

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka (INF)
SPECJALNOŚĆ: Inżynieria Internetowa (INT)

**PROJEKT NA ZAJĘCIA SYSTEMÓW
WBUDOWANYCH**

System do zarządzania zadaniami.

AUTOR:

Sebastian Wilgosz (nr indeksu: 195963), Rafał
Sztandera (nr indeksu: XXXXXX)

PROWADZĄCY:

dr. mgr inż. Marek Woda

OCENA PRACY:

Spis treści

1	Wstęp	3
1.1	Cel projektu	3
1.2	Opis problemu	3
1.3	Opis rozwiązania i wykorzystanych technologii	3
1.3.1	Aplikacja serwerowa	3
1.3.2	Instalacja i uruchomienie aplikacji serwerowej	4
2	Implementacja	5
2.1	Implementacja aplikacji serwerowej	5
2.2	Implementacja aplikacji rpi	5
3	Dostarczone API	7
3.1	Obsługa błędów	7
3.1.1	Forbidden	7
3.1.2	Not found	8
3.1.3	Unprocessable entity (422)	8
3.1.4	Invalid (unauthorized) (401)	8
3.1.5	Invalid (406)	9
3.2	Sesje logowania	9
3.2.1	Logowanie	9
3.3	Zarządzanie listami	10
3.3.1	Pobranie list zadań:	10
3.3.2	Pobranie pojedynczej listy zadań:	11
3.3.3	Tworzenie listy	12
3.4	Zarządzanie zadaniami	12
3.4.1	Pobranie zadań danej listy:	12
3.4.2	Pobranie pojedynczego zadania:	13
3.4.3	Tworzenie zadania	14

Rozdział 1

Wstęp

1.1 Cel projektu

1.2 Opis problemu

1.3 Opis rozwiązania i wykorzystanych technologii

1.3.1 Aplikacja serwerowa

Do zaimplementowania aplikacji serwerowej wykorzystaliśmy:

- język programowania Ruby, wersja 2.2.2 [1]
- Framework Ruby on Rails, wersja 4.2.3 [2]
- CoffeScript [3]
- Sass [4]
- Bibliotekę jQuery do języka JavaScript [5]
- Bazę danych SQLite

Wybór padł na technologię Ruby on Rails głównie ze względu na łatwość integracji z REST-owym API, ale ważnymi czynnikami były także: szybkość powstawania aplikacji internetowych w tym frameworku, oraz na obecną popularność na rynku.

1.3.2 Instalacja i uruchomienie aplikacji serwerowej

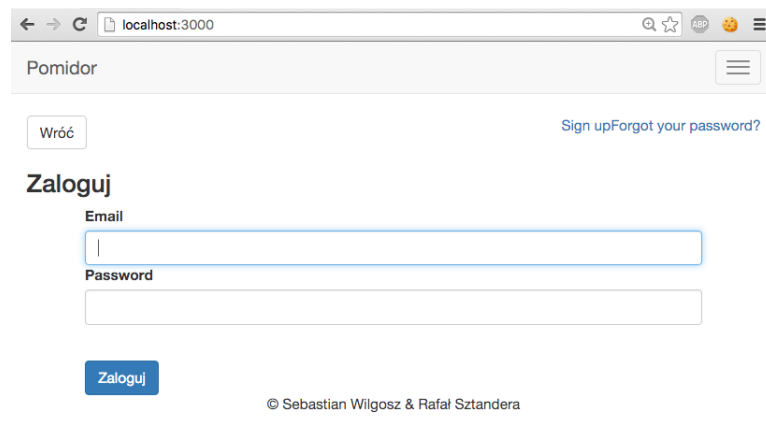
Aby uruchomić środowisko aplikacji, należy mieć zainstalowany język ruby oraz framework Ruby on Rails, wraz ze wszystkimi potrzebnymi wtyczkami. W systemie Ubuntu, aby to osiągnąć, należy najpierw zainstalować manager wersji Ruby (RVM [6]). Kiedy mamy to zrobione, należy uruchomić terminal, przejść do folderu aplikacji, a następnie zainstalować wersję 2.2.2 języka Ruby oraz odpowiedni komplet gemów.

```
rvm install 2.2.2
rvm use 2.2.2@test --create
gem install bundler
bundle install
```

Po zakończeniu wykonywania powyższych komend, środowisko aplikacji powinno być zainstalowane. Kolejnym krokiem jest uruchomienie lokalnego serwera aplikacji.

```
rails s
```

Od tej pory aplikacja powinna być dostępna pod adresem `http://localhost:3000`, co widać na rysunku 1.1,



Rysunek 1.1 Widok strony głównej aplikacji

Rozdział 2

Implementacja

Kompletne repozytorium aplikacji jest dostępne pod adresem: <https://github.com/pwr-wilgosz/pomidor>, poniżej przedstawiliśmy najważniejsze założenia i sposoby implementacji konkretnych rozwiązań.

2.1 Implementacja aplikacji serwerowej

2.2 Implementacja aplikacji rpi

Rozdział 3

Dostarczone API

Poniżej zamieszczona została szczegółowa dokumentacja zaimplementowanego API. Każda sekcja składa się z przykładowego zapytania do serwera, listy wymaganych argumentów, które należy przekazać przez URL, możliwe statusy odpowiedzi, które może wysłać serwer, oraz szczegółową odpowiedź przykładową serwera.

Każdy obiekt JSON zawiera klucz główny, definiujący typ zwracanego modelu, wewnątrz którego zdefiniowane są jego atrybuty.

3.1 Obsługa błędów

Poniżej znajduje się lista akceptowanych przez serwer zapytań od aplikacji, oraz lista zwracanych obiektów w formacie JSON.

Serwer nie powinien zwracać nieprzechwytywanych wyjątków, lecz obiekt JSON z odpowiednim polem statusu oraz odpowiednim opisem błędu:

3.1.1 Forbidden

Ten błąd zwracany jest w przypadku próby wykonania nieautoryzowanej akcji, np. próby utworzenia listy/zadania bez wcześniejszego zalogowania.

Format:

```
{
  "message": "You are not authorized to access this resource.",
  "status": 403
}
```

3.1.2 Not found

Próba odwołania do zasobu nieistniejącego na serwerze.

Format:

```
{
  "message": "Resource is not found.",
  "status": 404
}
```

3.1.3 Unprocessable entity (422)

Nieprawidłowy format zapytania, na przykład gdy w parametrach do zadania podamy argumenty bez nazwy obiektu:

```
{
  "name": "Test 1"
}
```

zamiast:

```
{
  "task": {
    "name": "Test 1"
  }
}
```

Format:

```
{
  "message": "Unprocessable entity.",
  "status": 422
}
```

3.1.4 Invalid (unauthorized) (401)

Autoryzacja jest możliwa, ale się nie udała (przy logowaniu, jeśli podamy nieprawidłowe dane)

Format:

```
{
  "message": "Incorrect email or password",
  "access_token" : null
  "status": 401
}
```

3.1.5 Invalid (406)

Występuje, gdy nie są spełnione wymogi walidacji, np

```
{
  "message": "List cannot be saved – please, review the errors below.",
  "list": {
    "errors": {
      "title": [
        "can't be blank",
        "has already been taken, but 'List 2 is available'"
      ]
    }
  }
}
```

3.2 Sesje logowania

3.2.1 Logowanie

Aby zalogować użytkownika i otrzymać odpowiadający mu klucz dostępu, należy wysłać zapytanie do serwera, przekazując w parametrach email oraz hasło. W przypadku pomyślnego logowania, serwer zwróci kod dostępu, *access_token*, który od tej pory będzie autoryzował użytkownika.

Każde zapytanie do serwera bez podanego w parametrze klucza dostępu zwróci obiekt JSON odpowiadający błędowi (403) 3.1

```
POST http://tomato-cal.herokuapp.com/login.json
```

Wymagane parametry:

- email
- password

Statusy odpowiedzi:

- 200 - ok
- 403 - unauthorised

Odpowiedzi:

Poprawna (200)

```
{
  "status": 200,
  "access_token" : "abcdefgh"
}
```

Nieprawidłowe dane logowania (403)

```
{
  "message" : "Invalid email or password",
  "status": 403
}
```

3.3 Zarządzanie listami

3.3.1 Pobranie list zadań:

Założenie jest takie, że każdy użytkownik ma dostęp wyłącznie do swoich list.

```
GET http://tomato-cal.herokuapp.com/lists?access_token="abc23@klj1309"
```

Wymagane argumenty:

- access_token - token identyfikujący użytkownika

Statusy:

- 200 - ok
- 403 - not authorized

Odpowiedzi:

Poprawna (200) - zwracana jest tablica list wraz z tablicami zadań danej listy

```
{
  "lists_count": 9,
  "lists": [
    {
      "id": 1,
      "user_id": 1,
      "name": "This is sample list",
      "identifier": "serv_cklsjef323sd3",
      "created_at": "2013-05-30T13:47:41Z",
      "updated_at": "2013-05-30T13:47:41Z"
    }
  ]
}
```

3.3.2 Pobranie pojedynczej listy zadań:

```
GET http://tomato-cal.herokuapp.com/lists/1?access_token="abc23@klj1309"
```

Wymagane argumenty:

- access_token - token identyfikujący użytkownika
- id - id identyfikujące daną listę

Statusy:

- itemize 200 - ok
- itemize 403 - not authorized

Odpowiedzi:

Poprawna (200) - zwracana jest pojedyncza lista wraz z tablicą zadań.

```
{
  "list":
  {
    "id": 1,
    "user_id": 1,
    "user": {
      "id": 1,
      "created_at": "2015-11-22T15:47:22.701Z",
      "updated_at": "2015-11-22T15:47:22.768Z",
      "email": "email1@example.com"
    }
    "name": "This is sample list",
    "identifier": "serv_cklsjef323sd3",
    "created_at": "2013-05-30T13:47:41Z",
    "updated_at": "2013-05-30T13:47:41Z"
    "tasks": [
      {
        "id": 1,
        "name": "Task 1",
        "identifier": "serv_cklsjef323sd3",
        "description": "Sample Text",
        "priority": 3,
        "duration": 2,
        "position_x": 12,
        "position_y": 84
      }
    ]
  }
}
```

3.3.3 Tworzenie listy

```
POST http://tomato-cal.herokuapp.com/lists.json
```

Wymagane argumenty

- access_token - token identyfikujący użytkownika§
- name - unikalny tytuł

Statusy:

- 200 - ok
- 403 - forbidden
- 406 - not acceptable - validation error

Odpowiedzi:

Correct (201)

```
{
  "list": {
    "id": 1,
    "user_id": 1,
    "user": {
      "id": >1,
      "created_at": >"2015-11-22T15:47:22.701Z",
      "updated_at": >"2015-11-22T15:47:22.768Z",
      "email": >"email1@example.com"
    },
    "name": "This is sample list",
    "identifier": "serv_cklsjef323sd3",
    "created_at": "2013-05-30T13:47:41Z",
    "updated_at": "2013-05-30T13:47:41Z",
    "tasks": []
  }
}
```

3.4 Zarządzanie zadaniami

3.4.1 Pobranie zadań danej listy:

Założenie jest takie, że każdy użytkownik ma dostęp wyłącznie do swoich zadań.

```
GET http://tomato-cal.herokuapp.com/lists/1/tasks?access_token="abc23@klj1309"
```

Wymagane argumenty

- access_token - token identyfikujący użytkownika
- list_id - identyfikacja listy

Statusy:

- 200 - ok
- 403 - not authorized

Odpowiedzi:

Poprawna (200) - zwracana jest tablica zadań

```
{
  "tasks_count": 9,
  "tasks": [
    {
      "id" => 1,
      "name" => "Test task",
      "priority" => 1,
      "created_at" => "2015-11-22T15:47:22.701Z",
      "updated_at" => "2015-11-22T15:47:22.768Z",
      "identifier" => "serv_llopY8Kz",
      "x" => 104,
      "y" => 45,
      "duration" => 3,
      "list_id" => 1
    }
  ]
}
```

3.4.2 Pobranie pojedynczego zadania:

```
GET http://tomato-cal.herokuapp.com/lists/1/tasks/1?access_token="abc23@klj1309"
```

Wymagane argumenty

- access_token - token identyfikujący użytkownika
- list_id - id identyfikujące daną listę
- id - id identyfikujące dane zadanie

Statusy:

- 200 - ok
- 403 - not authorized

Odpowiedzi:

Poprawna (200) - zwracane jest pojedyncze zadanie.

```
{
  "task" => {
    "id" => 1,
    "name" => "Test task",
    "priority" => 1,
    "created_at" => "2015-11-22T15:47:22.701Z",
    "updated_at" => "2015-11-22T15:47:22.768Z",
    "identifier" => "serv_13llzyI2k",
    "x" => 104,
    "y" => 45,
    "duration" => 3,
    "list_id" => 1,
    "list" => {
      "id" => 1,
      "name" => "Test list",
      "identifier" => "serv_13llzyI2k",
      "created_at" => "2015-11-22T15:47:22.701Z",
      "updated_at" => "2015-11-22T15:47:22.768Z",
      "user_id" => 1
    }
  }
}
```

3.4.3 Tworzenie zadania

```
POST http://tomato-cal.herokuapp.com/lists/1/tasks.json
```

Wymagane argumenty

- access_token - token identyfikujący użytkownika
- title - unikalny tytuł
- list_id - identyfikator listy

Statusy:

- 201 - created
- 403 - forbidden
- 406 - not acceptable - validation error

Odpowiedzi:

Poprawna (201)

```
{
  "task" => {
    "id" => 1,
    "name" => "Test task",
    "priority" => 1,
    "created_at" => "2015-11-22T15:47:22.701Z",
    "updated_at" => "2015-11-22T15:47:22.768Z",
    "identifier" => "serv_13llzyI2k",
    "x" => 104,
    "y" => 45,
    "duration" => 3,
    "list_id" => 1,
    "list" => {
      "id" => 1,
      "name" => "Test list",
      "identifier" => "serv_13llzyI2k",
      "created_at" => "2015-11-22T15:47:22.701Z",
      "updated_at" => "2015-11-22T15:47:22.768Z",
      "user_id" => 1
    }
  }
}
```


Rozdział 4

Wnioski

Bibliografia

- [1] Dokumentacja języka Ruby, <https://www.ruby-lang.org/en/>
- [2] Dokumentacja Ruby on rails, <http://guides.rubyonrails.org/>
- [3] Dokumentacja CoffeeScript, <http://coffeescript.org/>
- [4] Dokumentacja Sass, <http://sass-lang.com/>
- [5] Dokumentacja biblioteki jQuery, <https://jquery.com/>
- [6] Opis instalacji RVM (Ruby Version Manager), [https://rvm.io/rvm/
install](https://rvm.io/rvm/install)