Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej
i Systemów Informacyjno-Pomiarowych
Zakład Elektrotechniki Teoretycznej
i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Inżynieria oprogramowania

Implementacja portalu do grywalizacji z użyciem platformy SPRING i bazy danych Oracle

Jacek Kozieja
nr albumu 261053

promotor
dr inż. Jacek Korytkowski

WARSZAWA 2017

Implementacja portalu do grywalizacji z użyciem platformy SPRING i bazy danych Oracle

Streszczenie

Praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy, trzech rozdziałów (2-4) zawierających opis istniejących podobnych rozwiązań, komponentów rozpatrywanychjako kandydaci do tworzonego systemu i wreszcie zagadnień wydajności wirtualnych rozwiązań. Piąty rozdział to opis środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne. Ostatni rozdział pracy to opis możliwości dalszego rozwoju projektu.

Słowa kluczowe: Spring, Angular, Oracle

Implementation of gamification web portal based on SPRING framework and Oracle databse

Abstract

This thesis presents a novel way of using a novel algorithm to solve complex problems of filter design. In the first chapter the fundamentals of filter design are presented. The second chapter describes an original algorithm invented by the authors. Is is based on evolution strategy, but uses an original method of filter description similar to artificial neural network. In the third chapter the implementation of the algorithm in C programming language is presented. The fifth chapter contains results of tests which prove high efficiency and enormous accuracy of the program. Finally some posibilities of further development of the invented algoriths are proposed.

Keywords: Spring, Angular, Oracle

WARSZAWA, 16 maja 2017

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa inżynierska pt. Implementacja portalu do grywalizacji z użyciem platformy SPRING i bazy danych Oracle:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Jacek Kozieja.....

Spis treści

Podziękowania

Dziękuję serdecznie Panu dr. inż. Jackowi Korytkowskiemu za pomoc w przygotowaniu pracy. Dziękuję także moim wspaniałym rodzicom Bożenie i Robertowi Koziejom, dzięki którym miałem możliwość kształcić się i zdobywac cenną wiedzę.

Jacek Kozieja

Rozdział 1

Wstęp

1.1 Grywalizacja

Grywalizacja, gryfikacja lub gamifikacja, to zgodnie z definicją [Wikipedia] "Wykorzystanie mechaniki znanej np. z gier fabularnych i komputerowych, do modyfikowania zachowań ludzi w sytuacjach niebędących grami, w celu zwiększenia zaangażowania ludzi."Ta szeroka definicja obejmuje rozwiązania stosowane w marketingu, zarządzaniu projektami jak i edukacji. Założeniem stworzonej aplikacji jest motywowanie grup ludzi do wspólnej pracy poprzez system wyzwań i nagród. Należy jednak pamiętać że o ile aplikacja pomaga zarządzać zadaniami, prowadzić ranking i kontrolować rozwój użytkowników, o tyle weryfikacja wykonanych zadań czy przyznawanie fizycznych nagród pozostaje obowiązkiem administratora. Stworzone rozwiązanie było także pretekstem do zbadania współczesnych technologii wytwarzania aplikacji internetowych, opisanych w dalszej części pracy.

1.2 Projekt

Ideę działania aplikacji „Gamify" przybliżyć mogą poniższe przykładowe Przypadki Użycia:
UC Użytkownika:
Wykonanie Zadania

1. Użytkownik loguje się do Systemu.
2. Użytkownik wybiera opcję „Wykonaj Zadanie" -> (lub skanuje QR)
3. System wyświetla formularz wykonania Zadania.
4. Użytkownik wpisuje numer Zadania.

5. System dodaje zadanie i punkty Użytkownikowi

Przejrzenie tabeli Rankingu

1. Użytkownik loguje się do Systemu.
2. Użytkownik wybiera opcję „Ranking".
3. System wyświetla tabelę Użytkowników.
4. Użytkownik wybiera sortowanie malejąco, po Punktach.
5. System wyświela posortowaną listę.

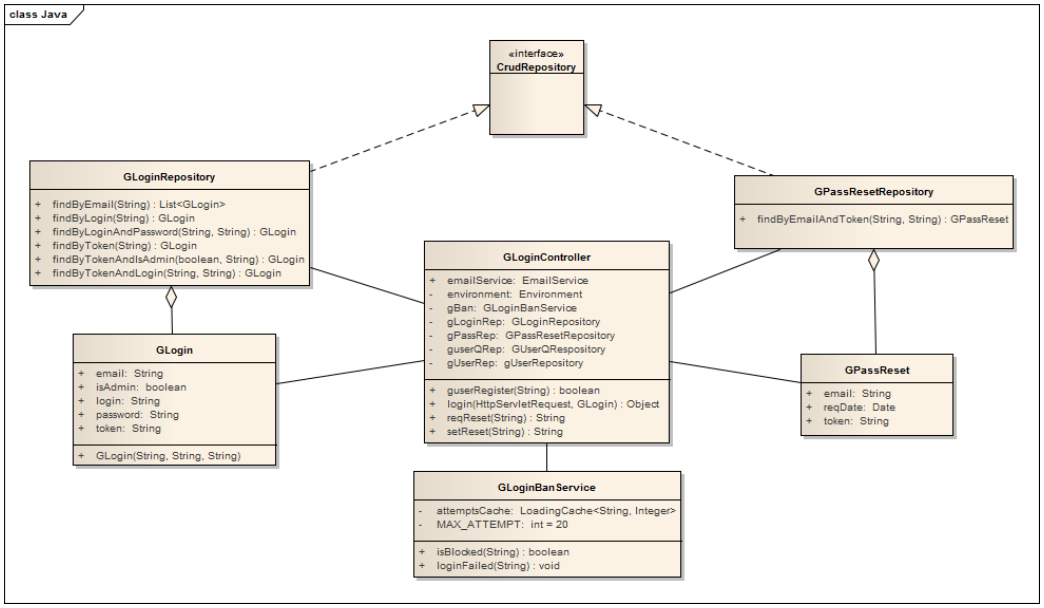
UC Administratora:
Dodanie nowego Zadania

1. Administrator loguje się do systemu
2. Administrator wybiera opcję „Dodaj Zadanie"
3. System wyświetla formularz dodania Zadania.
4. Administrator podaje opis, datę końcową i punkty Zadania.
5. System generuje numer Zadania i zapisuje je.

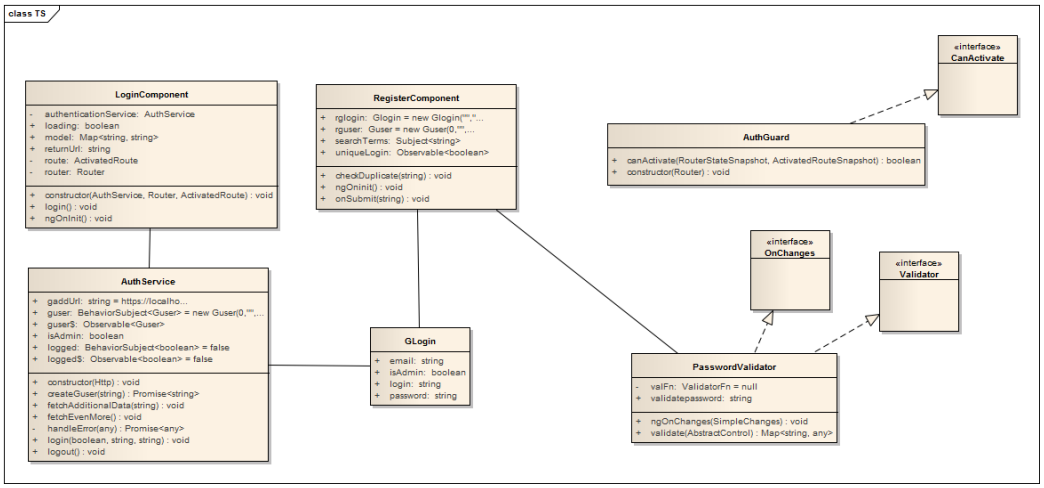
Przynanie Odznaki Użytkownikom

1. Administrator loguje się do Systemu.
2. Administrator wybiera opcję „Odznaki".
3. System wyświetla okno edycji Odznak.
4. Administrator wybiera opcję „Przyznaj odznakę"
5. System wybiera okno wyboru odznaki.
6. Administrator wybiera Odznakę.
7. System wyświetla okno wyboru Użytkowników.
8. Administrator wybiera UżytkownikóW.
9. System przyznaje odznakę wybranym Użytkownikom.

(Jak starczy czasu na koniec to może też diagram UC) Poniższe diagramy klas przedstawiają strukturę modułu GLogin - oddzielnie dla wnętrza („back-end") i fasady („front-end"). Są one reprezentatywne dla całości projektu, zaś stworzenie pełnej dokumentacji uważam za odpowiedni temat dla oddzielnej pracy dyplomowej. Prywatne zmienne posiadające publiczne metody dostępowe zostały przedstawione jako zmienne publiczne. Redukuje to liczbę metod i zwiększa czytelność diagramów.



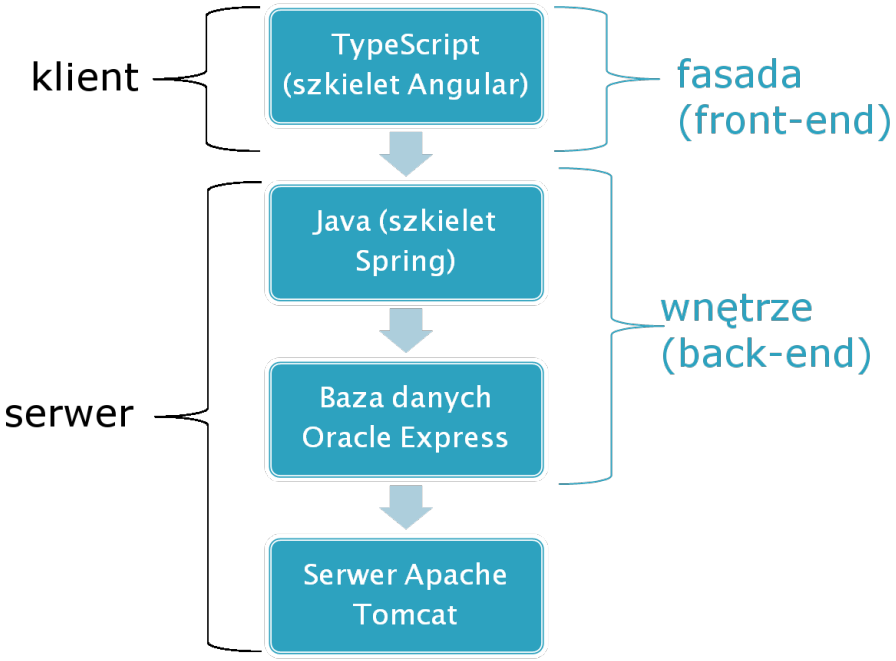
Rysunek 1.1: Diagram klas w języku Java.



Rysunek 1.2: Diagram klas w języku TypeScript.

1.3 Użyte technologie

Do wykonania aplikacji użyto poniższych technologii:



Rysunek 1.3: Języki, szkielety i systemy bedące częścią projektu.

1.4 Wzorzec MVC

MVC to wzorzec architektoniczny służący do organizowania struktury systemów interaktywnych. Składa się on z trzech części:

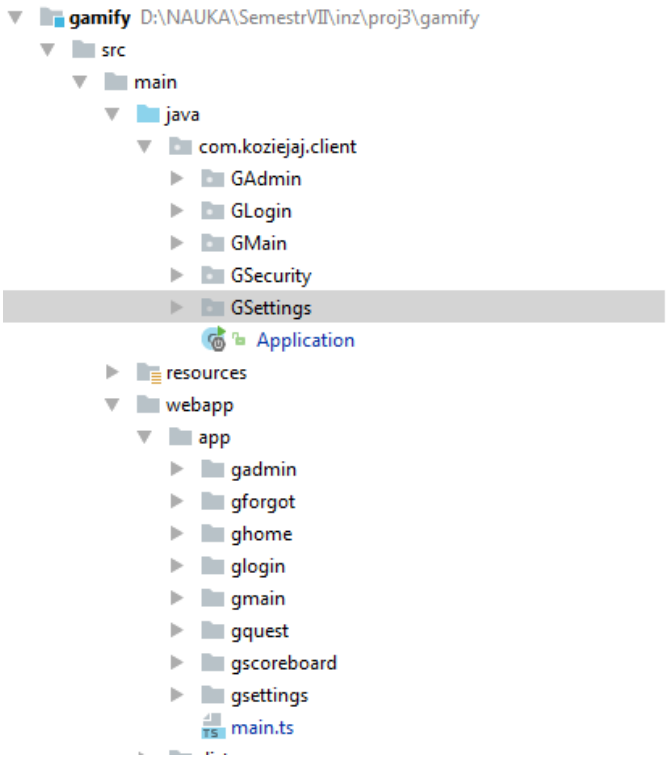
1. Model - reprezentuje dane i wykonuje logikę biznesową.
2. Widok - wyświetla dane pobrane z Modelu.
3. Kontroler - reaguje na dane wejściowe użytkownika. Przesyła żądania wykonania logiki biznesowej do Modelu i zmian Widoku.

Zarówno Angular jak i Spring samodzielnie realizują wzorzec MVC. Jednak gdy połączymy te dwie technologie, sytuacja zmienia się. Warstwa kontrolera przeniesiona jest do Angulara i jest wykonywana przez przeglądarkę. To samo samo dzieje się z warstwą widoku. Po stronie serwerowej Springa

pozostaje warstwa modelu. Ten rozdział powoduje że potrzebny jest odpowiedni sposób komunikacji klient-serwer. Nazywa się on REST API. Skróty te oznaczają Representational State Transfer i Application Programming Interface. Pierwszy skrót oznacza bezstanową wymianę tekstowych zasobów sieciowych. Przykładem są tu zapytania HTTP GET, POST, PUT, DELETE. Drugi skrót oznacza jednoznacznie zdefiniowany sposób komunikacji między komponentami. W wypadku aplikacji internetowej oznacza to że strona serwerowa odbierając zapytanie HTTP pod danym adresem, powinna zwrócić odpowiedź w postaci XML lub JSON. Modułowa struktura projektu przed-

Rysunek 1.4: Zapytanie do API i odpowiedź. Taka komunikacja pozwala wykorzystać tą samą warstwę modelu w kilku różnych aplikacjach.

stawia się następująco: wewnątrz projektu Springa rezydują obok siebie 3 foldery - z plikami źródłowymi Javy, plikami statycznymi (grafiką) i projektem Angulara. Obydwa języki posiadają moduły główne - tam przechowywane są konfiguracje i najczęściej używane klasy. Każda większa funkcjonalność posiada oddzielny folder. To samo tyczy się aplikacji admina która jest traktowana jako oddzielny moduł. Przedrostek g występujący w nazwach klas i modułów podyktowany jest zbieżnoscią nazw takich jak login czy user z natywnymi elementami użytych platform.



Rysunek 1.5: Modułowy podział projektu.

Rozdział 2

Spring

Spring to szkielet tworzenia aplikacji (ang. framework) przeznaczony dla Javy Enterprise Edition, czyli serwerowej odmiany tego języka. Powstał jako alternatywa dla oficjalnych standardów serwerowych Javy takich jak Enterprise JavaBeans. Spring nie narzuca jednego modelu programowania. Posiada wiele modułów które są przydatne przy pisaniu aplikacji, jednocześnie będąc całkowicie opcjonalnymi. Znaną cechą Springa jest **„odwrócenie sterowania”** (ang. Inversion of Control), czyli wzorzec w którym wykonywaniem programu steruje szkielet a nie kod programisty. To Spring decyduje kiedy wykonać dane akcje. Przykładem odwróconego sterowania jest chociażby **"wstrzykiwnie zależności"** (ang. Dependency Injection), kiedy to utworzone, zainicjowane obiekty przekazywane są obiektom które ich potrzebują. Obiekty nie tworzą wtedy samodzielnie instancji innych obiektów. Kolejną ważną cechą Springa jest **„programowanie aspektowe”** czyli wzorzec w którym rozdziela się fizycznie funkcjonalności i definiuje punkty komunikacji między nimi, dostępne wszędzie tam gdzie są potrzebne.

2.1 Metody konfiguracji

Konfiguracja projektu w Springu może odbywać się na dwa sposoby - przy pomocy plików XML lub adnotacji. Poniższy przykład prezentuje te same ustawienia w obydwu reprezentacjach:

XML	Adnotacje

<beans>
<bean id = ”myClass”
class = ”com.koziejaj.MyClass” />
</beans>
	@Configuration
	public class MyClassConfig {
	@Bean
	public MyClass myClass(){
	return new MyClass(); } }

Konfiguracja XML odbywa się w oddzielnych plikach, zaś adnotacjami - bezpośrednio w kodzie którego dotyczy. Warto wiedzieć że konfiguracja adnotacjami wykonuje się przed XML. W projekcie użyto adnotacji. Wydają się one czytelniejsze, są też kierunkiem w którym zmierza rozwój Springa. Nie bez powodu był on kiedyś określany kolokwialnie jako "XML hell". Wyjątkiem są pliki z rozszerzeniem .properties. Przechowują one stałe wartości, takie jak numer portu serwera, dane uwierzytelniające bazy danych.

2.2 Spring Boot

Brak narzuconych standardów szkieletu aplikacji ma także i swoje wady. Spring wymaga dużej ilości konfiguracji, a także dobrania modułów i bibliotek które będą nam potrzebne. Moduł Spring Boot zawiera wstępną konfigurację i najpotrzebniejsze biblioteki. Jednocześnie umożliwia dostosowanie projektu do specyficznych wymagań. Podczas uruchomienia sprawdza których elementów konfiguracji brakuje i dodaje je zgodnie z posiadaną wiedzą na temat projektu. Moduł ten można uznać za fundament stworzonej aplikacji. Ilość kodu wymaganą do napisania aplikacji serwerowej w ramach Spring Boot przedstawiają poniższe listingi:

```
package hello;

import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMapping;

@RestController
public class HelloController {

    @RequestMapping("/")
    public String index() {
    return "Witaj Swiecie! Pozdrowienia od Spring Boot!\n";
    }
}
```

Adnotacja @RestController informuje moduł Spring MVC że klasa będzie przystosowana do ubsługi zapytań sieciowych. Adnotacja @RequestMapping(”/”) oznacza ścieżkę która spowoduje wywołanie metody.

```
package hello;

import java.util.Arrays;
```

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Bean
    public CommandLineRunner commandLineRunner(ApplicationContext
        ctx) {
        return args -> {

            System.out.println("Klasy Projektu:");

            String[] beanNames = ctx.getBeanDefinitionNames();
            Arrays.sort(beanNames);
            for (String beanName : beanNames) {
                System.out.println(beanName);
            }

        };
    }

}
```

Adnotacja @SpringBootApplication jest wygodnym połączeniem kilku innych adnotacji:

- @Configuration - oznacza klasę jako źródło metod z adnotacją @Bean, które powinne być dostępne w kontekście aplikacji.
- @EnableAutoConfiguration - nakazuje Spring Boot dodać do kontekstu aplikacji wszystkie metody @Bean na podstawie ścieżki projektu, innych metod @Bean i dostępnych konfiguracji.
- @ComponentScan - skanuje pakiet (w tym przypadku „hello") w poszukiwaniu komponentów, konfiguracji i serwisów.

Metoda SpringApplication.run(); uruchamia aplikację, zaś metoda zwracająca CommandLineRunner jest uruchamiana podczas startu.

```
dServletContainer : Tomcat started on port(s): 8080 (http)
2017-05-19 14:19:48.415  INFO 6348 --- [           main] hello.Application
                  : Started Application in 6.988 seconds (JVM running for 7.689)

Klasy Projektu:
application
basicErrorController
beanNameHandlerMapping
beanNameViewResolver
characterEncodingFilter
conventionErrorViewResolver
defaultServletHandlerMapping
defaultValidator
defaultViewResolver
dispatcherServlet
dispatcherServletRegistration
duplicateServerPropertiesDetector
embeddedServletContainerCustomizerBeanPostProcessor
error
```

Rysunek 2.1: Fragment uruchomienia skompilowanego przykładu.

```
PS D:\NAUKA\SemestrVII\inz\spring-boot-quickstart\gs-spring-boot\initial> curl l
ocalhost:8080
Witaj Swiecie! Pozdrowienia od Spring Boot!
PS D:\NAUKA\SemestrVII\inz\spring-boot-quickstart\gs-spring-boot\initial>
```

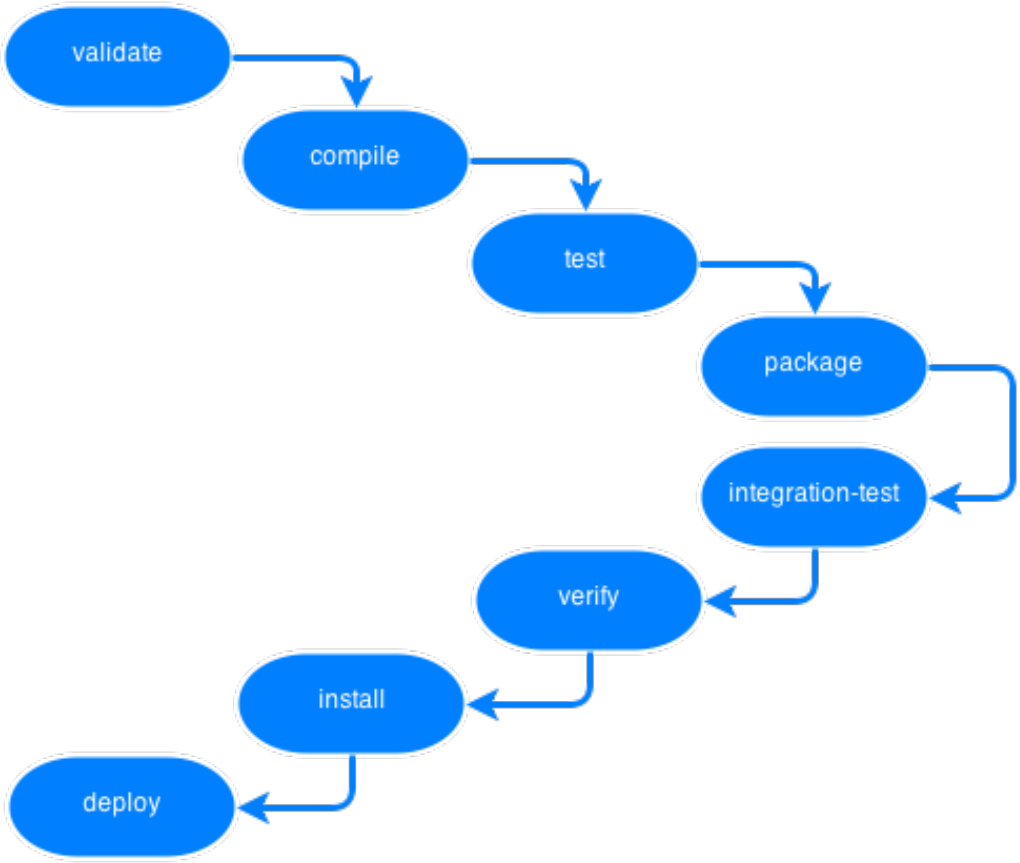
Rysunek 2.2: Zapytanie wysłane do przykładu.

2.3 Maven

Maven to w narzędzie do zarządzania projektem - jego zależnościami (używanymi bibliotekami) i strukturą (podziałem na moduły). Maven automatyzuje budowę projektu zgodnie z poniższymi krokami.

1. Validate - walidacja kodu. Sprawdzana jest poprawność kodu źródłowego, umożliwiająca jego kompilację.
2. Compile - kompilacja, czyli przetworzenie kodu źródłowego na kod bajtowy, wykonywany przez maszynę wirtualną Javy.
3. Test - wykonanie testów jednostkowych.
4. Package - budowa pojedynczej paczki wykonywalnej (plik .JAR) lub wykonywanej przez serwer (plik .WAR).
5. Integration-test - wykonanie testów integracyjnych.
6. Verify - weryfikacja jakościowa paczki zbudowanej w kroku 4.

7. Install - instalacja paczki. Oznacza to umieszczenie paczki z kroku 4 w repozytorium lokalnym lub zdalnym.
8. Deploy - umieszczenie projektu w repozytorium.



Rysunek 2.3: Uproszczony cykl życia projektu Maven.

Konfiguracja Maven znajduje sie pliku POM.xml. Poniższy przykład przedstawia skróconą wersję pliku POM stworzonego projektu:

```
<!-- Nagłówek określający używaną wersję -->
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http
    ://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
    apache.org/maven-v4_0_0.xsd">

<!-- Podstawowe informacje na temat projektu -->
```

```
<modelVersion>4.0.0</modelVersion>
<groupId>com.koziejaj.client</groupId>
<artifactId>gamify</artifactId>
<packaging>jar</packaging>
<version>0.1-dev</version>
<name>gamify</name>
<url>http://maven.apache.org</url>

<!-- Dziedziczenie po istniejącaym projekcie -->
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.4.2.RELEASE</version>
</parent>

<!-- Zależności - lista bibliotek które należy pobrać -->
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
</dependencies>

<!-- Dodatkowe ustawienia -->
<properties>
<java.version>1.8</java.version>
</properties>

</project>
```

2.4 Przydatne moduły

Oto lista najważniejszych modułów użytych w projekcie:

- Spring MVC - przy pomocy klasy DispatcherServlet przekazuje zapytania do odpowiednich kontrolerów, tak jak na listingu 2.2. W przypadku

wykonanego projektu, jest częścią modułu spring-boot-starter-web.

- Spring Data - rozbudowana, wielomodułowa biblioteka, wspierająca warstwę dostępu do danych. Wypiera popularną niegdyś bibliotekę Hibernate. W przypadku Gamify, użyto dwóch modułów:
  - JPA - służy do mapowania obiektowo-relacyjnego zgodnie ze standardem JPA. Pobrany jako moduł spring-boot-starter-data-jpa.
  - REST - Umożliwia tworzenie repozytoriów z automatycznie generowanymi metodami CRUD. Obiekt takiego repozytorium odpowiada tabeli w bazie danych. Mogą być one z łatwością rozbudowane o szczegółowe metody obróbki danych. Przykładowo, metoda

    ```
    GLogin findByLoginAndPassword(String login, String
        password);
    ```

    wyszuka użytkowników o określonym loginie i haśle bez potrzeby pisania jakiejkolwiek logiki biznesowej. REST umożliwia wysyłanie zapytań do repozytoriów w sposób analogiczny do kontrolerów 2.2, jednak ta funkcjonalność nie została wykorzystana w projekcie. Moduł pobrany jako: spring-boot-starter-data-rest.
- Spring Security - biblioteka zapewniająca bezpieczeństwo aplikacji, szerzej opisana w rozdziale Bezpieczeństwo 5.

2.5 Implementacja REST API

Implementacja serwerowej części aplikacji zostanie omówiona na przykładzie modułu panelu Administratora. Ograniczy to analizę do sześciu klas, zorganizowanych podobnie do wcześniejszego diagramu panelu logowania 1.1.

2.5.1 Mapowanie obiektowo - relacyjne

Klasy omówione w tej sekcji są definicjami tabel bazy danych. Klasa taka odpowiada pojedynczemu rekordowi tabeli i pozwala na jego modyfikację. Jedną z funkcjonalności panelu Administratora jest tworzenie i przyznawanie Odznak. Jest to taki wirtualny medal za specjalne zasługi, który Administrator może nadać Użytkownikowi. Instancja takiego medalu musi zawierać dwie informacje - unikatowy login Użytkownika który go posiada i nazwę Odznaki. Nazwa ta odpowiada plikowi graficznemu Odznaki w plikach statycznych projektu.

```
package com.koziejaj.client.GAdmin;


import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.IdClass;

//Adnotacja oznaczająca klasę odpowiadającą tabeli bazy danych
@Entity
//Adnotacja wczytująca definicję klucza głównego (primary key) - wyjaś
    nienie w kolejnym listingu
@IdClass(GBId.class)
public class GBadge{

    //Adnotacje klucza głównego
    @Id
    private String login;
    @Id
    private String badge;

    //Konstruktory
    public GBadge() {
    }

    public GBadge(String i, String v) {
    login = i;
    badge = v;
    }

    //Metody dostępowe
    public String getLogin() {
    return login;
    }

    public void setLogin(String login) {
    this.login = login;
    }

    public String getBadge() {
    return badge;
```

```
    }

    public void setBadge(String badge) {
    this.badge = badge;
    }
}
```

Chcąc wykorzystywać klucz główny złożony z kilku kolumn, należy zdefiniować specjalną klasę. Taki klucz gwarantuje że dany Użytkownik posiada co najwyżej jedną instancję danej odznaki.

```
package com.koziejaj.client.GAdmin;

import java.io. Serializable ;

//klasa klucza glownego implementuję serializację, co pozwala szkieletowi
      aplikacji porównywać ich wartości zapisane jako strumień bajtów.
public class GBId implements Serializable {

    private String login;
    private String badge;

    public GBId(){}

    public String getLogin() {
    return login;
    }

    public void setLogin(String login) {
    this.login = login;
    }

    public String getBadge() {
    return badge;
    }

    public void setBadge(String badge) {
    this.badge = badge;
    }
}
```

Administrator może zmieniać różne parametry wpływające na aplikację Użytkownika. Jest to funkcjonalność bardzo prostego Ŝystemu zarządzania treścią"(ang. CMS). Na ten moment obsługiwane wartości to: zawartość strony powitalnej, słowa oznaczające Punkty i Poziomy w aplikacji Użytkownika, arkusz css aplikacji, wzór na progi punktowe kolejnych Poziomów. Dodanie kolejnej funkcjonalności tego typu wymagałoby dodania rekordu w tabeli i zmodyfikowania aplikacji Użytkownika. Do przechowywania zmian potrzebna jest bardzo prosta klasa:

```
package com.koziejaj.client.GAdmin;

import javax.persistence.Entity;
import javax.persistence.Id;
@Entity
public class GLayout {

@Id
private String id;
private String value;

public GLayout() {}

    public GLayout(String i, String v) {
    id = i;
    value = v;
    }

    public String getId() {
    return id;
    }

    public void setId(String id) {
    this.id = id;
    }

    public String getValue() {
    return value;
    }

    public void setValue(String value) {
    this.value = value;
```

```
    }

    //Metoda ToString przesłonięta by ułatwić debugowanie aplikacji.
    @Override
    public String toString() {
    return String.format(
    "GLayout[id=%s, value=%s]",
    id,value);
    }

}
```

2.5.2 Repozytoria

Zarówno klasa GBadge jak GLayout mają swoje repozytoria. Można traktować je jako instancje tabel bazy danych. To w repozytoriach szuka się rekordów i zapisuje zmiany na stałe.

```
package com.koziejaj.client.GAdmin;

import org.springframework.data.repository.CrudRepository;
import java.util.List;

import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.
    RepositoryRestResource;
import org.springframework.transaction.annotation.Transactional;

@RepositoryRestResource(collectionResourceRel = "GBadges", path = "
    GBadges")
public interface GBadgeRepository extends CrudRepository<GBadge,
    String> {
List<GBadge> findByLogin( String login);
@Transactional
List<GBadge> removeByBadge(@Param("badge") String badge);
}
```

```
package com.koziejaj.client.GAdmin;

import org.springframework.data.repository.CrudRepository;
import java.util.List;
```

```
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.
    RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel = "GUsers", path = "
    GUsers")
public interface GLayoutRepository extends CrudRepository<GLayout,
     String> {


}
```

2.5.3 Kontroler

-framework ogólnego przeznaczenia
-Spring boot - co to?
- konfiguracja xml vs adnotacje
- maven(wklej i omów prosty przykład, napisz że mój jest bardziej pro, no ale)
- przydatne moduły Springa - MVC, Data, JPA, Security, Jackson
- moje REST API - klient
- moje REST API - admin

Rozdział 3

Baza danych Oracle

-Krótka historia baz oracla
- Konfiguracja bazy w Springu
-Rysunek gdzie są WSZYSTKIE tabele i powiązania między nimi
- metody optymalizacji - stronicowanie, te takie priorytety kolumn
- PLSQL - SQL na sterydach - porównaj, opisz możliwosci, krótkie zapytania na naszej bazie

Rozdział 4

Angular 2

-cechy charakterystyczne, dlaczego taki modny - standalone, własny serwer, SINGLE-PAGE APP
- Typescript vs Javascript - statyczne typowanie, kompatybilnosc z JS i bibliotekami - moja konfiguracja bibliotek i moja konfiguracja kompilowania i wrzucania do projektu Springa
- serwisy i modele po stronie angulara
-Zamiast tego co poniżej omów main.ts, app.module.ts, app.routing.ts i 1 moduł-scoreboard - moje kontrolery - klient - w tym opis podstawowych elementóW kontrolera, to że własny css
- moje kontrolery - admin

Rozdział 5

Bezpieczeństwo

- token do CSRF
- https
- token logowania wymieniany za hasło - policz też entropię, ile zajmie złamanie, wykaż że razem z punktem [BAN na IP] strona jest bezpieczna
- reset i zmiana hasła
- BAN na IP
- SQL Injection - nie działa bo Java

Rozdział 6

Wdrożenie systemu

-Tomcat i pliki jar - Jakie są mniej więcej chmury, konfiguracja i screeny jak wrzucam - TBA

Rozdział 7

Wnioski

- policz czasy ładowania strony i porównaj z pro stronami - wp, facebook. Dodaj dużo danych i sprawdź w stresie może.
-wady debugowania Angulara, czasy kompilacji Springa/Mavena/Tomcata - czy tyle warstw aplikacji na pewno jest nam potrzebne?
- Zmiany w technologii - im bliżej frontendu tym szybciej. Warto odpowiedzieć sobie na pytanie - w czym ta nowa technologia jest lepsza od mojej, ale też dobierać technologię do projektu?

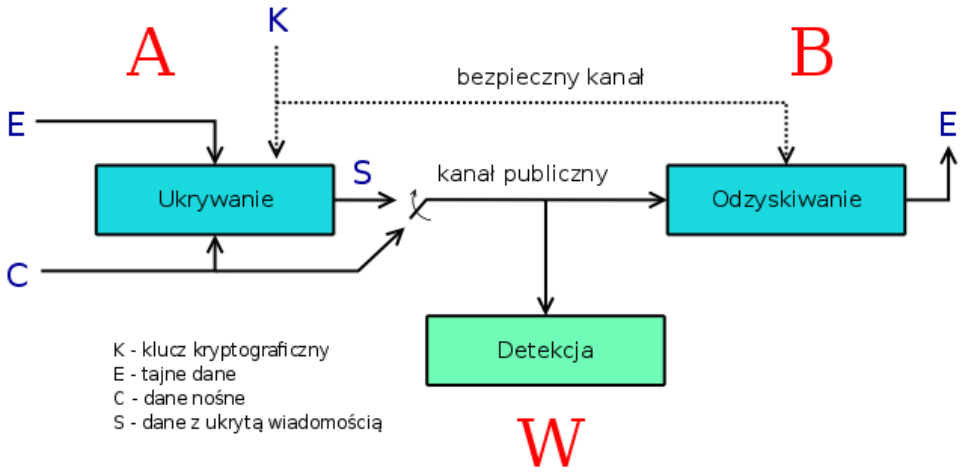
Rozdział 8

Schowek

8.1 Rzeczy

8 [7]

- Punkt 1
- Punkt 2



Rysunek 8.1: Schemat komunikacji steganograficznej

```
int main(String args){}
```

Bibliografia

[1] Dokumentacja Springa/Mądra książka o Springu

[2] Mądra książka o PLSQL - ta do wersji 10 co czytałes w PKO

[3] Dokumentacja Angulara

[4] Ta książka o gamifikacji albo wikipedia

[5] Cos o bezpieczenstwie - poszukaj w materialach do zajęć z bezpieczeństwa, oni ogarniali temat

[6] Thinking in JAVA?? Jak znajdziesz fragment który się przyda

[7] W. R. Stevens, G. R. Wright, „Biblia TCP/IP tom 1", RM, 1998.

Opinia

o pracy dyplomowej magisterskiej wykonanej przez dyplomanta
Zdolnego Studenta i Pracowitego Kolegę
Wydział Elektryczny, kierunek Informatyka, Politechnika Warszawska
Temat pracy
TYTUŁ PRACY DYPLOMOWEJ

Promotor: **dr inż. Miły Opiekun**
Ocena pracy dyplomowej: **bardzo dobry**

Treść opinii

Celem pracy dyplomowej panów dolnego Studenta i Pracowitego Kolegi było opracowanie systemu pozwalającego symulować i opartego o oprogramowanie o otwartych źródłach (ang. Open Source). Jak piszą Dyplomanci, starali się opracować system, który łatwo będzie dostosować do zmieniających się dynamicznie wymagań, będzie miał niewielkie wymagania sprzętowe i umożliwiał dalszą łatwą rozbudowę oraz dostosowanie go do potrzeb. Przedstawiona do recenzji praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy, trzech rozdziałów (2-4) zawierających opis istniejących podobnych rozwiązań, komponentów rozpatrywanychjako kandydaci do tworzonego systemu i wreszcie zagadnień wydajności wirtualnych rozwiązań. Piąty rozdział to opis przygotowanego przez Dyplomantów środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne. Ostatni rozdział pracy to opis możliwości dalszego rozwoju projektu. W ramach przygotowania pracy Dyplomanci zebrali i przedstawili w bardzo przejrzysty sposób duży zasób informacji, co świadczy o dobrej orientacji w nowoczesnej i ciągle intensywnie rozwijanej tematyce stanowiącej zakres pracy i o umiejętności przejrzystego przedstawienia tych wyników. Praca zawiera dwa dodatki, z których pierwszy obejmuje wyniki eksperymentów i badań nad wydajnością, a drugi to źródła skryptów budujących środowisko.

Dyplomanci dość dobrze zrealizowali postawione przed nimi zadanie, wykazali się więc umiejętnością zastosowania w praktyce wiedzy przedstawionej w rozdziałach 2-4. Uważam, że cele postawione w założeniach pracy zostały pomyślnie zrealizowane. Proponuję ocenę bardzo dobrą (5).

(data, podpis)

Recenzja

pracy dyplomowej magisterskiej wykonanej przez dyplomanta
Zdolnego Studenta i Pracowitego Kolegę
Wydział Elektryczny, kierunek Informatyka, Politechnika Warszawska
Temat pracy
TYTUŁ PRACY DYPLOMOWEJ

Recenzent: **prof. nzw. dr hab. inż. Jan Surowy**
Ocena pracy dyplomowej: **bardzo dobry**

Treść recenzji

Celem pracy dyplomowej panów dolnego Studenta i Pracowitego Kolegi było opracowanie systemu pozwalającego symulować i opartego o oprogramowanie o otwartych źródłach (ang. Open Source). Jak piszą Dyplomanci, starali się opracować system, który łatwo będzie dostosować do zmieniających się dynamicznie wymagań, będzie miał niewielkie wymagania sprzętowe i umożliwiał dalszą łatwą rozbudowę oraz dostosowanie go do potrzeb. Przedstawiona do recenzji praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy, trzech rozdziałów (2-4) zawierających bardzo solidny i przejrzysty opis: istniejących podobnych rozwiązań (rozdz. 2), komponentów rozpatrywanychjako kandydaci do tworzonego systemu (rozdz. 3) i wreszcie zagadnień wydajności wirtualnych rozwiązań, zwłaszcza w kontekście współpracy kilku elementów sieci (rozdział 4). Piąty rozdział to opis przygotowanego przez Dyplomantów środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne (5 ćwiczeń). Ostatni, szósty rozdział pracy to krótkie zakończenie, które wylicza także możliwości dalszego rozwoju projektu. W ramach przygotowania pracy Dyplomanci zebrali i przedstawili w bardzo przejrzysty sposób duży zasób informacji o narzędziach, Rozdziały 2, 3 i 4 świadczą o dobrej orientacji w nowoczesnej i ciągle intensywnie rozwijanej tematyce stanowiącej zakres pracy i o umiejętności syntetycznego, przejrzystego przedstawienia tych wyników. Drobne mankamenty tej części pracy to zbyt skrótowe omawianie niektórych zagadnień technicznych, zakładające dużą początkową wiedzę czytelnika i dość niestaranne podejście do powołań na źródła. Utrudnia to w pewnym stopniu czytanie pracy i zmniejsza jej wartość dydaktyczną (a ta zdaje się być jednym z celów Autorów), ale jest zrekompensowane zawartością merytoryczną. Praca zawiera dwa dodatki, z których pierwszy obejmuje wyniki eksperymentów i badań nad wydajnością, a drugi to źródła skryptów budujących środowisko. Praca zawiera niestety dość dużą liczbę drobnych błędów redakcyjnych, ale nie wpływają one w sposób istotny na na jej czytelność i wartość. W całej pracy przewijają się samodzielne, zdecydowane wnioski

Autorów, które są wynikiem własnych i oryginalnych badań. Rozdział 5 i dodatki pracy przekonują mnie, że Dyplomanci dość dobrze zrealizowali postawione przed nimi zadanie. Pozwala to stwierdzić, że wykazali się więc także umiejętnością zastosowania w praktyce wiedzy przedstawionej w rozdziałach 2-4. Kończący pracę rozdział szósty świadczy o dużym (ale moim zdaniem uzasadnionym) poczuciu własnej wartości i jest świadectwem własnego, oryginalnego spojrzenia na tematykę przedstawioną w pracy dyplomowej. Uważam, że cele postawione w założeniach pracy zostały pomyślnie zrealizowane. Proponuję ocenę bardzo dobrą (5).

(data, podpis)