POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (INF)

SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH (INS)

PRACA DYPLOMOWA

INŻYNIERSKA

Aplikacja webowa umożliwiająca zarządzanie domowym ogrodem

A web application that allows you to manage your home garden

AUTOR:

Łukasz Broll

PROWADZĄCY PRACĘ:

Dr inż, Marek Woda, Jednostka

OCENA PRACY:

WROCŁAW, 2018

**Spis treści**

[Spis rysunków 4](#_Toc530336034)

[Spis tabel 5](#_Toc530336035)

[Spis listingów 6](#_Toc530336036)

[Skróty 7](#_Toc530336037)

[1. Wstęp 8](#_Toc530336038)

[1.1. Wprowadzenie 8](#_Toc530336039)

[1.2. Cel pracy 8](#_Toc530336040)

[1.3. Zakres pracy 9](#_Toc530336041)

[1.4. Układ pracy 9](#_Toc530336042)

[2. Specyfikacja wymagań 10](#_Toc530336043)

[2.1. Wymagania funkcjonalne 10](#_Toc530336044)

[2.2. Wymagania niefunkcjonalne 11](#_Toc530336045)

[3. Projekt 12](#_Toc530336046)

[3.1. Architektura 12](#_Toc530336047)

[3.1.1. Wzorzec MVC 12](#_Toc530336048)

[3.1.2. Baza danych MySQL 13](#_Toc530336049)

[3.1.3. Aplikacja serwerowa Spring 13](#_Toc530336050)

[3.1.4. Aplikacja kliencka Angular 14](#_Toc530336051)

[3.2. Diagram encji 15](#_Toc530336052)

[3.2.1. User 15](#_Toc530336053)

[3.2.2. Garden 15](#_Toc530336054)

[3.2.3. Plant 16](#_Toc530336055)

[3.3. Diagram klas 16](#_Toc530336056)

[3.4. Narzędzia 17](#_Toc530336057)

[4. Implementacja 17](#_Toc530336058)

[5. Sekcja 17](#_Toc530336059)

[5.1. Sekcja poziomu 1 17](#_Toc530336060)

[5.1.1. Sekcja poziomu 2 18](#_Toc530336061)

[5.2. Kolejna sekcja poziomu 1 18](#_Toc530336062)

[5.2.1. Kolejna sekcja poziomu 2 18](#_Toc530336063)

[6. Podsumowanie i wnioski 20](#_Toc530336064)

[Literatura 21](#_Toc530336065)

[Dodatek A 22](#_Toc530336066)

# Spis rysunków

[Rys. 1. Przykład podpisu rysunku 18](#_Toc530334916)

# Spis tabel

[Tab. 1. Przykład podpisu tabeli 2](#_Toc465685652)

# Spis listingów

[Listing. 1. Początkowe żadanie HTTP 2](#_Toc465685644)

# Skróty

**OGC** (ang. *Open Geospatial Consortium*)

**XML** (ang. *eXtensible Markup Language*)

**SOAP** (ang. *Simple Object Access Protocol*)

**WSDL** (ang. *Web Services Description Language*)

**UDDI** (ang. *Universal Description Discovery and Integration*)

**GIS** (ang. *Geographical Information System*)

**SDI** (ang. *Spatial Data Infrastructure*)

**ISO** (ang. *International Standards Organization*)

**WMS** (ang. *Web Map Service*)

**WFS** (ang. *Web Feature Service*)

**WPS** (ang. *Web Processing Service*)

**GML** (ang. *Geography Markup Language*)

**SRG** (ang. *Seeded Region Growing*)

**SOA** (ang. *Service Oriented Architecture*)

**IT** (ang. *Information Technology*)

1. Wstęp
   1. Wprowadzenie

Roślinność to nieodzowny element życia ludzi. Bez tlenu, który wytwarzają rośliny, życie na Ziemi by nie istniało. W dobie bardzo dużego zużycia zasobów leśnych, m.in. wycinki lasów tropikalnych, ważne jest dbanie o możliwie jak najmniejszą degradację środowiska naturalnego. Każda, nawet najmniejsza roślina ma wpływ na natlenienie powietrza i redukcję dwutlenku węgla, dzięki czemu niweluje szkodliwy wpływ przemysłu na naturę. Czy to duży domowy ogród, czy kwiat na oknie, warto dbać o obecność roślin. Niestety każda roślinka wymaga mniejszej lub więcej uwagi do życia. Należy dbać o nawodnienie, nasłonecznienie i ochronę przed bakteriami i wirusami. Tylko regularna opieka gwarantuje profity z posiadania domowego ogrodu – wizualne oraz zdrowotne.

* 1. Cel pracy

Celem pracy jest utworzenie narzędzia wspomagającego zarządzanie domowym ogrodem, które wspomagać będzie regularne prace i obowiązki do utrzymania zdrowia roślin. Dzięki przechowywaniu w jednym miejscu informacji o wszystkich swoich roślinach i ogrodach, łatwiej będzie kontrolować bieżące obowiązki, a inteligentne listy roślin wymagających uwagi, ułatwią i przyspieszą obowiązki. Narzędzie zostanie zrealizowane w postaci aplikacji webowej, tj. programu działającego na serwerze, dostępnego tylko za pośrednictwem urządzenia z dostępem do sieci Internet, bez względu na używany system operacyjny, gdzie klientem aplikacji jest dostępna przeglądarka internetowa. Wszystkie zasoby i dane zapisywane przez użytkownika, w tego typu aplikacji, przechowywane są w zewnętrznej bazie danych. Zastosowane technologie mają gwarantować że program będzie responsywny, tj. będzie poprawnie skalować się na każdym typie urządzenia i rozmiarze ekranu, obsługa będzie intuicyjna, a interfejs nowoczesny.

* 1. Zakres pracy

Początkowym etapem pracy będzie specyfikacja wymagań funkcjonalnych oraz niefunkcjonalnych aplikacji. Na tym etapie, konieczne jest dokładne zdefiniowanie co ma oferować aplikacja oraz czego nie będzie oferować. W kolejnym etapie zostaną wybrane technologie, w których projekt zostanie zrealizowany, a następnie zostanie utworzony projekt aplikacji serwerowej i klienckiej oraz model bazy danych. Następnym krokiem będzie implementacja rozwiązania oraz jego testy. Na końcu wykonana zostanie dokumentacja aplikacji.

* 1. Układ pracy

W kolejnych rozdziałach umieszczone zostaną konkretne treści. W Rozdziale 5 będzie ...

1. Specyfikacja wymagań

W rozdziale opisane są wymagania, jakie musi spełniać aplikacja do zarządzani domowym ogrodem. Podstawowym wymaganiem jest udostępnienie użytkownikami aplikacji do zarządzania domowym ogrodem, która będzie dostępna w języku polskim. Poniżej wymienione zostały wymagania funkcjonalne oraz niefunkcjonalne.

* 1. Wymagania funkcjonalne

1. Logowanie do systemu opiera się o nazwę użytkownika oraz hasło. W przypadku rejestracji należy podać również imię, nazwisko oraz adres e-mail.
2. Hasło jest przechowywane w bazie danych w postaci zaszyfrowanej za pomocą funkcji bcrypt.
3. Użytkownik może tworzyć ogrody, które będą przypisane tylko do jego konta. Każdy ogród identyfikowany jest poprzez nazwę oraz można go edytować lub usunąć. Przechowywane są również informacje o utworzeniu i ostatniej edycji ogrodu.
4. W ogrodzie przechowywana jest lista roślin, każda roślina składa się na:
   1. nazwę,
   2. opis,
   3. notatki,
   4. prace związane z rośliną, tj. podlewanie, nawożenie, przesadzanie, dawkowanie środków ochrony roślin – w skład każdej pracy wchodzi:
      1. cykl wykonywania czynności,
      2. ostatni termin jej wykonania,
      3. aktualny status czynności.
   5. data utworzenia oraz ostatniej modyfikacji rośliny,
5. Wszystkie dane rośliny, oprócz ostatniego wykonywania każdej z czynności, można edytować lub usuwać roślinę oraz każde pole w formularzu jest walidowane przed zapisaniem.
6. Cykl wykonywania każdej pracy związanej z rośliną, jest wybierany z zdefiniowanej listy terminów.
7. Aktualny status pracy związanej z rośliną jest automatycznie aktualizowany, jeśli minął termin jej ostatniego wykonania.
8. Informacje o roślinie, na podstawie jej nazwy, można wyszukać bezpośrednio z poziomu aplikacji w zewnętrznych serwisach.
9. Dla każdej rośliny zdefiniowany jest terminarz prac oraz przypomnień w postaci listy zadań, w której znajdują się aktualne stany prac.
   1. Wymagania niefunkcjonalne
10. Aplikacja powinna działać poprawnie na poniższych przeglądarkach:

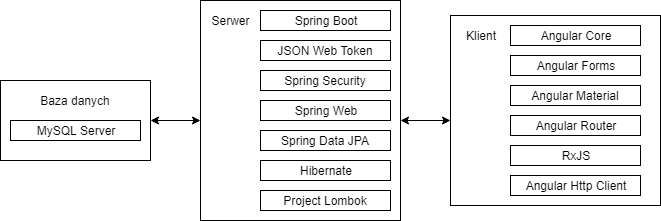
* Google Chrome w wersji 60 lub wyższej,
* Mozilla Firefox w wersji 54 lub wyższej,
* Microsoft Edge w wersji 14 lub wyższej,

1. Technologie oraz rozwiązania systemów informatycznych, wykorzystane do utworzenia aplikacji, powinny być darmowe.
2. Aplikacja powinna działać, niezależnie od sprzętu i systemu operacyjnego użytkownika.
3. Obsługa aplikacji powinna być intuicyjna i wygodna dla każdego użytkownika.
4. Projekt

Projekt został oparty o technologie związane z językiem Java. Taka decyzja został podjęta, ponieważ tok studiów jest ściśle związany z tym językiem oraz autor ma w nim największe doświadczenie. Czynnikami decydującymi o skorzystaniu z wybranych framework’ów oraz rozwiązań były ich popularność oraz nowoczesność, co wiąże się z dużą społecznością wsparcia i dokumentacji.

* 1. Architektura

Aplikacja została podzielona na trzy niezależne od siebie moduły: bazę danych, serwer oraz aplikację kliencką. Dzięki takiemu rozwiązaniu łatwiej jest kontrolować działanie całego systemu oraz modyfikacje nad poszczególnymi komponentami. W związku z tym, schemat systemu razem z wykorzystanymi technologiami, wygląda następująco:



Rysunek 1 Architektura systemu

* + 1. Wzorzec MVC

Rozwinięciem skrótu MVC to Model-View-Controler, czyli Model-Widok-Kontroler. Jest to wzorzec projektowy pozwalający na podział struktury aplikacji na moduły. W przypadku aplikacji webowej, wzorzec MVC wykorzystywany jest w następujący sposób:

* + - 1. Kontroler odbiera żądanie http, na którego podstawie odpowiednio przetwarza i interpretuje dane wejściowe.
      2. Model odbiera przetworzone dane wejściowe z kontrolera i przetwarza je, np. aktualizując stan obiektu.
      3. Widok odbiera z powrotem dane z kontrolera, w przypadku opisywanej aplikacji, w formacie JSON i aktualizuje wyświetlane komponenty.
    1. Baza danych MySQL

MySQL to jeden z najpopularniejszych relacyjnych systemów zarządzania bazą danych. Został wykorzystany w projekcie, ponieważ bardzo dobrze współpracuje z wykorzystanymi technologiami w aplikacji serwerowej, jest łatwy do skonfigurowania oraz umożliwia bezproblemowe przeniesienie całej bazy na np. zewnętrzny serwer.

* + 1. Aplikacja serwerowa Spring

Spring to szkielet, służący do tworzenia aplikacji głównie w języku Java. Oferuje rozwiązania usprawniające prace nad rozwijaniem oprogramowania, których wykorzystanie znacznie redukuje ilość kodu konieczną do napisania przez programisty. M.in. usprawnia obsługiwanie wyjątków, komunikację z bazą danych oraz umożliwia wstrzykiwanie zależności. W projekcie, w ramach Springa, wykorzystane zostały następujące moduły:

1. Spring Boot – narzędzie umożliwiające automatyzację konfiguracji oraz instalacji potrzebnych modułów.
2. Spring Security – biblioteka do zarządzania bezpieczeństwem aplikacji, chroni zasoby przed nieautoryzowanym dostępem.
3. Spring Web – biblioteka dostarczająca zbiór funkcjonalności usprawniających tworzenie aplikacji webowych.
4. Spring Data JPA – biblioteka usprawniająca dostęp do źródła danych, która na podstawie oficjalnego standardu JPA (Java Persistence API), umożliwia uproszczone mapowanie na obiekty zawartości bazy danych.
   * + 1. Project Lombok

Biblioteka ograniczająca generowanie wielu linijek powtarzającego się kodu podczas definiowania klas, szczególnie klas modelu. Za pomocą adnotacji możliwe jest automatycznie tworzenie np. konstruktora bezargumentowego (@NoArgsConstructor), metod zwracających pola klasy (@Getter), metod ustawiających pola klasy (@Setter) czy zabezpieczenie przed brakiem wartości pola (@NonNull).

* + - 1. JSON Web Token

Standard generowania tokenów uwierzytelniających sesje użytkownika. Jest przechowywany po stronie klienta i przesłany w zapytaniach do serwera, który na podstawie klucza weryfikuje czy zezwolić na wykonanie zapytania i dostęp do zasobów. Jak nazwa wskazuje, token jest oparty na formacie JSON i składa się z trzech części – nagłówka, zawartości i sygnatury. Wszystkie dane wewnątrz tokena są zakodowane algorytmem Base64. W nagłówku znajdują się informacje o wykorzystanym algorytmie szyfrującym. W drugiej części przechowywane są informacje o użytkowniku, m.in. nazwa użytkownika oraz czas sesji. W sygnaturze przechowywany jest cyfrowy podpis serwera.

* + 1. Aplikacja kliencka Angular

Angular to framework autorstwa Google, który ma spełniać podobne zadanie co Spring, jednak przeznaczony dla języka TypeScript. We wzorcu MVC odpowiada za Widok, czyli to co widzi użytkownik. Głównymi cechami charakteryzującymi Angulara są renderowanie po stronie serwera, zorientowania na komponenty oraz modularność. Wykorzystując dyrektywy, które są zagnieżdżone w kodzie HTML, możliwe jest łatwe kontrolowanie interfejsu użytkownika na podstawie logi napisanej w języku TypeScript. Wiele metod i funkcji jest już zdefiniowanych w szkielecie Angulara, przez co programista nie musi ręcznie tworzyć każdej dyrektywy. W projekcie, w ramach Angulara, wykorzystane zostały następujące moduły:

* + - 1. Angular Core – główny rdzeń i silnik Angulara, który odpowiada za działanie frameworka.
      2. Angular Forms – biblioteka dostarczająca zestaw metod i funkcjonalności usprawniających tworzenie formularzy, ich walidację oraz wysyłanie.
      3. Angular Material – biblioteka zawierająca komponenty wizualne stylu Material Design, który polega na dostarczeniu prostego, czytelnego interfejsu dostosowanego do różnych urządzeń.
      4. Angular Router – biblioteka usprawniająca przekierowania wewnątrz aplikacji, wspierająca m.in. parametry w adresie.
      5. RxJS – zestaw narzędzi który rozszerza możliwości Angulara o programowanie reaktywne, m.in. operacje na kolekcjach.
      6. Angular Http Client – biblioteka wspierająca komunikację z aplikacją serwerową poprzez protokół http.
  1. Diagram encji

Diagram przestawia encje bazy danych dla modelu klas w aplikacji.



Rysunek Diagram encji

* + 1. User

User, czyli użytkownik jest identyfikowany po polu username (nazwa użytkownika). Każdy użytkownik ma unikalny ‘username’ oraz ‘eamil’. Hasło użytkownika tj. pole ‘password’ jest zakodowane za pomocą funkcji bcrypt. Domyślną rolą każdego użytkownika jest „USER”. Użytkownik może posiadać nieograniczoną liczbę ogrodów.

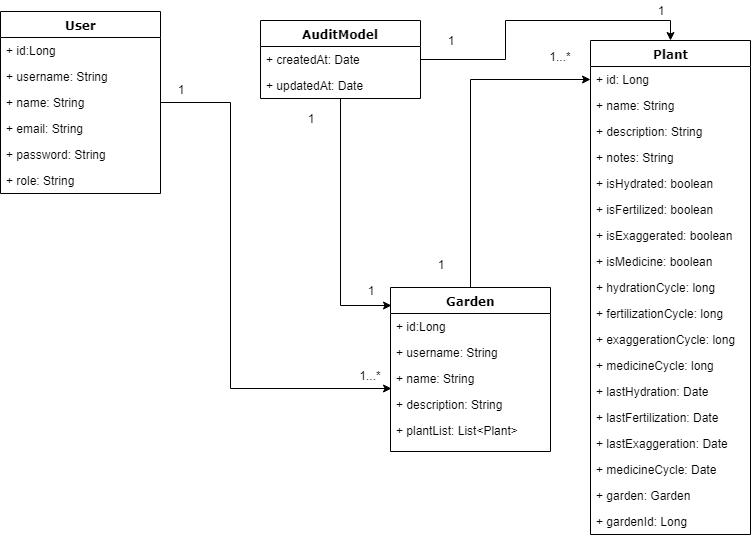
* + 1. Garden

Garden, czyli ogród jest identyfikowany po ID. Może istnieć tylko w relacji z użytkownikiem. Zawiera informacje o nazwie, opisie, dacie powstania i ostatniej modyfikacji. Każdy ogród może posiadać nieograniczoną liczbę roślin.

* + 1. Plant

Plant, czyli roślina jest identyfikowana po ID. Jest powiązana z konkretnym ogrodem, czyli również użytkownikiem. Każda roślina musi posiadać informacje o pracy związanej z podlewaniem, nawożeniem, przesadzaniem i podawaniem środków ochrony roślin.

* 1. Diagram klas

Na diagramie nie uwzględniono konstruktorów klas, getterów oraz seterów pól, które są generowane automatycznie zza pomocą biblioteki Lombok oraz klas, które przechowują modele odpowiedzialne za autoryzację i zabezpieczenia aplikacji tj. AuthToken, Constants, LoginUser oraz RegisterUser,

Rysunek Diagram klas

* 1. Narzędzia

Wybór narzędzi potrzebnych do utworzenia aplikacji, opierał się na zagwarantowaniu wsparcia dla wykorzystywanych technologii oraz języków programowania. Jednym z warunków była również darmowa licencja na oprogramowanie. Zdecydowano się wykorzystać następujące aplikacje:

1. JetBrains IntelliJ IDEA – rozbudowane zintegrowane środowisko programistyczne oferujące wsparcie podczas tworzenia aplikacji serwerowej oraz klienckiej. Dla studentów wersja aplikacji Ultimate jest darmowa.
2. MySQL Workbench – graficzny interfejs bazy danych MySQL, umożliwiający projektowanie, administrację oraz kontrolę bazy danych.
3. Postman – aplikacja służąca do testowania oprogramowania, symulująca zapytania do kontrolerów aplikacji serwerowej.
4. Maven – narzędzie służące do automatyzacji procesu budowy oprogramowania oraz ułatwiające korzystanie z niezależnych bibliotek w aplikacji.
5. Node.js – środowisko i silnik działania aplikacji opartych o JavaScript ( w tym TypeScript), w skład którego wchodzi również npm, czyli menadżer pakietów za pomocą którego zarządza się wartswą front-endu
6. Git – system kontroli wersji zapewniający możliwość kontrolowania procesu wytwarzania oprogramowania.
7. Google Chrome – przeglądarka internetowa na której testowana była budowa warstwy klienckiej aplikacji.

1. Implementacja

Opis implementacji.

* 1. Połączenia
  2. Kontrolery
  3. Zabezpieczenia

1. Testy
2. Aplikacja
3. Podsumowanie

1. Sekcja
   1. Sekcja poziomu 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

* + 1. Sekcja poziomu 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

* + - 1. Sekcja poziomu 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Tab. 1. Przykład podpisu tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

* 1. Kolejna sekcja poziomu 1
     1. Kolejna sekcja poziomu 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Rys. 1. Przykład podpisu rysunku

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Listing. 1. Początkowe żądanie HTTP

GET /script/Articles/Latest.aspx HTTP/1.1  
Host: www.codeproject.com  
Connection: keep –alive  
Cache -Control: max-age=0  
Accept: text/html ,application/xhtml+xml,application/xml|  
User -Agent: Mozilla/5.0 ...  
Accept -Encoding: gzip ,deflate ,sdch  
Accept -Language: en-US...  
Accept -Charset: windows -1251,utf -8...

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Przykład listy numerowanej:

1. Podsumowanie i wnioski

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus. Aliquam erat volutpat. Integer maximus est turpis, ut bibendum ligula accumsan et. Ut eget vestibulum libero. Aliquam erat volutpat. Nullam placerat mauris a lectus tincidunt, et aliquet turpis aliquam. Etiam in malesuada lacus. Proin dignissim augue sit amet auctor elementum. Suspendisse potenti. Vivamus suscipit vulputate massa ac molestie. Suspendisse a justo porttitor, commodo mi at, placerat risus. Integer lobortis augue ac neque suscipit, vel sodales lacus fringilla.

# Literatura

[1] M. Bickley, C. Slominski. A MySQL-based data archiver: preliminary results. Proceedings of ICALEPCS07, Paz. 2007. http://www.osti.gov/scitech/servlets/purl/922267 [dostęp dnia 20 czerwca 2015].

[2] J. Jędrzejczyk, B. Sródka. Segmentacja obrazów metodą drzew decyzyjnych. Raport instytutowy, Politechnika Wrocławska, Wydział Elektroniki, 2007.

# Dodatek A