

**Politechnika Krakowska  
im. Tadeusza Kościuszki**

Wydział Informatyki i Telekomunikacji

**Kraków 2020**

**Daniel Pawłowski**

Numer albumu: 132484

**Przeglądarkowa gra sieciowa czasu rzeczywistego oparta na serwerze aplikacji Spring Boot**

**A Web-based real time network game,**

**based on a Spring Boot server application**

**Praca magisterska  
 na kierunku informatyka**

Praca wykonana pod kierunkiem:  
**dr inż. Jerzego Białasa**

**Uzgodniona ocena:**......................................

...........................................................................podpisy promotora i recenzenta

Spis treści:

1. **Wstęp**
   1. Cele i zadania pracy
   2. Przegląd podobnych rozwiązań
   3. Zawartość pracy
2. **Ogólny zarys projektu**
3. **Wykorzystane technologie**
   1. Java
   2. HTML i CSS
   3. JavaScript
   4. Angular
   5. Spring framework
   6. MySQL
   7. Technologie bezpieczeństwa
      1. JWT
      2. Auth0
   8. jUnit i testy jednostkowe
   9. Technologie komunikacji
   10. Maven i biblioteki zewnętrzne
4. **Architektura aplikacji**
   1. Aplikacja frontendowa
   2. Aplikacja backendowa
   3. Architektura bazy danych
   4. Konfiguracja bezpieczeństwa
5. **Opis gry**

Podsumowanie pracy

Bibliografia

Wykaz rysunków i listingów

**1.Wstęp**

Rozwój techniki w XX wieku sprawił, że pojęcie rozrywki nabrało innego znaczenia, spora jej część przeniosła się czy to do telewizji czy też przed komputer osobisty. Jeśli w przypadku telewizji użytkownik jest tylko biernym odbiorcą przekazu, tak w przypadku rozrywki z komputerem lub telefonem ma już ogromny wpływ na to, co zobaczy. Jednym ze sposobów na spędzanie czasu są gry. Wraz z rozwojem technologii rozgrywka zmieniała swoje oblicze: od prostej gry Pong wydanej w 1972 roku, poprzez gry na automatach, później konsolach, a skończywszy na grach na komputery osobiste i na urządzenia mobilne. Według szacunków firmy Newzoo przychody branży gier wideo w 2020 roku wyniosą około 160 miliardów dolarów [1], co jeszcze kilkadziesiąt lat temu wydawało się niemożliwe. W świecie wirtualnej rozrywki każdy znajdzie coś dla siebie, ulubiona drużyna przegrała mecz? Istnieje możliwość podjęcia tego samego przeciwnika na konsoli. Walka na frontach drugiej wojny światowej? Istnieje bardzo wiele symulatorów pozwalających wcielić się w żołnierza. Poza tym liczba gatunków gier jest niezliczona, między innymi: strategie, symulatory, gry platformowe, gry zręcznościowe czy gry logiczne. Innym ważnym czynnikiem uatrakcyjniającym rozgrywkę było powstanie Internetu, który pozwala na rozgrywkę z dowolną liczbą osób podłączonych do sieci. Na początku lat 90 powstał protokół http, pozwalał on na przesyłanie tak zwanego hipertekstu. Tekst taki jest następnie przetwarzany przez przeglądarkę internetową, a przetworzona treść wyświetlana jest na ekranie dla użytkownika. W poniższej pracy opisana zostanie gra przeniesiona do przeglądarki Internetowej.

**1.1 Cel i zadania pracy**

Celem pracy jest opis projektu, który jest przeglądarkową grą czasu rzeczywistego. Gracz rozpoczyna rozgrywkę w momencie rejestracji i trwa ona nieprzerwanie aż do momentu, w którym gracz postanowi usunąć swoje konto lub zostanie ono usunięte z innego powodu. Większość obliczeń jest wykonywana na głównym serwerze aplikacji, który działa nieprzerwanie, akcje wykonywane przez użytkownika w przeglądarce pozwalają na przesyłanie zapytań przy użyciu protokołu http do serwera, który następnie zapisuje te dane do bazy danych, a także wysyła do użytkownika informacje zwrotne. Realizacja projektu składa się z:

* Oprogramowanie silnika gry
* Stworzenie interfejsu użytkownika (jako projekt Angular),
* Konfiguracja zewnętrznego serwera uwierzytelniania i autoryzacji (z wykorzystaniem strony auth0.com),
* Utworzenie struktury bazy danych do przechowywania informacji o grze,
* Utworzenie testów jednostkowych zarówno dla serwera aplikacji jak i aplikacji Angular,
* Stworzenie serwera aplikacji, który będzie:
  + obsługiwał zapytania użytkowników,
  + wykonywał obliczenia związane bezpośrednio z rozgrywką,
  + dbał o bezpieczeństwo danych aplikacji,
  + komunikował się z bazą danych

Gra stworzona na potrzeby tej pracy jest grą strategiczną umieszczoną w świecie fantasy, gdzie rozgrywka toczy się nieprzerwanie, posiada ona aspekty ekonomiczne i wojenne. Została ona zaprojektowana w taki sposób aby jej dalszy rozwój był prosty i nie wymagał zmiany głównych części projektu.

**1.2 Przegląd podobnych rozwiązań**

W sieci istnieje wiele podobnych gier jednak każda z nich będzie posiadać cechy, które będą rozróżniać je od innych. Do takich cech należeć będą:

* Ekonomia rozgrywki,
* System walki,
* Uniwersum, w którym odbywa się rozgrywka
* I wiele innych

W rozdziale tym zostanie omówione kilka najbardziej rozpoznawalnych gier przeglądarkowych

**OGame**



**Rys. 1.1 Logo gry OGame**

Gra ta została stworzona przez niemiecką firmę Gameforge w 2002 roku. W roku 2005 została ona przetłumaczona na język polski i wtedy też powstał pierwszy polski serwer. Rozgrywka toczy się w przestrzeni kosmicznej, każdy gracz po założeniu konta otrzymuje jedną planetę i niewielką ilość zasobów na dalszy rozwój. W grze co jakiś czas powstają nowe serwery, nazywane uniwersami. W każdym uniwersum istnieje 9 galaktyk, a w każdej galaktyce 500 układów słonecznych, w których jest miejsce dla 15 planet. Każda z planet może mieć pozycję od 1 do 15 w danym układzie słonecznym, im wyższy numer pozycji planety, tym na planecie występuje niższa średnia temperatura. Każdy gracz może zasiedlać nowe planety (tworzyć kolonie), a ich limit ograniczony jest poziomem techniki astrofizyki. W rozgrywce występują trzy główne surowce: metal, kryształ i deuter. Mogą one zostać wykorzystane do rozbudowy budynków, przeprowadzania badań, tworzenia jednostek obronnych lub statków kosmicznych, a deuter jest też wykorzystywany jako paliwo dla floty. Oprócz trzech wspomnianych powyżej surowców każda planeta posiada też energię, którą można pozyskać za pomocą budowy elektrowni lub umieszczania satelitów słonecznych na orbicie planety. Energia służy do zasilania kopalni, może także pomóc w badaniach. Gracze wchodzą w interakcje między innymi poprzez handel lub ataki. W grze istnieją także sojusze, które mogą toczyć ze sobą wojny.



**Rys. 1.2 Ekran główny gry OGame**

**Plemiona**



**Rys. 1.3 Logo gry Plemiona**

Plemiona (zwane też w oryginale jako Tribal Wars) jest grą czasu rzeczywistego, której akcja toczy się w średniowieczu. Twórcą jest studio InnoGames, powstała ona w 2003 roku. Polska wersja gry została uruchomiona w 2006 roku. Istnieje wiele serwerów zwanych też światami. Każdy nowy gracz rozpoczyna rozgrywkę z jedną wioską. W grze istnieje trzy rodzaje surowców: drewno, glina i żelazo pozyskiwane odpowiednio z tartaku, cegielni i huty żelaza. Ilość surowców jaka otrzymywana jest w określonym okresie czasu zależna jest od poziomu powyższych budynków. Zasoby te przechowywane są w spichlerzu i należy pamiętać o rozbudowywaniu go, gdyż przekroczenie jego pojemności powoduje utratę surowców. Do tworzenia jednostek bojowych należy zadbać o odpowiednią liczbę ludności, którą można zwiększyć poprzez podnoszenie poziomu zagrody. Rozbudowa rynku pozwala na handlowanie surowcami wraz z najbliższymi sąsiadami. W zależności od poziomów różnych budynków dostępne są różne rodzaje wojsk. W trakcie rozgrywki istnieje możliwość dołączenia do plemienia, współplemieńcy mogą pomóc w rozbudowie wioski lub też wspomóc swoimi jednostkami militarnymi w przypadku ataku innych graczy. Wraz ze startem serwera administratorzy gry określają warunki, po spełnieniu których gra zostaje wygrana.



**Rys 1.4 Pogląd wioski w grze Plemiona**

**Shakes & Fidget**



**Rys 1.5 grafika z gry Shakes & Fidget**

Shakes & Fidget jest przeglądarkową grą, gdzie gracz posiada swoją postać. Tworząc ją do wyboru jest 8 profesji oraz 8 ras. Głównymi zdolnościami w grze są siła, zwinność, magia, wytrzymałość oraz szczęście, których dobór zależy od gracza. Po stworzeniu postaci każda osoba otrzymuje 1 sztukę złota oraz 15 sztuk grzybów - są to 2 główne surowce w grze. Można za nie kupować rzeczy do ekwipunku w zbrojowni oraz gabinecie magii, a także eliksiry na potrzeby ulepszania zdolności. Postać można rozwijać poprzez wybieranie się na wyprawy, walki na arenie, atakowanie lochów czy też walki wraz z członkami gildii. Najważniejszym miejscem do rozwoju postaci jest karczma, w której należy wybrać jedną z trzech proponowanych przez karczmarza misji, po ukończeniu których otrzymać można złoto oraz doświadczenie, a rzadziej także grzyby. Aby misje te trwały krócej niezbędne jest posiadanie wierzchowca. Można je wynajmować na okres 14 dni. Ważnym elementem rozgrywki są gildie, a każdy może założyć swoją w cenie 10 sztuk złota. Bycie członkiem gildii pozwala na zwiększone zdobycze zarówno jeśli chodzi o doświadczenie jak i o złoto. Oferują one także możliwości wojen gildyjnych czy wspólnych wypraw przeciw potworom. Celem gry jest zdobycie jak najpotężniejszej postaci czy to poprzez zdobywanie jak największej ilości doświadczenia czy też przez skompletowanie najlepszego ekwipunku.

**1.3 Zawartość pracy**

W rozdziale drugim opisany zostanie ogólny zarys projektu – przedstawione zostaną główne elementy, jakie role będzie posiadał każdy z nich i w jaki sposób odbywa się komunikacja między nimi.

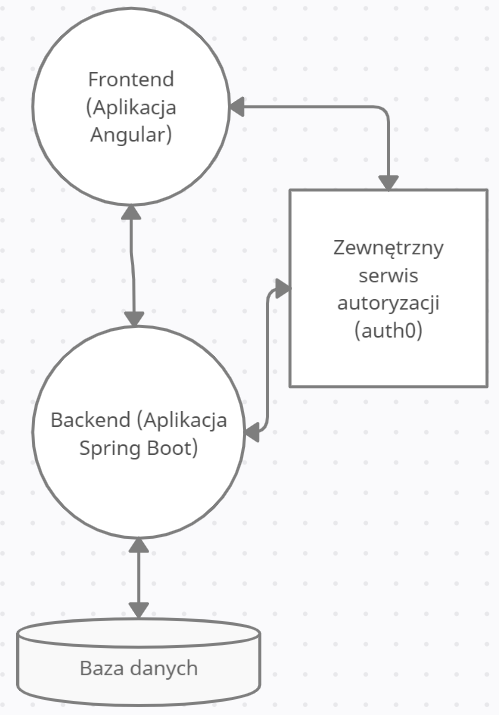
Rozdział trzeci zostanie poświęcony przedstawieniu wszystkich wykorzystanych technologiach – krótkich historiach ich powstania, opisie, zastosowaniach a także zastosowaniach w projekcie.

W kolejnym, czwartym rozdziale każdy z elementów aplikacji zostanie dogłębnie omówiony, zostaną wskazane najważniejsze komponenty, a także przedstawione zostaną konfiguracje niezbędne do prawidłowego działania.

Piąty rozdział opisywał będzie właściwą rozgrywkę – jakie akcje może wykonać użytkownik, a także zasady działania wszystkich elementów rozgrywki.

Rozdziały szósty i siódmy skupią się na podsumowaniu projektu, a także na wskazaniu materiałów źródłowych, na podstawie których powstała ta praca.

**2. Ogólny zarys projektu**

****

Rys 2.1 Ogólna architektura rozwiązania

Pierwszym elementem aplikacji jest komponent frontendowy. Został on napisany w technologii Angular. Jego głównym zadaniem jest wyświetlanie treści użytkownikowi. Komunikuje się on bezpośrednio z serwerem backendowym, a także z zewnętrznym serwisem autoryzacyjnym. W komponencie tym użyte zostały technologie takie jak:

* Html – wykorzystywna do tworzenia treści i struktury wyświetlanych stron,
* Css – odpowiedzialny jest za wygląd i wyświetlanie się elementów html
* JavaScript – dzięki niemu możliwe jest dodanie dynamicznych elementów pozwalających na interakcję z użytkownikiem,
* JSON – format danych wykorzystywany w przesyle informacji pomiędzy komponentami

Autoryzacja użytkownika możliwa jest poprzez wykorzystanie zewnętrznego serwisu autoryzacji o nazwie auth0. Na potrzeby projektu została utworzona w serwisie tym aplikacja. Dzięki odpowiedniej konfiguracji pozwala ona na rejestrację oraz logowanie w bezpieczny sposób.

Aplikacja backendowa spina wszystkie komponenty. W tym miejscu odbywają się wszystkie obliczenia dotyczące bezpośrednio rozgrywki, wysyłania danych niezbędnych do wyświetlenia w aplikacji frontendowej, danych potrzebnych do zapisania w bazie danych i decyzji do jakich zasobów powinien mieć dostęp dany użytkownik. Najważniejsze technologie użyte w tym komponencie:

* Java,
* Spring,
* Maven

Baza danych – jak wskazuje nazwa, służy do przechowywania danych o kontach użytkowników. Wykorzystuje ona model tak zwanej relacyjnej bazy danych, a systemem zarządzania jest MySQL.

Komunikacja między aplikacją frontendową a serwisem auth0 odbywa się za pomocą protokołu http. Aby możliwa była jakakolwiek akcja podjęta przez użytkownika potrzebuje on tak zwanego tokenu autoryzacyjnego. Jeśli takowy nie jest posiadany – musi odbyć się autoryzacja – polega ona na tym, że korzystający z aplikacji zostaje przekierowany na specjalną stronę logowania, gdzie po wprowadzeniu swoich danych i udanej autoryzacji otrzymuje on token. Nie będzie on widoczny dla użytkownika ale zostanie zapisany przez aplikację i do czasu jego wygaśnięcie będzie służył do autoryzacji wszystkich podjętych akcji.

Komunikacja aplikacji angular i spring boot podobnie jak w poprzednim przypadku odbywa się przy wykorzystaniu protokołu http. Wszelkie aktywności, które wykonuje użytkownik zostają upakowane w zapytanie http wykorzystujące format danych JSON jako ciało zapytania i odpowiednie nagłówki (w tym nagłówek autoryzacyjny). Skutki podjętych działań są następnie przetwarzane przez aplikację backendową i zostaje wysłana informacja zwrotna.

Aplikacja spring boot komunikuje się z serwisem auth0 aby sprawdzić poprawność tokena podawanego w zapytaniach użytkownika i aby uzyskać dodatkowe informacje o tymże użytkowniku.

Ostatnim komponentem istniejącym w aplikacji jest baza danych, do której dostęp ma komponent backendowy. Komunikacja odbywa się przez interfejs JDBC.

**3. Wykorzystane technologie**

**3.1 Java**



**Rys 3.1 Logo języka programowania Java**

Java jest obiektowym językiem programowania opartym o klasy. Główną domeną języka Java jest „write once, run anywhere” (napisz raz, uruchom wszędzie), oznaczająca że raz skompilowany kod może działać na dowolnej maszynie wspierającej ten język programowania. Kod Javy kompilowany jest na tak zwany kod bajtowy, który może zostać uruchomiony przez maszynę wirtualną Javy (JVM) niezależnie od systemu na którym działa. Składnia tego języka jest podobna do języka C/C++ jednak nie posiada ona kilku niskopoziomowych funkcjonalności (takich jak na przykład wskaźniki), a posiada na przykład wbudowany mechanizm zarządzania pamięcią (ang. garbage collector) czy też mechanizm refleksji, który nie występuje w tradycyjnych językach programowania. Głównym założeniem Javy jest obiektowość. Oznacza to, że procedury i obiekty są ze sobą ściśle powiązane. Java jest też językiem silnie typowanym, co rozumieć należy przez to, że każde wyrażenie będzie mieć przypisany typ danych i próba wykorzystania operacji dla typu do tego nie przeznaczonego będzie kończyć się niepowodzeniem. Pierwsza wersja Javy oznaczona jako 1.0 została wydana w 1996 roku przez firmę Sun Microsystems. Wykorzystywana była ona głównie przez przeglądarki podczas uruchamiania tak zwanych apletów na stronach internetowych. Java jest obecnie jednym z najpopularniejszych języków programowania. Według tak zwanego wskaźnika TIOBE index, wskazującego popularność różnych języków programowania tworzonego przez firmę TIOBE, za grudzień 2020 roku, z wynikiem 12,53% Java jest drugim najpopularniejszym językiem programowania i ustępuje tylko językowi C [przypis].

**3.2 HTML i CSS**



**Rys 3.2 Logo HTML5 i CSS3**

Podstawowymi narzędziami niezbędnymi do tworzenia stron internetowych są HTML (ang. HyperText Markup Language) I CSS (ang. Cascading Style Sheets).

* HTML jest to język znacznikowy używany do strukturyzowania i wyświetlania stron internetowych.
* CSS jest to język służący do formatowania wyglądu, a także wyświetlania elementów html.

Pomimo że oba języki są odpowiedzialne za inne rzeczy to są ze sobą nierozerwalnie połączone. Określenie, że HTML to język znacznikowy oznacza, że używa on znaczników (ang. tags) do tworzenia elementów strony. W dawnych latach były to znaczniki między innymi „<div>” czy „<span>” jednak w nowszych wersjach dodano znaczniki takie jak: „<section>”, „<nav>” czy „<header>”. Mają one na celu usprawnić pracę wyszukiwarek internetowych. Odpowiednio zaprojektowana witryna internetowa dużo łatwiej może przekazać informacje o tym, gdzie znajduje się treść właściwa danej strony, a gdzie jest menu nawigacji. W języku CSS opisuje się selektory za pomocą bloków deklaracji. Każda deklaracja składa się z właściwości i wartości. Właściwościami mogą być: szerokość, wysokość, kolor czy położenie. W zależności od typu właściwości, wartości mogą przyjmować postać pikseli, procentów lub nazw kolorów. CSS może być dołączony do pliku HTML na trzy sposoby:

* Poprzez zapisanie bloku deklaracji tuż za znacznikiem html (tak zwany „inline css”),
* Poprzez umieszczenie znacznika „<style>” wewnątrz dokumentu HTML,
* Poprzez import z pliku za pomocą znacznika <link rel=”stylesheet” href=”style.css”/>

**3.3 JavaScript**



**Rys 3.3 Logo JavaSript**

JavaScript jest językiem skryptowym programowania wysokiego poziomu. Wraz z CSS i HTML jest podstawą interaktywnych stron internetowych. Oprócz tego JavaScript posiada szersze zastosowanie, wykorzystuje się go w aplikacjach desktopowych, dokumentach PDF czy też aplikacjach internetowych po stronie serwera. JS jest językiem interpretowalnym co oznacza, że program napisany w takim języku nie jest kompilowany (nie jest zamieniany na postać bitową), a przechowywany jest jako kod źródłowy. Następnie podczas uruchomienia takiego programu jest on czytany, interpretowany, a na koniec interpreter tego języka wykonuje go. Język ten posiada wiele paradygmatów, można w nim programować imperatywnie, obiektowo i funkcyjnie.

JavaScript posiada następujące cechy:

* Kod wykonywany jest po stronie klienta co pozwala na odciążenie serwera,
* Skrypty mogą zostać wykonanie na dowolnej stronie internetowej niezależnie jakie posiada ona rozszerzenie
* JavaScript jest niezależny od systemu operacyjnego
* Składnia języka jest relatywnie prosta
* Do zmiennej można przypisać funkcję
* Wykorzystywane jest typowanie dynamiczne co oznacza że typ zmiennej nadawany jest podczas działania programu i może się zmieniać.
* Typowanie słabe – jeśli zachodzi taka potrzeba, typ danych zmiennej zostaje zmieniony

**3.4 Angular**



**Rys 3.4 Logo narzędzia Angular**

Angular jest jednym z wielu narzędzi ułatwiających budowę interaktywnych stron internetowych. Innymi podobnymi rozwiązaniami są na przykład Vue.js, React czy Backbone. Pierwotnie projekt ten nazywał się AngularJS jednak od wersji 2 przyjęła się nazwa Angular. Narzędzie to zostało stworzone przez pracownika Google – Misko Hevery’ego, z czasem stało się oficjalnym produktem tej firmy i jest przez nią aktywnie wspierane. Mimo zdobycia ogromnej popularności – AngularJS posiadał sporo problemów, wiele funkcjonalności wymagało większej standaryzacji i unowocześnienia, a wydajność pozostawiała wiele do życzenia. W 2015 została opublikowana wersja 2, a narzędzie to przyjęło nową nazwę – Angular. Od tego czasu co pół roku publikowana jest nowa wersja Angulara,

Główne cechy narzędzia Angular:

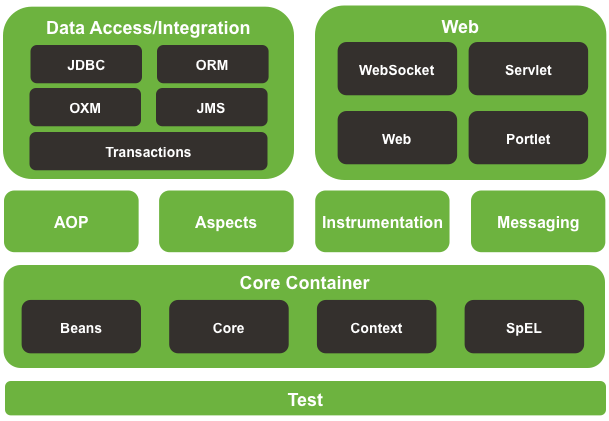
* Wstrzykiwanie zależności (dependency injection) – podobnie jak w przypadku Springa, pozwala na tworzenie obiektów zawierających inne obiekty jako pola klasy.
* Własne komponenty – zawierające w sobie szablon (template), style css, klasę wraz ze wszystkimi polami i metodami, a także testy jednostkowe. Komponenty mogą być zagnieżdżane w innych komponentach i modułach poprzez dodanie w pliku html odpowiedniego znacznika.
* Dyrektywy – pozwalają na dodawanie nowych funkcjonalności do elementów na stronie.
* Zone.js – używany jest jako narzędzie obsługujące zdarzenia takie jak działania użytkownika, przychodzące zapytania http, a także wydarzenia występujące czasowo. Niektóre ze zdarzeń mogą powodować zmiany stanu (np. zmiana imienia spowoduje zmianę wyświetlania pełnej nazwy). Zone.js oblicza jakie akcje należy podjąć ponieważ zaszło dane zdarzenie i dopiero po obliczeniu pełnego łańcucha zdarzeń wysyłane jest powiadomienie do Angluara o liście akcji niezbędnych do podjęcia.
* Wszechstronny rendering – współczesne strony internetowe muszą być czytelne na wszelkiego rodzaju urządzeniach – od smartfonów, przez tablety aż po ekrany komputerów. Angular posiada wsparcie dla wyświetlania stron na wielu rodzajach urządzeń końcowych.
* Projekt Angular posiada bardzo szczegółową dokumentację opisującą wszystkie jego elementy wraz z ogromną ilością praktycznych przykładów wykorzystania.
* CLI – jest to narzędzie wspierające tworzenie aplikacji Angular. Pozwala za pomocą komend w terminalu na wykonanie wielu czynności, których wykonanie ręczne zajęłoby wiele więcej czasu, są to na przykład:
  + Stworzenie aplikacji,
  + Dodanie do projektu komponentu, dyrektywy czy też serwisu,
  + Wykonanie testów jednostkowych,
  + Zbudowanie projektu.

**3.5 Spring Framework**



**Rys 3.5 Logo Spring Framework**

Spring Framework jest wszechstronnym narzędziem programistycznym dla nowoczesnych aplikacji biznesowych tworzonych w języku Java. Narzędzie to działa na licencji open-source co oznacza, że kod jest ogólnie dostępny i nie jest wymagane uiszczanie żadnych opłat ze używanie go. Pierwsza wersja została napisana przez Roda Johnsona w 2003 roku i opublikowana na licencji Apache 2.0. Jednym z głównych mechanizmów wykorzystywanych przez to narzędzie jest tak zwane wstrzykiwanie zależności (ang. dependency injection). Spring Framework posiada tak zwany „kontener” (inversion of control container). Jest on odpowiedzialny za zarządzaniem obiektami: tworzy je, uruchamia metody inicjalizujące, a także konfiguruje je tworząc odpowiednie powiązania. Obiekty stworzone przez to narzędzie nazywane są „beanami”. Konfiguracja takiego kontenera może być zapisana w pliku .xml lub zostać umieszczona w plikach klas Javy z użyciem adnotacji. Platforma Spring posiada wiele modułów ułatwiających tworzenie aplikacji, ogólny schemat został pokazany na rysunku 3.6.



**Rys 3.6 Schemat modułów Spring Framework**

Spring Data – głównym zadaniem tego modułu jest zapewnienie przyjaznego i zwięzłego modelu dostępu do bazy danych. Pozwala w prosty sposób połączyć się z relacyjnymi i nierelacyjnymi bazami danych, a także usługami sieciowymi zapewniającymi dostęp do danych. Projekt ten jest zbiorem pomniejszych modułów zapewniających dostęp do różnych implementacji bazodanowych takich jak JDBC, JPA czy MongoDB. Spring Data ułatwia także konstruowanie i wysyłanie zapytań bazodanowych na przykład poprzez tworzenie zapytania na podstawie nazwy metody.

Spring Web MVC – moduł służący do budowy aplikacji przeglądarkowych opartych o architekturę Model-View-Controller. Komponenty te odpowiedzialne są za:

* Model – definiuje i przechowuje dane używane w aplikacji,
* View – odpowiedzialny jest za wyświetlanie danych zawartych w modelu, zazwyczaj używa kodu html przetwarzanego następnie przez przeglądarkę,
* Controller – służy do przetwarzania zapytań użytkownika, budowę odpowiedniego modelu i przekazywanie go do komponentu „view” aby ten mógł wyświetlić odpowiednie dane.

Spring Web MVC oparty jest o tak zwany DispatcherServlet, którego schemat działania został przedstawiony na rys 3.7. Element ten zarządza wszystkimi zapytaniami http, zarówno przychodzącymi jak i wychodzącymi.



**Rys 3.7 Diagram działania DispatcherServleta.**

Przychodzące zapytanie http zostaje przetwarzane w następujących krokach:

1. Odebrane zapytanie zostaje odczytane przez HandlerMapper, który następnie uruchamia odpowiedni Controller.
2. Controller odbiera zapytanie i wywołuje odpowiednią metodę w serwisie w zależności od wywołanej usługi i metody http (POST, GET itd.). Zapytanie zostaje przetworzone według logiki biznesowej zawartej w wywołanej metodzie, a następnie ustawiony zostaje model, który zostaje wysłany z powrotem do DispatcherServleta.
3. Kolejnym krokiem jest pobranie z View Resolvera odpowiedniego widoku.
4. Ostatecznie widok zostaje wysłany do przeglądarki gdzie będzie wyświetlony.

Wszystkie wspomniane powyżej elementy (Handler Mapper, Controller, View Resolver) są częściami kontekstu webowego (WebApplicationContext).

Spring AOP – jest to moduł służący jako element pośredniczący między różnymi komponentami. Moduł ten pozwala na programowanie zorientowane aspektowo. Oznacza to, że tworzony kod jest bardziej modularny co z kolei oznacza możliwość na przykład oddzielenia warstwy technicznej od logiki biznesowej.

Spring Security – moduł służący do zapewnienia bezpieczeństwa aplikacji. Posiada mechanizmy wspierające autoryzację i dostęp do zasobów dla aplikacji bazujących na frameworku spring. Moduł ten pozwala ochronić aplikację przed atakami takimi jak: przechwytywanie sesji, clickjacking czy cross site request forgery.

Spring Boot –pozwala w prosty sposób uruchomić aplikację opartą o spring. Głównymi cechami tego rozwiązania są:

* Proste uruchomienie – w przeciwieństwie do „podstawowego” springa, Spring Boot posiada wbudowany serwer aplikacji, a także wszystkie pozostałe elementy niezbędne do jej uruchomienia,
* Automatyczna konfiguracja – aby uruchomić aplikację nie jest potrzebna żadna dodatkowa konfiguracja.
* Szybkość – wykorzystanie Spring Boota pozwala na dużo prostsze tworzenie aplikacji co pozwala na szybszy i tańszy proces developmentu.

**3.6 SQL i MySQL**



**Rys 3.8 Logo MySQL**

Structured Query Language (w skrócie SQL) jest to język zapytań wykorzystywany w relacyjnych bazach danych. Został on stworzony przez firmę IBM w latach siedemdziesiątych XX wieku i z czasem w przypadku komunikacji z serwerami baz danych stał się standardem, pierwszy standard został opracowany w 1986 roku, a po raz pierwszy w komercyjnym zastosowaniu wykorzystany został przez firmę Oracle. W składni SQL wyróżnia się cztery główne podzbiory:

* SQL DCL (język kontroli nad danymi) – służy do zarządzania dostępem i uprawnieniami do obiektów bazodanowych. Do najważniejszych poleceń w tym zbiorze zaliczyć można:
  + GRANT – służy do nadawania uprawnień dla użytkownika lub do obiektu,
  + REVOKE – służy do odbierania uprawnień użytkownikowi,
  + DENY – służy do zabronienia wykonywania danej operacji.
* SQL DDL (język definicji danych) - służy do modyfikacji struktur, w których przechowywane są dane. Do najważniejszych poleceń w tym zbiorze zaliczyć można:
  + CREATE – służy do tworzenia struktury (indeksu, bazy, tabeli lub innych),
  + DROP – służy do usuwania struktur,
  + ALTER – służy do modyfikowania struktur.
* SQL DQL (język definiowania zapytań) – służy do pobierania rekordów z bazy danych. Głównym poleceniem tej grupy jest SELECT.
* SQL DML (język modyfikacji danych) – służy do wykonywania czynności związanych z danymi. Najważniejszymi poleceniami są:
  + INSERT – służy do dodawania danych w bazie,
  + UPDATE – służy do modyfikacji danych w bazie,
  + DELETE – służy do usuwania danych z bazy,

MySQL jest systemem zarządzania relacyjnymi bazami danych. Został stworzony w 1995 roku przez firmę MySQL AB, a współcześnie rozwijany jest przez firmę Oracle. Określenie „relacyjna baza danych” oznacza, że tabele będące częścią bazy danych posiadają połączenia między sobą, nazywane kluczami. Tabele pozostają ze sobą w różnych „relacjach”. Mogą być to:

* Jeden do jednego,
* Wiele do jednego,
* Wiele do wielu.

Klucz główny posiada następujące cechy:

* W ramach jednej tabeli może być tylko jeden klucz główny,
* Składa się z jednej lub wielu kolumn,
* Wszystkie kolumny w ramach klucza głównego muszą zawierać wartości NOT NULL,
* Klucz główny identyfikuje rekord w danej tabeli,
* Wartości klucza głównego muszą być unikatowe dla każdego z rekordów w danej tabeli.

Klucz obcy to kolumna w danej tabeli, która musi wskazywać na klucz główny w innej tabeli.

Systemy bazodanowe posiadają także mechanizm indeksów. Pozwalają one na sprawniejsze pozyskiwanie danych z tabel kosztem zwiększenia czasu wstawiania i edycji danych. Ich działanie można porównać z tworzeniem spisu treści dla książki.

**3.7 Technologie bezpieczeństwa**

**3.7.1 JWT**

Jednym ze sposobów uwierzytelniania zapytań do aplikacji są tokeny bezpieczeństwa ( ang. security tokens). Są to podpisane struktury danych zawierające dane o eminencie (ang. issuer) i podmiocie (ang. subject) a także o czasie wygaśnięcia. Typowy scenariusz użycia tokenu bezpieczeństwa:

1. Podmiot (subject) wysyła zapytanie o token.
2. Eminent (issuer) tworzy token i wysyła go do klienta.
3. Podmiot wysyła zapytanie wraz z tokenem do zasobu (resource).
4. Zasób, który ma zaufane powiązanie z eminentem sprawdza poprawność tokenu i wysyła dane o które w zapytaniu prosił podmiot.

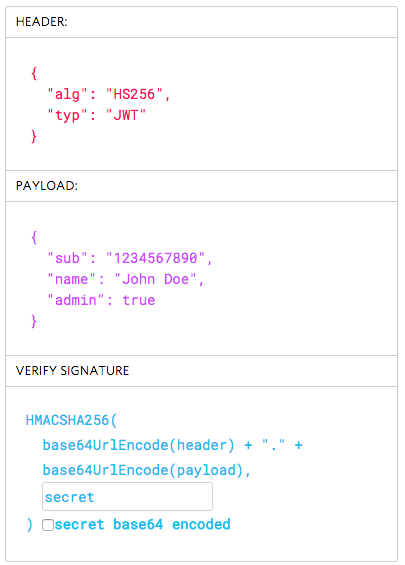
Istnieje kilka rodzajów mechanizmów bezpieczeństwa:

* SAML (ang. Security Assertion Markup Language) – jest to protokół wykorzystywany w uwierzytelnianiu użytkownika. Bazuje on na standardzie XML. Protokół ten wykorzystuje się głównie w mechanizmie pojedynczego logowania (SSO).
* SWT (ang. Simple Web Token) – jest to token używany w przeglądarkach internetowych, który zawierany jest w adresie (url) jako dopisane pary klucz-wartość. Używa się w nim wyłącznie kluczy symetrycznych.
* JWT (ang. JSON Web Token) – mechanizm zapisu informacji bazujący na formacie JSON. JWT składa się z :
  + Nagłówka (ang. header) – zawierającego informację o metadanych, a także zastosowanych algorytmach i użytych kluczach.
  + Danych właściwych (ang. payload) – zawierających informacje takich jak:
    - Eminent
    - Podmiot
    - Data wystawienia
    - Data wygaśnięcia i inne.
  + Podpisu (ang. signature).

Elementy te zakodowane są algorytmem BASE64URL. Przykładowy JWT w postaci zakodowanej został ukazany na rysunku 3.9, a odkodowany na rysunku 3.10. W odróżnieniu od SWT, JWT posiada o wiele więcej możliwości jeśli chodzi o podpis – możliwe jest używanie zarówno symetrycznych jak i niesymetrycznych sposobów szyfrowania. JWT używany jest jako wartość tak zwanego access tokena. W zapytaniach do serwera aplikacji używany jest nagłówek „Authorization”, który ma wartość „Bearer + < token>”.



**Rys 3.9 Przykładowy zakodowany JWT.**



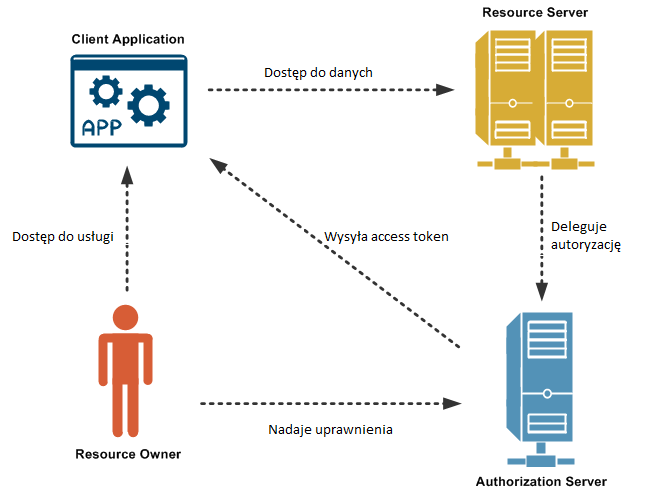
**Rys. 3.10 Odkodowany JWT z rys. 3.8**

**3.7.2 OAuth 2.0**

Narzędzie OAuth 2.0 jest protokołem, który pozwala użytkownikowi uzyskać dostęp do zasobów serwera bez przekazywania do niego swoich poufnych danych. OAuth wprowadza warstwę autoryzacyjną i oddziela rolę klienta od roli właściciela zasobu. Klient wykonuje zapytanie o dostęp do zasobów posiadanych przez ich właściciela znajdujących się na serwerze zasobów i pozyskuje je za pomocą innych danych uwierzytelniających niż np. dane logowania użytkownika. W tym przypadku używany jest tak zwany „Access Token” zawierający w sobie dane takie jak zakres udzielanych zgód, czas ważności i inne. Token taki jest udzielany dla klienta przez serwer autoryzacyjny za pozwoleniem właściciela zasobów. Po jego uzyskaniu klient używa go aby uzyskać dostęp do zasobów znajdujących się na serwerze zasobów. Przepływ OAuth obejmuje następujące role:

* Właściciel zasobów (ang. Resource Owner) – jednostka, która może udostępnić dostęp do chronionego zasobu, najczęściej jest to użytkownik końcowy.
* Serwer zasobów (ang. Resource Server) – serwer, na którym znajdują się chronione zasoby, jest to aplikacja, do której dostęp chcemy uzyskać.
* Aplikacja kliencka (ang. Client Application) – jest to aplikacja, która w imieniu właściciela zasobów wysyła zapytania o zasób.
* Serwer autoryzacyjny (ang. Authorization Server)– serwer, który uwierzytelnia właściciela zasobów i przyznaje mu access token po udanej autoryzacji.

Sposób interakcji między rolami został przedstawiony na rysunku 3.11



**Rys 3.11 Role i ich interakcje w protokole OAuth**

W protokole OAuth zdefiniowane są cztery sposoby uzyskania access tokena. Każdy z nich opisany jest wartością „grant type”. W zależności od typu aplikacji używany będzie inny grant type. Wspomniane sposoby to:

1. Authorization Code Flow – używany w aplikacjach przeglądarkowych.
2. Implicit Flow with Form Post – używany w aplikacjach przeglądarkowych posiadających tylko jeden plik html.
3. Resource Owner Password Flow – używany w przypadku aplikacji o wysokim zaufaniu
4. Client Credentials Flow – używany w przypadku komunikacji między M2M (machine-to-machine).

Protokół OAuth posiada dwa punkty końcowe (ang. endpoints), /authorization i /oauth/token. Pierwszy z nich służy do interakcji z właścicielem zasobu aby uzyskać dostęp do chronionego zasobu. Parametrami zapytania do tego punktu końcowego są:

* response\_type – określające który grant type będzie używany,
* client\_id – określający identyfikator klienta,
* redirect\_url – wskazujący pod jaki adres należy przekierować użytkownika po udanej autoryzacji,
* scope – lista pozwoleń o które prosi klient,
* state – wartość używana aby zabezpieczyć aplikację przed atakami CSRF.

/authorize jest używany przez pierwsze dwa sposoby autoryzacji. Dla Authorization Code zwróci on kod autoryzacyjny (ang. authorization code), a w przypadku Implicit Grant zwróci on access token. Drugi z dostępnych punktów końcowych w przypadku Authorization Code wymienia przesłany kod autoryzacyjny na access token, a w przypadkach Client Credentials Flow i Resource Owner Passowrd Credentials Flow po otrzymaniu danych logowania klienta zwraca access token.

**3.8 jUnit i testy jednostkowe**



**Rys 3.11 Logo jUnit**

jUnit jest biblioteką służącą do sprawdzania poprawności działania programu napisanego w języku Java. W celu przeprowadzenia testów jednostkowych danej klasy należy utworzyć w katalogu „Test” klasę o takiej samej nazwie z dopiskiem „Test”. Aby dana klasa stała się klasą testową należy utworzyć co najmniej jedną publiczną metodę i ponad nią dodać adnotację „@Test”. Narzędziem sprawdzającym czy funkcjonalność działa poprawnie jest tak zwana asercja. Najważniejszymi asercjami są:

* assertEquals – sprawdzające czy dane wartości są identyczne,
* assertNull/assertNotNull – sprawdające czy dana wartość jest/nie jest równa null
* assertTrue/assertFalse – sprawdzające czy dane wyrażenie jest prawdziwe lub fałszywe
* assertSame – sprawdzające czy obiekty mają tą samą referencję.

Test jednostkowy jest to sposób w jaki przeprowadza się testowanie programu. Wydziela się w nim najmniejszą jego część i poddaje testom.

**3.9 Technologie komunikacji**

W projekcie do komunikacji między komponentami używany jest protokół http. Całość oparta jest jednak na architekturze REST którą można określić jako „sposób w jaki powinno korzystać się z http”. REST opisuje aby metody http wykorzystywać w następujący sposób:

* GET – metodę tę wykorzystuje się do pobierania danych. W przypadku zapytań za pomocą tej metody występuje wyłącznie adres URL i nagłówki zapytania.
* POST – służy do tworzenia nowych danych. Dane te zazwyczaj przesyłane są w ciele zapytania (body).
* PUT – jest metodą podobną do POST jednak służy ona do modyfikacji istniejącego już zasobu.
* DELETE – tak jak wskazuje nazwa służy do usuwania zasobu.

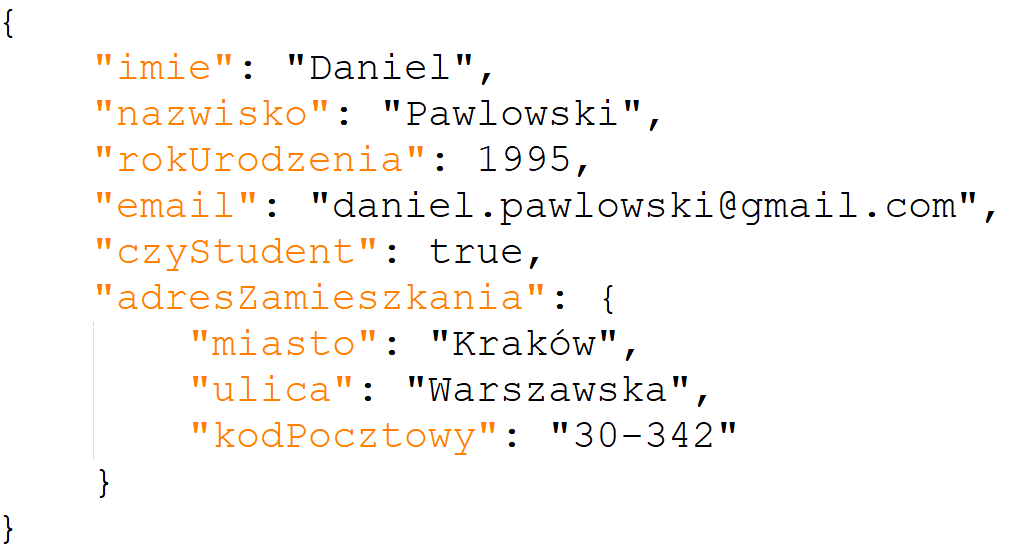
W odpowiedzi na zapytanie http tuż za nazwą i wersją protokołu występuje kod odpowiedzi, który informuje o sposobie realizacji zapytania klienta. Taki kod może należeć do jednej z pięciu grup:

* 1xx – kody informacyjne,
* 2xx – kody powodzenia,
* 3xx – kody przekierowania,
* 4xx – kody błędu po stronie klienta,
* 5xx – kody błędu po stronie serwera.

Aplikacje wykorzystujące architekturę REST powinny być tworzone według ściśle określonych zasad. Zaliczyć do nich można:

* Niezależność interfejsu użytkownika od operacji na serwerze – oznacza to, że klient poprzez swoje zapytanie nie może mieć wpływu na operacje wykonywane po stronie serwera. Serwer natomiast nie ma wpływu na interfejs użytkownika.
* Wszystkie niezbędne informacje muszą zostać zawarte w zapytaniu klienckim – serwer nie powinien przechowywać żadnych informacji o sesji.
* Dane, które wysyła serwer muszą być określone jako stałe (ang. cacheable) lub ulotne (ang. non-cacheable), jest to istotne w przypadku danych, które dość często będą zmieniać swoją wartość.
* Punkty końcowe (ang. endpoints) muszą jednoznacznie wskazywać na zasób do którego się odwołują.
* Warstwy logiki biznesowej i prezentacji muszą być wyraźnie odseparowane od siebie. Warstwy nie powinny bezpośrednio oddziaływać na siebie, a użycie zewnętrznych bibliotek powinno być ukryte.

Dane otrzymywane w odpowiedzi serwera opartego o architekturę REST najczęściej wysyłane są w lekkim i intuicyjnym formacie JSON (JavaScript Object Notation). Wszystkie dane zapisane w formacie JSON są zmiennymi i posiadają wartości. Każda zmienna otoczona jest znakami cudzysłowu , a wartości mogą przyjmować typy string, number, null, false, true lub być tablicą elementów. Dane w tym formacie mogą być w dowolny sposób zagnieżdżane. Przykładowy dane zapisane w formacie JSON zostały ukazane na rysunku 3.12.



**Rys 3.12 Przykładowe dane w formacie JSON**

**3.10 Maven i biblioteki zewnętrzne**



**Rys 3.13 Logo narzędzia Maven**

Maven jest to narzędzie ułatwiające proces tworzenia aplikacji w języku Java. Pozwala on na zarządzanie takimi czynnościami jak budowanie, kompilacja, testowanie, pobieranie zależności czy tworzenie dokumentacji. Konfigurację projektu opartego o narzędzie Maven umieszcza się w pliku pom.xml, w którym to umieszcza się elementy takie jak nazwa i wersja projektu oraz tak zwany groupId czyli identyfikator organizacji lub grupy która stworzyła dany projekt. Poza tym w pliku tym znajdować się będą zależności (ang. dependecies), są to zewnętrzne biblioteki, które są niezbędne do poprawnego zbudowania projektu. Biblioteki włączane są do projektu z lokalnego repozytorium, a jeśli ich tam nie ma pobrane zostaną z zdalnego repozytorium Mavena lub innego wskazanego w konfiguracji. Ostatnim ważnym elementem znajdującym się w pom.xml jest znacznik build, w którym opisane będą informacje na temat procesu budowania. Umieszczone tu mogą być na przykład lokalizacje zasobów, profile budowania użyte wtyczki (ang. plugins) i ich konfiguracja. Do najważniejszych poleceń tego narzędzia zaliczyć można:

* Validate – sprawdzający czy konfiguracja projektu pozwala na jego zbudowanie,
* Compile – wykonuje kompilację projektu jeśli kod zmienił się od ostatniej kompilacji,
* Test – przeprowadza testy jednostkowe,
* Integration-test – przeprowadza testy integracyjne
* Package – tworzy archiwum (.jar lub .war) na podstawie skompilowanego kodu,
* Verify – sprawdza czy utworzone archiwum jest poprawne,
* Install – archiwum zostanie dodane do lokalnego repozytorium i możliwe będzie jego importowanie w innych projektach za pomocą zależności,
* Deploy – przeprowadza wdrożenie,
* Clean – usuwa wszystkie archiwa, które zostały stworzone przez poprzednie budowania,
* Site – generuje dokumentację.

Biblioteki zewnętrzne użyte na potrzeby projektu:

* Jackson Databind – jest to narządzie służące do serializacji obiektów. Może zostać wykorzystane na dwa sposoby:
  + Zamiana obiektu POJO (Plain Old Java Object) na postać tego obiektu w formacie JSON,
  + Zamiana obiektu JSON na obiekt POJO.

Wykorzystywany jest w aplikacji backendowej do komunikacji z innymi komponentami.

* Lombok – jest to narzędzie służące do uproszczenia tworzenia klas przechowujących dane. Bez użycia Lomboka w przypadku stworzenia klasy zawierającej prywatne pola imię i nazwisko należałoby utworzyć mutatory (metody pozwalające na pobieranie i ustawianie wartości obu zmiennych), a do tego konstruktor, metodę porównującą dwa obiekty czy też metodę zamieniającą obiekt na postać tekstową. W przypadku użycia narzędzia Lombok, przy dodaniu odpowiednich adnotacji wszystkie powyższe metody zostaną wygenerowane automatycznie co pozwoli na oszczędność czasu i uszczupli klasę o wiele linii kodu.