A close up of a logo

Description automatically generated

Instytut Automatyki i Robotyki



na kierunku Automatyka, Robotyka i Informatyka Przemysłowa

w specjalności Informatyka Przemysłowa

Aplikacja kliencka Android do serwisu maszyn oraz webserwis

numer pracy według wydziałowej ewidencji prac {liczba}

Mateusz Jaworski

Numer albumu 289702

promotor

dr inż. Paweł Wnuk

konsultacje

{tytuł/stopień naukowy, Imię i Nazwisko}

WARSZAWA 2020

|  |  |
| --- | --- |
| **PRACA DYPLOMOWA inżynierska** | |
| *Specjalność:* Informatyka Przemysłowa | |
| *Instytut prowadzący specjalność:* Instytut Automatyki i Robotyki  *Instytut prowadzący pracę:* Instytut Automatyki i Robotyki | |
| ***Temat pracy:* Aplikacja kliencka Android do serwisu maszyn oraz webserwis** | |
| ***Temat pracy (w jęz. ang.):* Android client application for machine service and web service** | |
| ***Zakres pracy:***   1. Wstęp 2. Przegląd technologii do tworzenia aplikacji mobilnych oraz web serwisów 3. Kompletny opis systemu 4. Implementacja 5. Testy 6. Podsumowanie   ***Podstawowe wymagania:***   1. Podstawowa znajomość webserwisów 2. Umiejętność programowania w języku Java oraz Kotlin | |
| ***Literatura:*** | |
| ***Słowa kluczowe:*** Android, web service, Java, REST, Kotlin | |
| *Praca dyplomowa jest realizowana we współpracy z przemysłem*  *Tak/****Nie****,*  *Nazwa firmy: ……………………………………………………………………………………* | |
| *Imię i nazwisko dyplomanta: Mateusz Jaworski* | *Imię i nazwisko promotora: dr inż. Paweł Wnuk* |
| *Imię i nazwisko konsultanta:* |
| *Temat wydano dnia:* | *Termin ukończenia pracy:* |
| ***Miejsce wykonywania praktyki przeddyplomowej:*** Marcin Harbuz | |
| ***Zatwierdzenie tematu*** | |
|  |  |
| Opiekun specjalności | Z-ca Dyrektora Instytutu |

**Streszczenie**

**Tytuł:** Aplikacja kliencka Android do serwisu maszyn oraz webserwis

**Słowa kluczowe:** Android, web service, Java, REST, Kotlin

Niniejsza praca przedstawia aplikację androidową dla pracowników firmy produkcyjnej. Drugim komponentem jest weberwis, który umożliwia udostępnianie oraz pobieranie z niej danych.

W dzisiejszych czasach jednym z kluczowych czynników rozwoju oraz prosperowania firmy jest czas. Aplikacja będzie posiadać czytnik kodów QR. Użytkownik w prosty sposób będzie mógł zeskanować unikatowy kod danego urządzenia i wysłać go do webserwisu. Nie musi on wpisywać całego ciągu znaków ręcznie. W informacji zwrotnej dostanie dane danej maszyny w celu przeprowadzenia przeglądu, uzyskania informacji, zgłoszenia błędu. Większość urządzeń jest aktualizowana, instrukcje obsługi ulegają zmianie. Kolejną funkcjonalnością będzie pobierania na aplikacje mobilną instrukcji obsługi w formacie PDF. Pozwoli to mieć dostęp do najnowszej wersji tego dokumentu. Pracownicy będą mogli wysyłać raporty z opisem rozwiązań danego problemu. Osoby z kadry zarządzającej mogą wysłać również danemu pracownikowi zadanie, w którym ma on wykonać jej serwis.

W pierwszym rozdziale pracy został przedstawiony jej cel oraz problem jaki ten projekt rozwiązuje. Rozdział „Wprowadzenie” obejmuje przegląd technologii do tworzenia aplikacji mobilnych oraz webserwisów. Zawiera on typy ich, opis oraz uzasadnienie wyboru danej technologii. W rozdziale trzecim opisany jest kompletny system, użyte technologie, komponenty szkieletów do budowy aplikacji. W celu poprawności działania aplikacji w rozdziale „Testy” prezentowane są testy jednostkowe dla webserwisu oraz testy funkcjonalne dla aplikacji mobilnej. Cała praca została zakończona podsumowaniem, które zawiera wnioski, trudności napotkane podczas tworzenia projektu oraz potencjalne ścieżki rozwoju.

**Abstract**

**Title:** Android client application for machine service and web service

**Descriptors:** Android, web service, Java, REST, Kotlin

This work presents an android application for employees of a production company. The second component is web service, which allows sharing and downloading data from it.

Nowadays, one of the key factors for the development and prosperity of a company is time. The application will have a QR code reader. The user will easily be able to scan the unique code of the device and send it to the webservice. He does not have to enter the entire string manually. The feedback will get the data of the machine in order to carry out a review, obtain information, and report a bug. Most devices are updated, the operating instructions are changed. Another feature will be downloading the user manual in PDF format to mobile applications. This will give you access to the latest version of this document. Employees will be able to send reports describing the solutions to a given problem. People from the management team can also send the employee a task in which he is to perform his service.

The first chapter of the paper presents its purpose and the problem that this project solves. The "Introduction" chapter includes a review of technologies for creating mobile applications and web services. It contains their types, description and justification for choosing a given technology. The third chapter describes the complete system, technologies used, skeleton components for building the application. In order to ensure correct operation of the application, the "Tests" chapter presents unit tests for the webservice and functional tests for the mobile application. All the work was completed with a summary, which contains conclusions, difficulties encountered during the creation of the project and potential development paths.



Politechnika Warszawska

………………………..

miejscowość i data

……………………………….

imię i nazwisko studenta

……………………………….

numer albumu

……………………………….

kierunek studiów

Oświadczenie

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

* niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
* niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,
* niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
* wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
* znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

……………………………….

czytelny podpis studenta

**Spis treści**

[1. Wstęp 10](#_Toc45708699)

[1.1. Cel pracy 10](#_Toc45708700)

[2. Wprowadzenie 10](#_Toc45708701)

[2.1. Backend 10](#_Toc45708702)

[2.1.1. Webserwisy 10](#_Toc45708703)

[2.1.2. Przegląd technologii 10](#_Toc45708704)

[2.1.3. Uzasadnienie wyboru 10](#_Toc45708705)

[2.2. Klient 10](#_Toc45708706)

[2.2.1. System operacyjny android 10](#_Toc45708707)

[2.2.2. Przegląd technologii 10](#_Toc45708708)

[2.2.3. Uzasadnienie wyboru 10](#_Toc45708709)

[3. Kompletny opis systemu 10](#_Toc45708710)

[3.1. Backend 10](#_Toc45708711)

[3.1.1. Struktura projektu 10](#_Toc45708712)

[3.1.2. Relacyjna baza danych MySQL 10](#_Toc45708713)

[3.1.3. Mapowanie relacyjno-obiektowe przy użyciu JPA 10](#_Toc45708714)

[3.1.4. Hibernate 10](#_Toc45708715)

[3.1.5. Spring Framework 10](#_Toc45708716)

[3.1.6. Spring Data 10](#_Toc45708717)

[3.1.7. Maven 10](#_Toc45708718)

[3.1.8. Mockito 11](#_Toc45708719)

[3.1.9. Swagger 12](#_Toc45708720)

[3.1.10. JSON Web Token 14](#_Toc45708721)

[3.1.11. Wzorzec projektowy fasady 14](#_Toc45708722)

[3.2. Klient 14](#_Toc45708723)

[3.2.1. Struktura projektu 14](#_Toc45708724)

[3.2.2. Room Persistance Library 14](#_Toc45708725)

[3.2.3. LiveData 14](#_Toc45708726)

[3.2.4. Gradle 14](#_Toc45708727)

[3.2.5. Wzorzec projektowy MVVM (Model-view-viewmodel) 14](#_Toc45708728)

[3.3. Narzędzia i środowiska programistyczne 14](#_Toc45708729)

[3.3.1. IntelliJ IDEA 14](#_Toc45708730)

[3.3.2. Android Studio 14](#_Toc45708731)

[3.3.3. Git 14](#_Toc45708732)

[4. Implementacja 14](#_Toc45708733)

[5. Testy 14](#_Toc45708734)

[5.1. Testy jednostkowe webserwisu 14](#_Toc45708735)

[5.2. Testy funkcjonalne aplikacji mobilnej 14](#_Toc45708736)

[6. Podsumowanie 14](#_Toc45708737)

[6.1. Trudności 14](#_Toc45708738)

[6.2. Potencjalne ścieżki rozwoju 14](#_Toc45708739)

[6.3. Wnioski 14](#_Toc45708740)

[7. Bibliografia 14](#_Toc45708741)

[8. Spis rysunków 14](#_Toc45708742)

[9. Spis tabel 14](#_Toc45708743)

# Wstęp

## Cel pracy

Celem pracy jest wykonanie aplikacji androidowej dla serwisantów maszyn oraz webserwisu, który umożliwi pobieranie i udostępnianie jej danych. Motywacją i wyborem tej pracy jest możliwość rozwiązania danych problemów poprzez niżej wymienione funkcjonalności:

* Używanie czytnika kodów QR do szybkiej identyfikacji maszyny oraz pozyskaniu o niej informacji w celu przeprowadzenia serwisu lub zgłoszenia problemu,
* Pobieranie instrukcji obsługi danego urządzenia pozwoli na stały dostęp do najnowszej wersji aktualizacji, brak konieczności korzystania z papierowych instrukcji,
* Wysyłanie raportów z opisem błędów przez pracowników, dzięki czemu użytkownicy pracujący w innej firmie będą mieli dostęp do nich. Późniejsza możliwość pobrania ich pozwoli zaoszczędzić czas i przyspieszyć rozwiązanie naprawę,
* Zarządzanie zadaniami dla serwisantów. Z poziomu webserwisu będzie można wysyłać konkretne zadania dla danej grupy pracowników bądź konkretnego pracownika. Poprzez to rozwiązanie nie trzeba kontaktować się z każdym indywidualnie,
* Możliwość zgłoszenia błędu wraz z opisem działania aplikacji mobilnej przez pracownika. Przy dalszym rozwoju pozwoli to programiście na produktywniejszą pracę.

# Wprowadzenie

## Backend

### Webserwisy

### Przegląd technologii

### Uzasadnienie wyboru

## Klient

### System operacyjny android

### Przegląd technologii

### Uzasadnienie wyboru

# Kompletny opis systemu

## Backend

### Struktura projektu

### Relacyjna baza danych MySQL

### Mapowanie relacyjno-obiektowe przy użyciu JPA

Korzystając z wybranej relacyjnej bazy danych operacje bazodanowe wykonywane są na wartościach z tabel. Natomiast podczas programowania w Javie programista używa obiekty. W celu usprawnienia oraz automatyzacji powstał mechanizm mapowania tabel na obiekty i odwrotnie (tzw. Object-Relational Mapping).

*„Java Persistence Api (JPA) to specyfikacja opisująca sposób zarządzania relacyjnymi bazami danych”* [9]. Używając JPA programista operuje na encjach, a relacje bazodanowe tworzone są po przez adnotacje (rodzaj interfejsu [10]) oraz pliki konfiguracyjne.

### Hibernate

Hibernate jest to framework implementujący JPA, który pozwala ułatwić i uprościć komunikację z bazą danych. Korzyściami jakie to narzędzie daje są między innymi:

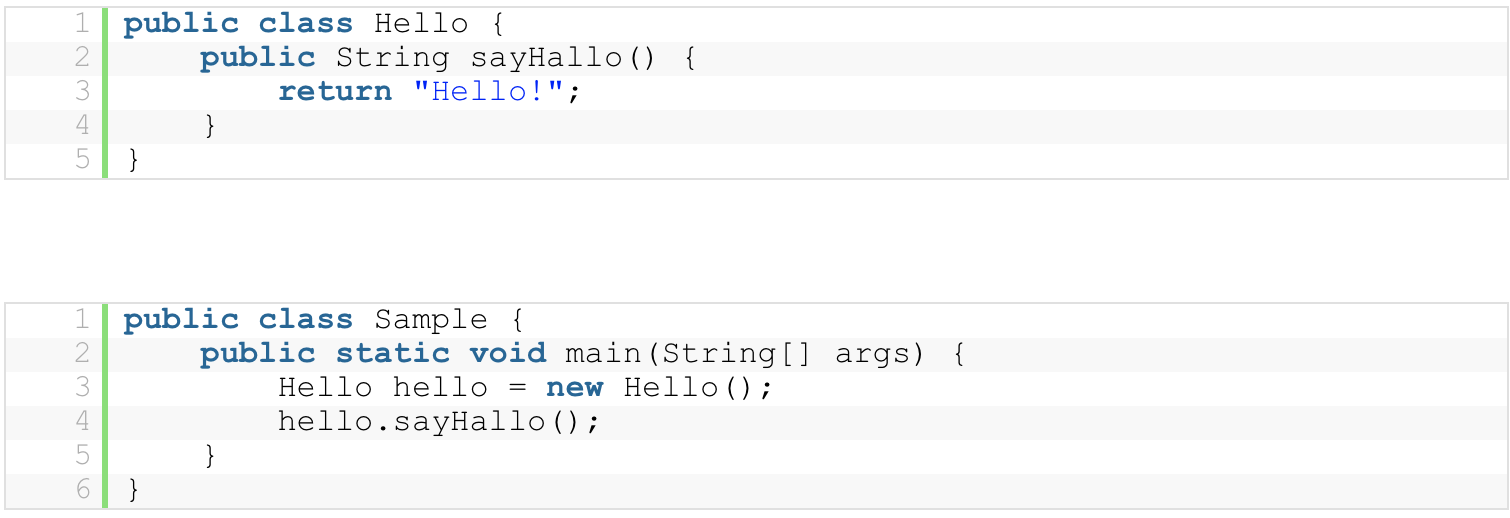
* brak konieczności tworzenia ręcznych zapytań SQL,
* szybszy proces tworzenia aplikacji,
* możliwość wytworzenia struktury bazy danych używając frameworka,
* utrzymanie czystego kodu [11].

### Spring Framework

Na początku Framework - „*jest to szkielet do budowy aplikacji, który dostarcza niezbędne biblioteki i komponenty oraz definiuje strukturę i działanie danej aplikacji*” [16]. Zanim powstał Spring aplikacje tworzone były w Java EE (tzw. Java Enterprise Edition), dzięki niej można było tworzyć aplikacje biznesowe. Zdaniem uznanego programisty jakim jest Rod Johnson to rozwiązanie było zbyt problematyczne, tworzone jest za dużo implementacji oraz kopiowania kodu. Powstał Spring, a jego zaletami są:

* dobra dokumentacja,
* łatwy w testowaniu aplikacji,
* modułowość - istnieją moduły do różnych integracji (np. integracja z bazą danych, pracą w chmurze),
* uniwersalność – używając tego frameworka możemy utworzyć aplikacje mobilne, desktopowe, webowe oraz integrować się z innymi serwisami,
* łatwość przy wrażaniu aplikacji
* odwrócone sterowanie (Inversion Of Control, IoC) - w typowym przypadku programista mówi kiedy tworzyć obiekty np. przez słowo „*new”.* Używając IoC obiekty są tworzone przez narzędzie. W pierwszym kroku uruchamiany jest kontener Springa, po czym odbywa się skanowanie komponentów oraz wstrzykiwanie zależności [17].

Ważnym aspektem jest tzw. wstrzykiwanie zależności (Dependency Injection). Pozwala to automatycznie zarządzać instancjami klas. Używając tego rozwiązania programista nie musi tworzyć klas z użyciem słowa *„new”* oraz nie trzeba przekazywać instancji klasy, aby używać jej w innej klasie [18]. Poniżej przedstawione porównanie:



Rysunek 1. Kod napisany w Java SE

*Źródło [18]*

A screenshot of a social media post

Description automatically generated

Rysunek 2. Kod napisany w Springu

*Źródło [18]*

Kolejne ważne pojęci o którym należy wspomnieć jest tzw. *Bean.* Jest to obiekt zarządzany przez Springa, dzięki czemu programista nie musi tworzyć obiektów ręcznie. W procesie odwróconego sterowania te które są oznaczone adnotacjami są rozpoznawane przez Springa. Poniżej kilka podstawowych adnotacji, które zamieniają klasę, na klasę beana:

* *@Configuration -* wskazuje na aby dana klasa była klasą konfiguracyjną,
* *@Component* -klasa oznaczoną tą adnotacją będzie realizowana w kontekście Springa,
* *@Controller -* klasa odpowiedzialna za warstwę prezentacji (Spring MVC),
* *@RestController -* klasa stanowić będzie interfejs restowy,
* *@Service -* klasa dostarczająca usługę,
* *@Repository* - klasa zarządzająca danymi [19].

A close up of a sign

Description automatically generated

Rysunek 3. Wyspecjalizowane typy adnotacji

*Źródło: [2]*

Jak widać na powyższej grafice są to wyspecjalizowane typy komponentu, które pozwalają rozpoznać do czego dana klasa ma służyć.

### Spring Data

Wraz z kilkunastoletnim rozwojem Springa i wytwarzaniu aplikacji programści zauważyli, że prawie za każdym razem dla danej encji muszą oni tworzyć operacje pobierania, dodawania, edycji, usuwania elementów z bazy. Kod był powtarzalny i bardzo podobny więc powstało Spring Data w celu redukcji zredukowania go [8]. Jest to biblioteka wspomagająca pracę przez automatyczne generowanie kodu repozytoriów. Poniżej przykładowy kod użycia Spring Data.

A picture containing drawing

Description automatically generated

Rysunek 4. Przykład repozytorium z wykorzystaniem Spring Data

*Źródło: Opracowanie własne*

W celu użycia należy użyć zależności *org.springframework.data* (opisane w rozdziale 3.1.7) oraz dodanie adnotacji *@Repository.* Interfejs musi rozszerzać JpaRepository z parametrem:

* Location – nazwa encji danej klasy,
* Long – typ klucza głównego tej klasy.

Programista nie potrzebuje dodatkowych implementacji by mieć dostęp do metod:

* findAll() – zwracająca listę danej encji,
* findById(ID id) – wyszukuje rekord po id,
* save(S entity) – dodanie lub edycja encji S,
* deleteById(ID id) – usuwa rekord po id.

Następnym przydatne użycie pozwala na tworzenie własnych metod wzorując się na konwencji przedstawionej na rysunku poniżej.

A screenshot of a cell phone

Description automatically generated

Rysunek 5. Przykład definiowania metod w repozytorium z wykorzystaniem Spring Data

*Źródło: [9]*

Gdy potrzeba zadać bardziej złożone zapytania do bazy danych istnieje możliwość napisania własnej metody z natywnym zapytaniem SQL [9].

### Maven

Maven jest to narzędzie służące do zarządzania projektami. Jego najważniejszymi zastosowaniami w projekcie to zarządzanie zależnościami oraz kompilacją. Podczas tworzenia aplikacji programista może potrzebować dodatkowych zależności np. Spring, Hibernate, Swagger. Jednym ze sposobów jest pobranie ich ręcznie z danej strony po czym dodanie ich do ścieżki z tworzonym projektem. Istnieje centralne repozytorium, gdzie znajdują się zależności.

A screenshot of a cell phone

Description automatically generated

Rysunek 6. Przykład struktury zależności

*Źródło: [3]*

Używając Maven’a należy wyszukać, umieścić strukturę danych zależności w pliku *pom.xml* (plik konfiguracyjny), po czym pobierane są one automatycznie i dostępne w czasie kompilacji aplikacji [2].

### Mockito

Mockito jest to framework do pisania testów w języku Java. Pisanie testów jest bardzo ważnym i istotnym elementem programowania obiektowego. Podczas tworzenia aplikacji, wraz z dalszym jej rozwojem może, nastąpić potrzeba jej rozbudowy bądź dodania nowej funkcjonalności. Wydając taką wersję programista powinien sprawdzić czy te „stare” funkcjonalności działają poprawnie i czy nie zostały naruszone. Gdyby w aplikacji nie było testów należało by sprawdzić wszystko ręcznie. Jeśli projekt jest rozbudowany było by to bardzo długie i żmudne zadanie, dlatego powinno dążyć się do automatyzacji [4].

Używając Mockito możemy tworzyć obiekty zwane „mokowalnymi”, które są imitacją obiektu. Symulują one zachowanie danego obiektu, klasy czy interfejsu. W momencie mokowania danej klasy nadpisywane są wszystkie metody, które zwracają domyślną wartość. W celu użycia Mockito oprócz zależności *mockito-core* potrzebna będzie również biblioteka JUnit oraz klasa testowa musi być oznaczona adnotacją *@RunWith(MockitoJUnitRunner.class).*

*A screenshot of a social media post

Description automatically generated*

Rysunek 7. Przykładowy projekt szablonowy testu

*Źródło: [5]*

W powyższym szablonie można wyróżnić poszczególne sekcje:

* *@Mock -* adnotacja pozwala na wstrzyknięcie obiektu który programista będzie imitować,
* *@Before -* metoda oznaczoną tą adnotacją będzie wykonywać się za każdym razem wewnątrz tej klasy,
* *@Test -* zamieszczana jest logika testująca,
* *@After -* metoda oznaczoną tą adnotacją będzie wykonywać się za każdym razem po skończonym teście [5].

Warto również wspomnieć o dwóch adnotacjach, które są wykorzystywane podczas tworzenia testów:

* *@InjectMocks*- stosowane jest jeśli w projekcie jest klasa agregująca w sobie jakieś elementy. W celu dostępu do nich używana jest ta adnotacja. Np. kontroler posiada w konstruktorze serwis, jeśli będzie on zmokowany to używając *@InjectMocks* można wstrzyknąć do kontrolera ten serwis,
* *@Spy-* spy różni się od mocka tym, że zachowuje się tak samo jak obiekt [6]

W rozdziale 5.1 zostały omówione oraz przeprowadzone testy aplikacji.

### Swagger

Swagger jest to biblioteka, zestaw narzędzi dzięki którym programiści mogą w łatwy sposób dokumentować oraz używać REST API. Swagger UI umożliwia wizualizację jego zasobów z nakładką graficzną. Używając go należy pobrać dwie zależności z repozytorium Mavena: *springfox-swagger2* i *springfox-swagger-ui.* Druga zależność nie jest jednak konieczna, bez niej dokumentacja wygeneruje się w postaci JSON. W celu uruchomienia biblioteki w klasie konfiguracyjnej Springa konieczne jest dodanie adnotacji *@EnableSwagger2.* Ostatnim krokiem jest sprawdzenie dokumentacji pod adresem *localhost:8080/swagger-ui.*html (domyślny adres) [1]. Dokumentacja jest generowana automatycznie po przebudowaniu projektu. Osoba korzystająca z API może sprawdzić oraz przetestować przypadki użycia endpointów oraz modeli klas. Sprawia to, że Swagger jest to niezwykle pomocne narzędzie dla programistów frontend.

A screenshot of a cell phone

Description automatically generated

Rysunek 8. Dokumentacja API z wykorzystaniem Swagger UI

*Źródło: [1]*

### Autoryzacja z JSON Web Token

Autoryzacja jest procesem polegającym na kontroli dostępu do danego źródła. Określa jaki użytkownik ma uprawnienia do danych zasobów i co może z nimi robić [12] (np. administrator posiada inną rolę niż pracownik). JSON Web Token to ciąg znaków, zakodowany JSON w przy użyciu base64 (rodzaj kodowania [13]). Budowa tokena składa się z następujących elementów:

* Header - zapisane są w nim informacje o rodzaju tokena i algorytmu, którym został on zaszyfrowany,
* Payload - zapisane są tu dowolne dane (np. rola użytkownika, czas życia tokena),
* Verify - suma kontrolna dwóch wyżej wymienionych komponentów [14].

A picture containing device

Description automatically generated

Rysunek 9. Elementy JWT

*Źródło: [14]*

Autoryzacja przy użyciu JWT zachodzi pomiędzy klientem a serwerem:

A screenshot of a cell phone

Description automatically generated

Rysunek 10. Komunikacja JWT

*Źródło: [15]*

Klient podczas procesu logowania wysyła zapytanie z danymi użytkownika do serwera, po czym następuje proces uwierzytelnienia. Jako rezultat zwrócony zostaje token, który użytkownik musi przechowywać. W celu dostępu do danych zasobów jako nagłówek zapytania musi on go wysyłać, w innym przypadku dostęp będzie zabroniony. Po upływie ustawionego czasu token przestaje być ważnym co wiążę się z utratą dostępu. W zależności od wymagań projektu proces należy powtórzyć lub odświeżyć przed upływem czasu życia.

### Wzorzec projektowy fasady

Budując duże aplikacje programiści powinni projektować je w taki sposób, żeby w przyszłości były łatwe w rozbudowie oraz utrzymaniu. Jednym z rozwiązań jak sprostać temu problemowi są tak zwane wzorce projektowe. Wzorce projektowe to: „*zbiór najlepszych praktyk z gotowymi rozwiązaniami dla wybranych problemów napotykanych w trakcie projektowania rozwiązań zorientowanych obiektowo*” [7]. Dzielą się one na kilka kategorii, jedną z nich są wzorce strukturalne do których należy wzorzec fasady. Pośrednia warstwa serwisu jest przeznaczona dla niestandardowej logiki biznesowej oraz pobierania danych z różnych źródeł.

A close up of a computer

Description automatically generated

Rysunek 11. Przykład zastosowania wzorca projektowego fasady

*Źródło: [2]*

Jeśli programista potrzebuje korzystać z danych z różnych lokalizacji np.:

* Baza danych I – podstawowe informacje o użytkowniku (przez Customer Dao),
* Baza danych II – informacje o kupionych produktach przez klientów (przez Sales DAO),
* Baza danych III – licencje na kupione produkty (przez License DAO).

Używając takiej struktury poprzez warstwę serwisu twórca aplikacji może przekazać łączne zintegrowane razem dane [2].

Stosując wzorce projektowe zespół programistów może czerpać korzyści związane z dobrymi praktykami programowania, utrzymania czystego kodu oraz lepszej komunikacji w zespole.

## Klient

### Struktura projektu

### Room Persistance Library

### Dagger

### Retrofit

### LiveData

### Gradle

### Wzorzec projektowy MVVM (Model-view-viewmodel)

## Narzędzia i środowiska programistyczne

### IntelliJ IDEA

### Android Studio

### Git

# Implementacja

# Testy

## Testy jednostkowe webserwisu

## Testy funkcjonalne aplikacji mobilnej

# Podsumowanie

## Trudności

## Potencjalne ścieżki rozwoju

## Wnioski

# Bibliografia

[1] <https://bykowski.pl/swagger-ui-przejrzysta-wizualizacja-zasobow-api/>

[2] <https://www.udemy.com/course/spring-hibernate-tutorial/>

[3] <https://mvnrepository.com/artifact/org.hibernate/hibernate-core/5.4.18.Final>

[4] <https://www.youtube.com/watch?v=n2SWC-yMg6o>

[5] <https://bykowski.pl/mockito-wprowadzenie-i-przyklady/>

[6] <https://www.youtube.com/watch?v=J4Tad0UCPFc>

[7] <https://bykowski.pl/wzorce-projektowe/>

[8] <https://www.youtube.com/watch?v=WIMCuNYfdE4>

[9] <https://nullpointerexception.pl/spring-data-jpa-wprowadzenie/>

[10] <https://www.samouczekprogramisty.pl/adnotacje-w-jezyku-java/>

[11] <https://bykowski.pl/czym-jest-hibernate/>

[12] <https://mfiles.pl/pl/index.php/Autoryzacja>

[13] <https://www.base64decode.org>

[14] <https://bykowski.pl/json-web-token-w-spring-boot/>

[15] <https://nullpointerexception.pl/spring-security-i-json-web-token/>

[16] <https://marketingwsieci.pl/slownik-e-marketingu/framework/>

[17] <https://bykowski.pl/8-powodow-dla-ktorych-warto-uczyc-sie-spring-framework/>

[18] <https://bykowski.pl/spring-boot-czesc-2-adnotacje-obiekty-zarzadzane/>

[19] <https://bykowski.pl/tworzenie-beanow-w-spring/>

[20]

# Spis rysunków

[Rysunek 1. Kod napisany w Java SE 11](#_Toc47867897)

[Rysunek 2. Kod napisany w Springu 12](#_Toc47867898)

[Rysunek 3. Wyspecjalizowane typy adnotacji 12](#_Toc47867899)

[Rysunek 4. Przykład repozytorium z wykorzystaniem Spring Data 13](#_Toc47867900)

[Rysunek 5. Przykład definiowania metod w repozytorium z wykorzystaniem Spring Data 13](#_Toc47867901)

[Rysunek 6. Przykład struktury zależności 14](#_Toc47867902)

[Rysunek 7. Przykładowy projekt szablonowy testu 15](#_Toc47867903)

[Rysunek 8. Dokumentacja API z wykorzystaniem Swagger UI 16](#_Toc47867904)

[Rysunek 9. Elementy JWT 17](#_Toc47867905)

[Rysunek 10. Komunikacja JWT 17](#_Toc47867906)

[Rysunek 11. Przykład zastosowania wzorca projektowego fasady 18](#_Toc47867907)

# Spis tabel