie HTTP klienta. Najważniejszą adnotacją do stworzenia kontrolera jest „@Controller”, która wskazuje, że dana klasa jest

kontrolerem, i pozwala na użycie takiej adnotacji jak „@RequestMapping”, za pomocą której można przetwarzać GET i POST żądania klienta. Te adnotacje grają ważną rolę w kontrolerach, ponieważ DispatcherServlet wysyła do klas z tymi adnotacjami żądanie na wykonania określonych funkcji.

Dla realizacji tego zadania został stworzony pakiet Java z nazwą „controllers”, który zawiera cztery klasy. Na rysunku 3.25 pokazany wygląd pakietu w drzewie projektu.



Rys. 3.25. Zawartość pakietu „controller”

Jak można zobaczyć każda grupa użytkowników ma osobny kontroler, który wykonuje określone funkcję. Każdy kontroler ma osobne konfiguracje oraz metody. Zawartość jednego z nich jest przedstawiona niżej.

```
StudentController.java
19 19 @Controller
20 20 @RequestMapping("/mainStudent")
21 21 @SessionAttributes(value = "studentJSP")
22 22 public class StudentController {
23
24     @Autowired
25     private StudentService studentService;
26     @Autowired
27     private SubjectService subjectService;
28     @Autowired
29     private MarksService marksService;
30     @Autowired
31     private TimeTableService tableService;
32     @Autowired
33     private TeacherSubjectClassService tsc;
34     @Autowired
35     private FrequencyService frequencyService;
36     @Autowired
37     private HomeworkService homeWorkService;
38     @Autowired
39     private TestService testService;
34     @Autowired
35     private AccountService accountService;
36
37     @GetMapping
38     public ModelAndView studentMainPage(@AuthenticationPrincipal AccountUserDetails account, Map<String, Object> map) {
39         map.put("student", studentService.findStudentByIdAccount(account.getId()));
40         ModelAndView modelAndView = new ModelAndView();
41         modelAndView.setViewName("student/studentPage");
42         modelAndView.addObject("attributeName: studentJSP", studentService.findStudentByIdAccount(account.getId()));
43
44         return modelAndView;
45     }
46
47     @GetMapping("/")
48     public String studentDayBook() {
49
50         return "student/studentPage";
51     }
52
53     @GetMapping("/marks")
54     public String studentMarks(@AuthenticationPrincipal AccountUserDetails account, Map<String, Object> map) {
55         map.put("subjects", subjectService.findSubjectsByIdClass(studentService.findStudentByIdAccount(account.getId())
56             .getgetClass().getIdclass()));
57         map.put("marksSubject", marksService.findMarksByIdStudentAndIdSubject(studentService
58             .findStudentByIdAccount(account.getId()),
59             subjectService.findSubjectsByIdClass(studentService.findStudentByIdAccount(account
60                 .getId()).getgetClass().getIdclass())));
61         return "student/studentMarks";
62     }
63
64 }
```

Rys. 3.26. Zawartość klasy „StudentController”, która jest kontrolerem

Adnotacja „@RequestMapping”, która znajduje się przed definicją klasy, pozwala na przetwarzanie żądań, które zaczynają się z „/mainStudent”. Także za pomocą adnotacji „@SessionAttributes”, została stworzona zmienna, która będzie zapisywać się w sesji. W tym przypadku, to pozwala na zapisania obiektu ucznia do sesji, który zalogował się.

Następnym krokiem do stworzenia kontrolera, była deklaracja pól prywatnych, które są obiektami serwisu i są zadeklarowane za pomocą adnotacji „@Autowired”, który automatycznie podstawia obiekt do egzemplarza. Za pomocą tych obiektów można pobierać dane z BD oraz zmieniać ich.

Głavnym krokiem tworzenia kontrolera jest tworzenia metod, pozwalających na przetwarzanie żądań HTTP. Jak widać wszystkie te metody są opisane adnotacjami „@GetMapping” lub „@PostMapping”, co pozwala na przetwarzania metod Get i Post. W nawiasach obok tych adnotacji, jako atrybut, znajdują się nazwy tych żądań. Większość metod zwracają obiekty typu String, w których znajdują się ścieżki do plików przedstawiających widok serwisu (do plików z rozszerzeniem JSP). Te metody mogą mieć dowolne parametry, w przypadku gdy żądania zawiera parametry, to potrzebnym jest dodać ich do deklaracji metody, w takiej kolejności jak znajdują się oni w zapytaniu. Na rysunku 3.27 przykładowo pokazano, jak wygląda to w kodzie.



```
170  @
171  @RequestMapping(value = "/editData", method = RequestMethod.POST)
172  public String editData(@AuthenticationPrincipal AccountUserDetails account,
173                         @RequestParam(value="address")String address,
174                         @RequestParam(value="email")String email,
175                         @RequestParam(value="phone")String phone) {
176     Student student = studentService.findStudentByIdAccount(account.getId());
177     student.setAddress(address);
178     student.setPhone(phone);
179     student.setEmail(email);
180     studentService.editStudent(student);
181 }
```

Rys. 3.27. Metoda kontrolera, która przetwarza żądania klienta

Ta metoda przetwarza żądania metody POST, która ma odpowiednie parametry, i te parametry zostały dodane do deklaracji metody, w potrzebnej kolejności, za pomocą adnotacji „@RequestParam”. Ta adnotacja opisuje konkretny argument, jako parametr z żądania, i jest on automatycznie wypełniony potrzebnym obiektem. Jak widać, ta metoda nie zwraca ścieżkę do plików z widokiem, a przekierowuje do innej metody, która obsługuje to żądania. Dana metoda służy do edycji danych użytkownika, dlatego obsługuje metodę POST. W ciele tej funkcji jest szukany obiekt ucznia, za pomocą identyfikatora konta, i później do tego obiektu są wstawione nowe dane, które są argumentami tej funkcji. Następnie wykonuje się metoda „editStudent”, która zmienia dane w BD.

Na poniższym przykładzie przedstawiono funkcję która zwraca nazwę widoku i wstawia określone dane do niego.

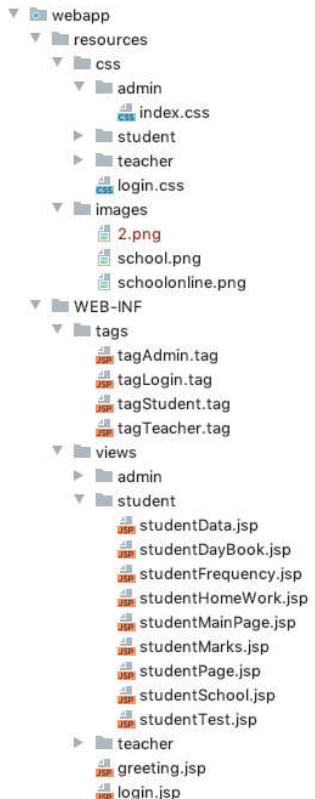
```
59 @GetMapping("/marks")
60 public String studentMarks(@AuthenticationPrincipal AccountUserDetails account, Map<String, Object> map){
61     map.put("subjects", subjectService.findSubjectsByIdClass(studentService.findStudentByIdAccount(account.getId())
62             .getgetClass().getIdClass()));
63     map.put("marksSubject", marksService.findMarksByIdStudentAndIdSubject(studentService
64             .findStudentByIdAccount(account.getId()),
65             subjectService.findSubjectsByIdClass(studentService.findStudentByIdAccount(account
66                 .getId()).getgetClass().getIdClass())));
67     return "student/studentMarks";
68 }
69 }
```

Rys. 3.28. Metoda kontrolera, zwracająca nazwę pliku widoku

Ta metoda zawiera dwa argumenty: pierwszym jest obiekt, który zawiera dane konta, drugim argumentem jest mapa obiektów. Za pomocą tej mapy można dodawać różne obiekty do listy z odpowiednim kluczem, za pomocą którego można później te dane pobrać w plikach JSP.

3.4.9. Tworzenie widoku serwisu

Tworzenia widoku serwisu polegało na tworzeniu plików JSP, które generują stronę HTML, oraz podłączeniu do nich plików CSS. Dla tego zadania do projektu został dodany katalog z nazwą „webapp”, w którym znajdują się tylko pliki przeznaczone do widoku serwisu. Na poniższym rysunku znajduje się wygląd tego katalogu w drzewie projektu.



Rys. 3.29. Hierarchia plików i katalogów, odpowiadających za wygląd serwisu

Jak widać, w katalogu „WEB-INF” znajdują się wyłącznie pliki, które wykorzystują technologie JSP. W katalogu „tags” znajdują się wyłącznie pliki z rozszerzeniem „.tag”. Takie rozszerzenie oznacza że dany plik jest układem. Takie układy pozwalają tworzyć szablony stron (layout), które składają się z trzech głównych części. Pierwszą częścią jest nagłówek, drugą – ciało, trzecią – stopka. Na rysunku 3.30 jest przedstawiona zawartość pliku „tagStudent.tag”, który jest układem dla stron ucznia.

```

1 <%@tag description="Overall Page template" pageEncoding="UTF-8"%>
2 <%@attribute name="header" fragment="true" %>
3 <%@attribute name="footer" fragment="true" %>
4 <html>
5 <body>
6 <header>
7   <div id="pageHeader">
8     <div class="container" id="container-header">
9       <div class="png"></div>
10      <div class="greeting">
11        <div class="png"></div>
12      </div>
13    <jsp:invoke fragment="header"/>
14
15
16    <div class="rightNav">
17      <div class="userLogged">
18        <p>${studentJSP.name} ${studentJSP.surname}</p>
19      </div>
20      <div class="logOut">
21        <form action="${pageContext.request.contextPath}/logout" method="post">
22          <input type="submit" value="Sign Out" />
23        </form>
24      </div>
25    </div>
26
27    <div class="container" id="container-header2">
28      <div id="witryna">
29        <p><b>Witryna Ucznia</b></p>
30      </div>
31    </div>
32
33  </header>
34
35  <main id="content-body">
36    <div class="container" id="container-body">
37
38      <div class="navLinks">
39        <ul>
40          <li>
41            <a href="${pageContext.request.contextPath}/mainStudent/marks" class="links">0ceny</a>
42          </li>
43          <li>
44            <a href="${pageContext.request.contextPath}/mainStudent/frequency" class="links">Frekwencja</a>
45          </li>
46          <li>
47            <a href="${pageContext.request.contextPath}/mainStudent/dayBook" class="links">Dziennik</a>
48          </li>
49          <li>
50            <a href="${pageContext.request.contextPath}/mainStudent/homeWork" class="links">Zadanie domowe</a>
51          </li>
52          <li>
53            <a href="${pageContext.request.contextPath}/mainStudent/test" class="links">Sprawdziany</a>

```

Rys. 3.30. Zawartość pliku „tagStudent.tag”, który jest szablonem dla stron ucznia

Jak widać, ten plik jest podzielony na trzy części za pomocą atrybutów: „header”, „main” i „footer”. Do każdej części są dodane różne fragmenty, które będą widoczne na każdej stronie, która korzysta z tego szablonu. Te fragmenty muszą być jednakowymi dla wszystkich stron które implementują dany układ.

W katalogu „views” znajdują się pliki z rozszerzeniem „.jsp”, które generują widok serwisu i korzystają z odpowiednich szablonów. Na rysunku 3.31 przykładowo pokazana została zawartość pliku „studentMarks.jsp”, w celu przedstawienia sposobu tej realizacji.

```

8  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
9  <%@ taglib prefix="c"
10 <%@ uri="http://java.sun.com/jsp/jstl/core" %>
11 <%@taglib prefix="t" tagdir="/WEB-INF/tags" %>
12 <html>
13 <head>
14   <title>Title</title>
15   <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/student/index.css" />
16 </head>
17 <body>
18 <t:tagStudent>
19   <jsp:attribute name="header">
20   </jsp:attribute>
21   <jsp:attribute name="footer">
22   </jsp:attribute>
23   <jsp:body>
24     <div id = "tabName">
25       <p>Oceny</p>
26     </div>
27     <div class="hr">
28       <hr>
29     </div>
30     <div id="tabNameInfo">
31       <p>Twoje oceny uzyskane w tym semestrze:</p>
32     </div>
33     <table width="100%" border="2px solid blue">
34       <tr>
35         <th style="...">Przedmiot</th>
36         <th style="...">Oceny</th>
37         <th>Ocena śródroczna </th>
38       </tr>
39       <c:forEach var="subjectItem" items="${subjects}" >
40         <tr>
41           <td>${subjectItem.name}</td>
42
43           <td>
44             <c:forEach var="marks" items="${marksSubject}">
45               <c:forEach var="mark" items="${marks}">
46                 <c:if test="${mark.subject.idssubject==subjectItem.idssubject}">
47                   ${mark.mark}
48                 </c:if>
49               </c:forEach>
50             </td>
51           </tr>
52         </c:forEach>
53       </table>
54     </jsp:body>
55   </t:tagStudent>
56 </body>
57 </html>

```

Rys. 3.31. Zawartość pliku „studentMarks.jsp”, który realizuje układ „tagStudent.tag”

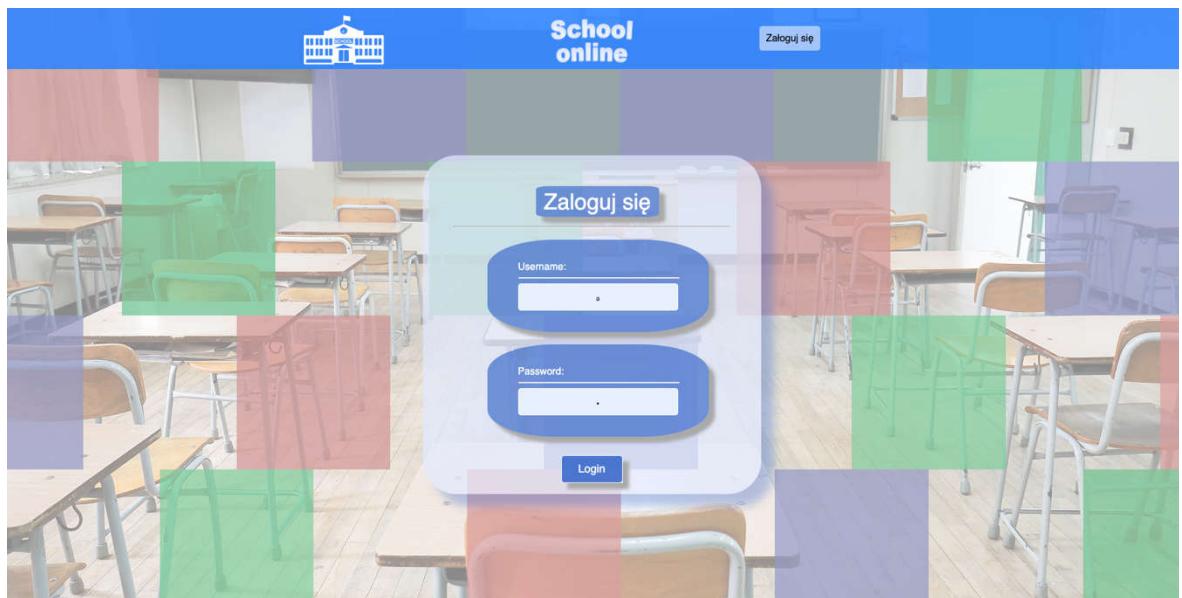
Jak można zobaczyć, w 9 linijce pliku jest podłączona biblioteka „jstl”, funkcjami której można za pomocą użycia prefiksu „c”. W 11 linijce jest podana ścieżka do katalogu z szablonami, z których można korzystać. W 15 linijce jest podłączony plik stylów CSS. W 18 linijce jest otwierany szablon z nazwą „tagStudent”, do którego niżej są dodawane różne fragmenty. W tym przypadku do ciała strony jest dodana różna informacja oraz tablica z ocenami, które uzyskał uczeń. Te oceny znajdują się w liście, która wcześniej została dodana do mapy w kontrolerze. Zawartość tej listy można uzyskać za pomocą klucza, który też był dodany do mapy, oraz używając pętli „foreach”. Po uruchomieniu projektu - z tego pliku jest generowana zwykła strona HTML.

4. Dokumentacja możliwości zaimplementowanego serwisu

W tym rozdziale zostaną przedstawione na rysunkach zaimplementowane funkcjonalności zrealizowanego serwisu do zarządzania szkołą. Wszystkie opcje, które zostały opisane we wcześniejszym rozdziale, zostały zaimplementowane i działają poprawnie, dla każdego użytkownika. Ten rozdział będzie podzielony na trzy części, które będą przedstawiać funkcjonalność od strony administratora, nauczyciela oraz ucznia.

4.1. Widok serwisu od strony administratora

Na rysunku 4.1 przedstawiono stronę logowania, do której ma dostęp każdy użytkownik. Podając login i hasło użytkownik może zalogować się do serwisu.



Rys. 4.1. Strona logowania do serwisu

Po zalogowaniu na konto administratora, użytkownik może zobaczyć wszystkich uczniów, którzy znajdują się w szkole, po naciśnięciu zakładki „Ucznie”, która znajduje się w menu (rys. 4.2). Także jest możliwość wyszukiwania ucznia, podając jego nazwisko lub imię albo klasę, w której on znajduje się.

Nr	ID	Nazwisko Imię	Data urodzenia	Klasa
1	1	Bohdan Artem	2000-01-15	7a
2	94	Bojczuk Vitalia	2000-02-17	2 EF-DI
3	11	Demianets Iwan	2003-06-12	7a
4	8	Dydynski Volodymyr	2001-02-15	7b
5	9	Kinash Oleksandr	2003-02-06	7a
6	6	Kocan Paweł	2000-02-17	7b
7	121	Kravetska Jewgenia	2002-03-04	7a

Rys. 4.2. Widok strony „Ucznie”

Jak widać na tym przykładzie, w prawym górnym rogu tej witryny jest przycisk „Dodaj ucznia”, który pozwala na dodawanie nowego ucznia do serwisu. Także jest możliwość edycji danych o uczniach. Można to zrobić wybierając jednego ucznia z tabeli. Strona edycji jest identyczna ze stroną dodawania uczniów, tylko ma wypełnione pola z odpowiednimi danymi, które można zmienić (rys. 4.3).

Rys. 4.3. Widok strony pozwalającej na zmianę danych o uczniu

Jak można zauważyć, na tej stronie jest przycisk „Usuń ucznia”, co pozwala na usunięcie ucznia z bazy danych.

Strona z punktu menu „Nauczyciele” jest identyczna ze stroną „Ucznie”, tylko że w tabeli znajdują się dane o wszystkich nauczycielach szkoły, dlatego ta strona nie jest przedstawiona.

Strona „Klasy” zawiera dane o wszystkich klasach szkoły, i jest przedstawiona na rysunku 4.4.

The screenshot shows the 'School online' admin interface. At the top, there's a blue header with the 'School online' logo, an 'Admin' button, and a 'Sign Out' button. Below the header, on the left, is a sidebar titled 'Witryna Admina' (Admin Site) with links for 'Uczniowie', 'Nauczyciele', 'Klasy', 'Przedmioty', 'Zajęcia', and 'Plan zajęć'. The main content area is titled 'Klasy' (Classes) and shows a table of existing classes:

ID	Nazwa Klasy	Usuń
1	10e	Usuń
2	2 EF-DI	Usuń
3	3 JA	Usuń
4	3EFDI	Usuń
5	7a	Usuń
6	7b	Usuń
7	9a	Usuń
8	9b	Usuń

Below the table is a form for adding a new class, labeled 'Dodaj Klasę' (Add Class), with a text input field for 'Nazwa:' (Name) and a 'Dodaj' (Add) button.

Rys. 4.4. Widok strony „Klasy” z opcją usunięcia klasy

Jak można zobaczyć, na tej stronie można od razu stworzyć nową klasę i dodać ją do bazy danych, także jest możliwość usunięcia klasy.

Na stronie „Przedmioty” można zobaczyć wszystkie przedmioty, które są prowadzone w szkole (rys. 4.5). Na tej stronie jest także możliwość dodawania i usuwania danych.

ID	Nazwa Przedmiotu	Usuń
1	Geografia	Usuń
2	Historia i społeczeństwo	Usuń
3	Informatyka PD	Usuń
4	Język angielski	Usuń
5	Język francuski	Usuń
6	Język niemiecki	Usuń
7	Język polski	Usuń
8	Matematyka	Usuń
9	Religia	Usuń

Rys. 4.5. Widok strony „Przedmioty” z możliwością usunięcia

W zakładce „Zajęcia” można dodawać zajęcia dla wybranej klasy. Działa to w taki sposób, że wybiera się przedmiot oraz nauczyciela, który będzie go prowadzić dla wybranej klasy (rys. 4.6).

Nr	Przedmiot	Nauczyciel	Usuń
1	Język angielski	Jakub Czachor	Usuń
2	Geografia	Dudak Aneta	Usuń
3	Język polski	Koszorz Małgorzata	Usuń
4	Język niemiecki	Wierzbicki Łukasz	Usuń
5	Religia	Czernecka Ewa	Usuń
6	Język francuski	Śnieżek Marek	Usuń
7	Matematyka	Klecha Tadeusz	Usuń
8	Zajęcia z wychowawcą	Grizman Jacek	Usuń

Rys. 4.6. Widok strony „Zajęcia” z możliwością dodawania i usunięcia zajęć

Jak widać, w prawym górnym rogu jest możliwość wyboru klasy, dla której są wyświetlane. Dodawać zajęcia można wybierając przedmioty i nauczycieli, które znajdują się w BD.

Jeżeli klasa ma zajęcia, to jest możliwość tworzenia planu zajęć w zakładce „Plan zajęć” (rys. 4.7). Ustawienia planu zajęć odbywa się za pomocą wyboru jednego z dodanych zajęć, do odpowiedniego miejsca w tabeli (jak to przykładowo pokazano na poniższym rysunku).

Nr	Pora lekcji	Monday	Tuesday	Wednesday	Thursday	Friday
1	08:30 09:15	Zajęcie Język angielski	Zajęcie Geografia - D	Zajęcie Język polski -	Zajęcie Matematyka -	Zajęcie Zajęcia z wyc.
2	09:25 10:10	Zajęcie Geografia - D	Zajęcie Język angielski	Zajęcie Geografia - D	Zajęcie Religia - Czer.	Zajęcie Język angielski
3	10:20 11:05	Zajęcie Język francuski	Zajęcie Matematyka -	Zajęcie Matematyka -	Zajęcie Język angielski	Zajęcie Język francuski
4	11:15 12:00	Zajęcie Religia - Czer.	Zajęcie Język polski -	Zajęcie Język angielski	Język angielski - Jakub Czachor Geografia - Dudak Aneta Język polski - Koszorek Małgorzata Język francuski - Kowalczyk Łukasz Matematyka - Czernecki Ewa ✓ Język francuski - Śniadek Marek Matematyka - Klecha Tadeusz Zajęcia z wychowawcą - Grziman Jacek	
5	12:15 13:00	Zajęcie Matematyka -	Zajęcie Język niemiecki	Zajęcie -	Zajęcie -	Zajęcie -

Rys. 4.7. Widok strony „Plan zajęć” z możliwością edytowania planu zajęć

4.2. Widok serwisu od strony nauczyciela

Po zalogowaniu do serwisu jako nauczyciel, w zakładce „Zajęcia” użytkownik może wybrać zajęcie i datę tego zajęcia, w którym chce zmienić lub dodać odpowiednią informację (rys. 4.8). Na tej stronie nauczyciel może wpisywać oceny oraz sprawdzać frekwencję uczniów. W prawym górnym rogu użytkownik wybiera jedno z możliwych zajęć, a poniżej wybiera datę tego zajęcia, spośród wszystkich możliwych dat, lista których tworzy się na podstawie planu zajęć.

Zajęcia

Zajęcia: przedmiot - "Język angielski", klasa - "7a". Data 2020-01-21

Temat zajęcia **Zadanie domowe**

Nr	Uczeń	Ocena
1	Bohdan Artem	n
2	Demianets Iwan	3
3	Kinash Oleksandr	
4	Kravetska Jewgenia	2
5	Markus Wirktor	
6	Padko Sofia	4
7	Smereka Sergii	
8	Tkachenko Serhii	

Rys. 4.8. Widok strony „Zajęcia” z opcją wpisywania ocen i nieobecności

Na rysunku 4.9 przedstawiona jest zakładka „Oceny”, która pokazuje wszystkie uzyskane oceny uczniów, dla wybranego zajęcia.

Oceny

Przedmiot - "Język angielski", klasa - "7a"

Oceny uczniów uzyskane w tym semestrze

Nr	Uczeń	Oceny
1	Artem Bohdan	5 5 2 2 4 6 4 5 2 3
2	Iwan Demianets	1 2 4 2 3 3
3	Oleksandr Kinash	3 3 4 1 3
4	Jewgenia Kravetska	6 2
5	Wirktor Markus	5
6	Sofia Padko	2 4
7	Sergii Smereka	2 5 4 3
8	Serhii Tkachenko	2 2 5 2 2 3 1
9	Maciej Wiktor	4 5 4 3

Rys. 4.9. Widok strony „Oceny” na koncie nauczyciela

Na rys. 4.10 jest przedstawiona strona „Frekwencja”, na której znajduje się informacja o frekwencji uczniów na wybranych zajęciach.

Frekwencja

Przedmiot - "Język angielski", klasa - "7a"

Nr	Uczeń	Liczba nieobecności	Procent obecności, %
1	Artem Bohdan	6	94,12
2	Iwan Demianets	2	98,04
3	Oleksandr Kinash	1	99,02
4	Jewgenia Kravetska	3	97,06
5	Wirktor Markus	0	100,00
6	Sofia Padko	0	100,00
7	Sergii Smereka	2	98,04
8	Serhii Tkachenko	0	100,00
9	Maciej Wiktor	1	99,02

Copyright 2019, Politechnika Rzeszowska, Mykita Vovk.

Rys. 4.10. Widok strony „Frekwencja” na koncie nauczyciela

Na tej stronie widoczny jest procent obecności, który jest wyliczony na podstawie liczby wszystkich zajęć wybranego przedmiotu, które odbyły się do dzisiejszego dnia.

Na stronie „Sprawdziany” nauczyciel może dodać temat oraz datę sprawdzianu, który odbędzie się kiedyś w przyszłości (rys. 4.11).

Sprawdziany

Przedmiot - "Język angielski", klasa - "7a"

Nr	Sprawdzian	Data	Usuń
1	Present Continues	2019-10-28	Usuń
2	Present Perfect	2020-01-30	Usuń
3	Past Simple	2020-02-05	Usuń
4	Present Perfect	2020-02-21	Usuń

Dodaj sprawdzian

Wpisz temat sprawdzianu

Wybierz datę sprawdzianu

2019-09-02

Dodaj

Rys. 4.11. Widok strony „Sprawdziany” z możliwością dodawania i usunięcia sprawdzianów

W zakładce „Plan zajęć” nauczyciel może zobaczyć swój własny plan zajęć (rys. 4.12), który był skonfigurowany na stronie administratora.

Nr	Pora lekcji	Monday 2020-01-20	Tuesday 2020-01-21	Wednesday 2020-01-22	Thursday 2020-01-23	Friday 2020-01-24
1	08:30 09:15	Język angielski 7a	Język angielski 7b	Język francuski 10e		
2	09:25 10:10	Język francuski 10e	Język angielski 7a		Język francuski 2 EF-DI	Język angielski 7a
3	10:20 11:05	Język angielski 7b		Język angielski 7b	Język angielski 7a	
4	11:15 12:00		Język angielski 3EF-DI	Język angielski 7a		Język francuski 10e
5	12:15 13:00	Język francuski 2 EF-DI		Język francuski 2 EF-DI		
6	13:10 13:55				Język angielski 3EF-DI	
7	14:05 14:50					
8	15:00					

Rys. 4.12. Widok strony „Plan zajęć” dla nauczyciela

Na stronie „Moja szkoła” użytkownik może zobaczyć główne informacje o szkole oraz zobaczyć listę uczniów w klasie, w której jest on wychowawcą (rys. 4.13).

Lp.	Uczeń
1	Artem Bohdan
2	Iwan Demianets
3	Oleksandr Kinash
4	Jewgenia Kravetska
5	Wirktor Markus
6	Sofia Padko
7	Sergii Smereka
8	Serhii Tkachenko

Rys. 4.13. Widok strony "Moja szkoła"

Na stronie „Dane nauczyciela” użytkownik może zobaczyć dane osobiste (rys. 4.14).

Imię nazwisko	Czachor Jakub
Data urodzenia	1988-07-14
Klasa	7a
Adres	Rzeszow ul. Hetmańska 52
Email	czchor@rzeszow.pl
Telefon	576 245 114

Rys. 4.14. Widok strony „Dane nauczyciela”

Jak można zauważyć, w prawym górnym rogu jest przycisk „Edytuj profil”, który pozwala na edycję własnych danych (rys. 4.15).

Imię nazwisko	Czachor Jakub
Data urodzenia	1988-07-14
Klasa	7a
Adres	Rzeszow ul. Hetmańska 52
Email	czchor@rzeszow.pl
Telefon	576 245 114

Zmień hasło

Aktualne hasło:

Nowe hasło:

Powtórz nowe hasło:

Zmień

Rys. 4.15. Widok strony pozwalającej na edycję danych osobistych

Na tej stronie można zmienić dane osobiste oraz jest możliwość zmiany hasła konta.

4.3. Widok serwisu od strony ucznia

Po zalogowaniu do serwisu jako uczeń, w zakładce „Oceny” użytkownik może zobaczyć wszystkie oceny uzyskane przez siebie (rys. 4.16).

The screenshot shows the 'School online' student portal interface. At the top, there's a blue header with the school logo, the text 'School online', and user information ('Artem Bohdan' and 'Sign Out'). Below the header is a sidebar titled 'Witryna Ucznia' containing links: 'Oceny', 'Frekwencja', 'Dziennik', 'Zadanie domowe', 'Sprawdziany', 'Szkoła i nauczyciele', and 'Dane ucznia'. The main content area is titled 'Oceny' and displays a table of grades for various subjects. The table has three columns: 'Przedmiot' (Subject), 'Oceny' (Grades), and 'Ocena śródroczna' (Midterm Grade). The data is as follows:

Przedmiot	Oceny	Ocena śródroczna
Geografia	3 2 5	
Język angielski	5 5 2 2 4 6 4 5 2 3	
Język francuski	3	
Język niemiecki	2 4 3	
Język polski	4 2 3	
Matematyka	3 2 3 4 4	
Religia		
Zajęcia z wychowawcą		

Rys. 4.16. Widok strony „Oceny” na koncie ucznia

W zakładce „Frekwencja” uczeń może zobaczyć informację o swojej frekwencji na różnych zajęciach (rys. 4.17).

The screenshot shows the 'School online' student portal interface, similar to the previous one but with a different section selected. The sidebar and header are identical. The main content area is titled 'Frekwencja' and displays a table of attendance data for various subjects. The table has three columns: 'Przedmiot' (Subject), 'Nieobecności' (Absences), and 'Procent obecności, %' (Percentage of attendance). The data is as follows:

Przedmiot	Nieobecności	Procent obecności, %
Geografia	2	96,77
Język angielski	6	94,12
Język francuski	1	98,36
Język niemiecki	0	100,00
Język polski	3	92,68
Matematyka	4	95,12
Religia	0	100,00
Zajęcia z wychowawcą	0	100,00

Rys. 4.17. Widok strony „Frekwencja” na koncie ucznia

W zakładce „Dziennik” uczeń może zobaczyć swój plan zajęć, oraz oceny i frekwencję, które uzyskał w ciągu wybranego tygodnia (rys. 4.18).

Nr	Pora lekcji	Monday 2020-01-20	Tuesday 2020-01-21	Wednesday 2020-01-22	Thursday 2020-01-23	Friday 2020-01-24
1	08:30 09:15	Język angielski	Geografia 5	Język polski	Matematyka 3	Zajęcia z wychowawcą
2	09:25 10:10	Geografia 2	Język angielski n	Geografia	Religia	Język angielski
3	10:20 11:05	Język francuski	Matematyka n	Matematyka n	Język angielski	Język francuski 3
4	11:15 12:00	Religia	Język polski	Język angielski	Język francuski n	Język niemiecki
5	12:15 13:00	Matematyka	Język niemiecki 3			
6	13:10 13:55					

Rys. 4.18. Widok strony „Dziennik”

Jak można zobaczyć, w tabeli są wyświetlane zajęcia odpowiednio do planu zajęć. W razie uzyskania przez ucznia oceny, ta ocena jest wyświetlana w odpowiednim miejscu i jest oznaczona niebieskim kolorem. Czerwonym kolorem jest oznaczona nieobecność ucznia na tym czy innym zajęciu.

W zakładce „Zadania domowe” uczeń może zobaczyć wszystkie wpisane przez nauczycieli zadania domowe na wybrany tydzień, które on ma wykonać (rys. 4.19).

Monday 2020-01-20	Tuesday 2020-01-21	Wednesday 2020-01-22	Thursday 2020-01-23	Friday 2020-01-24
Język francuski №345	Geografia strona 317-324 Język polski Zadanie 15, 17	Geografia strona 346-352 Język angielski ex.7 p.48	Język francuski №347	Język francuski №348 Język niemiecki Zadanie 4, strona 15

Rys. 4.19. Widok strony „Zadania domowe”

Na rysunku 4.20 pokazana jest strona „Sprawdziany”, na której użytkownik może zobaczyć wszystkie sprawdziany dodane przez nauczycieli na najbliższe cztery tygodni.

Rys. 4.20. Widok strony „Sprawdziany”

Na stronie „Szkoła i nauczyciele” uczeń może zobaczyć ogólną informację o szkole, i informację o wszystkich swoich przedmiotach i nauczycielach, którzy prowadzą je (rys. 4.21).

Rys. 4.21. Widok strony „Szkoła i nauczyciele”

Strona „Dane ucznia” jest identyczną do strony „Dane nauczyciela”, dlatego nie jest ona przedstawiona.

5. Podsumowanie i wnioski

W tej pracy został zaimplementowany serwis do zarządzania szkołą. Ten serwis pozwala użytkownikom na zarządzanie tokiem szkolenia. Zostały zaimplementowane liczne funkcje, które są niezbędne dla działania takiego serwisu. Funkcjonalność projektu została podzielona na trzy części. Korzystają z nich różnego rodzaju użytkownicy. Logika projektu została przemyślana i poprawnie zrealizowana.

W trakcie realizacji serwisu, najpierw potrzebnym było utworzenie projektu w środowisku programistycznym IntelliJ Idea. Ten program zawiera wiele szablonów projektów, dla różnego typu zadań. W tym momencie został wygenerowany projekt, który pozwala na tworzenia aplikacji na podstawie architektury MVC. Następnie została zaimplementowana relacyjna baza danych, ze wszystkimi encjami, które są potrzebne dla takiego serwisu. Baza danych powstała dzięki użyciu biblioteki Hibernate. Później została przemyślana logika oraz funkcjonalności serwisu.

W następnych etapach projekt został rozbudowany. Do projektu zostało dołączone wiele plików, katalogów oraz klas Java. W tych etapach głównym zadaniem było napisanie kodu, i na każdym z tych etapów powstawania serwisu, zostały poprawione błędy dla szybkiej i poprawnej reakcji systemu. W kodzie zostały zaimplementowane funkcje, które gwarantują odporność serwisu na wprowadzenie nieprawidłowych danych.

Dzięki użyciu biblioteki Spring, projekt został poprawnie skonfigurowany, a za pomocą lokalnego serwera TomCat dało się ten projekt uruchomić. Dzięki użyciu technologii Maven udało się dołączyć do projektu wszystkie potrzebne narzędzia i zbudować ten serwis. Projekt został zabezpieczony oraz uodporniony od złośliwego kodu dzięki użyciu technologii Git i GitHub, które pozwalają zapisywać kopie projektu na serwerze. Cały projekt został zrealizowany za pomocą języka Java, który jest nowoczesnym językiem oraz pozwala na tworzenie takiego serwisu za pomocą różnych narzędzi. Baza danych została zaimplementowana oraz dostępna na serwerze lokalnym MySQL, który jest zgodny i przydatny dla tworzenia takiej relacyjnej bazy danych.

Po zakończeniu realizacji projektu, serwis został przetestowany na poprawność działania wszystkich funkcji. W momencie, gdy funkcje serwisu działały niepoprawnie lub

serwer projektu pokazywał błąd, zostały poprawione funkcje w kodzie, które nie były odporne na niektóre działania klienta.

Od strony klienta ten serwis został w całości zrealizowany – został stworzony widok serwisu, który jest zgodny i przydatny, co pozwala na to, żeby można było z niego korzystać w dowolnej szkole średniej. Ten serwis może pomóc szkołom przejść od pisemnej dokumentacji, która dotyczy toku szkolenia, do cyfrowej. Używając tego serwisu, uczniowie mogą korzystać z elektronicznego dziennika oraz oglądać inną informację, która dotyczy toku ich szkolenia. Nauczyciele mogą wprowadzać różne dane, dotyczące ich zajęć i klas. Administrator może zarządzać wszystkimi danymi, które dotyczą uczniów, nauczycieli i zajęć.

Autor za własny wkład w realizację tej pracy uważa:

- poszukiwanie i analizę podobnych serwisów (do zarządzania szkołą),
- wybór najlepszych narzędzi pozwalających na tworzenie nowoczesnego serwisu internetowego,
- zaprojektowanie i zimplementowanie relacyjnej bazy danych zawierającej tabele danych o uczniamach, nauczycielach i zajęciach,
- zaprojektowanie oraz organizację wszystkich funkcjonalności dla każdej grupy użytkowników: nauczyciel, uczeń i administrator,
- zaprojektowanie i zaprogramowanie całego backendu i frontendu dla serwisu internetowego do zarządzania szkołą, w środowisku IntelliJ Idea.

Literatura

- [1] Herbert Schildt: Java. Kompendium programisty, Warszawa 2018, str. 33-50.
- [2] <https://medium.com/nuances-of-programming/плюсы-и-минусы-программирования-на-java-2861f4c2a0d5>. Dostęp 05.01.2020
- [3] <https://www.kv.by/archive/index2009291108.htm>. Dostęp 05.01.2020
- [4] <https://www.docsconsole.com/Spring-Core/Spring-Framework-Introduction-Features-And-Architecture>. Dostęp 05.01.2020.
- [5] <https://habr.com/ru/post/336816/>. Dostęp 06.01.2020.
- [6] https://www.devpragmatic.com/2016/10/spring-security-jak-to-dziaa_28.html. Dostęp 06.01.2020.
- [7] <https://habr.com/ru/post/435144/>. Dostęp 06.01.2020.
- [8] <https://eduprojekt.neocities.org/poradnik/szbd.htm>. Dostęp 08.01.2020.
- [9] <https://habr.com/ru/company/mailru/blog/266811/>. Dostęp 08.01.2020.
- [10] <https://www.hostinger.ru/rukovodstva/shto-takoje-mysql/>. Dostęp 08.01.2020.
- [11] <https://java-master.com/что-такое-hibernate/>. Dostęp 08.01.2020.
- [12] <https://howtodoinjava.com/hibernate-tutorials/>. Dostęp 09.01.2020.
- [13] https://javarush.ru/groups/posts/tomcat-v-java#Tomcat_что_это. Dostęp 09.01.2020.
- [14] <https://metanit.com/java/javaee/3.1.php>. Dostęp 13.01.2020.
- [15] <https://jsehelper.blogspot.com/2016/05/maven-1.html>. Dostęp 13.01.2020.
- [16] <https://wiki.rookee.ru/css/>. Dostęp 13.01.2020.
- [17] <https://ipro.ua/product/jetbrains-intellij-idea/?tab=description>. Dostęp 13.01.2020.
- [18] <https://tproger.ru/translations/difference-between-git-and-github/>. Dostęp 13.01.2020.

Sygnatura:

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza
Wydział Elektrotechniki i Informatyki

Rzeszów, 2020

STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ

SERWIS INTERNETOWY DO ZARZĄDZANIA SZKOŁĄ

Autor: Mykyta Vovk, nr albumu: EF-DI-154922

Opiekun: dr inż. Antoni Szczepański

Słowa kluczowe: serwis internetowy do zarządzania szkołą, Java, Spring, MVC, relacyjna baza danych

Celem tej pracy inżynierskiej było opracowanie serwisu internetowego do zarządzania szkołą średnią. Serwis webowy został zaimplementowany z wykorzystaniem wielu bibliotek i narzędzi, które wspomagają proces powstawania aplikacji opartej o architekturę model-widok-kontroler (ang. MVC). Programowanie odbywało się w środowisku programistycznym IntelliJ Idea, za pomocą obiektowego języka Java. Jedną z najważniejszych bibliotek, którą wykorzystano w tym projekcie, jest biblioteka Spring. W celu przechowywania i zapisywania danych została utworzona relacyjna baza danych MySQL, zawierająca informacje o wszystkich użytkownikach systemu oraz o relacjach zachodzących między nimi. Baza danych została utworzona z użyciem biblioteki Hibernate, pozwalającej na modyfikowanie jej przez kod w języku Java. Wszystkie zakładane zadania zostały zrobione i przetestowane na lokalnym serwerze.

RZESZOW UNIVERSITY OF TECHNOLOGY

Rzeszow, 2020

Faculty of Electrical and Computer Engineering

DIPLOMA THESIS (BS) ABSTRACT

WEBSITE FOR SCHOOL MANAGEMENT

Author: Mykyta Vovk, code: EF-DI-154922

Supervisor: Antoni Szczepański, PhD, Eng.

Key words: high school management system, Java, Spring, MVC, relational database

The purpose of this engineering work was to develop a website for managing high school. The website has been implemented with the use of many libraries and tools that support the process of creating an application based on the model-view-controller (MVC) architecture. Programming took place in the IntelliJ Idea programming environment, using the object-oriented Java language. One of the most important libraries used in this project is the Spring library. To store and save data, a MySQL relational database has been created that contains information about all system users and the relationships that occur between them. The database was created using the Hibernate library, which allows it to be modified by Java code. All assumed tasks have been done and tested on the local server.