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), Rafal Sztandera (nr indeksu: 200774)

PROWADZĄCY:
dr. mgr inż. Marek Woda

OCENA PRACY:

Spis treści

Rozdział 1

Wstęp

1.1 Cel projektu

Celem projektu jest stworzenie systemu mającego rozwiązać problem niskiej efektywności pracy osoby zaangażowanej w wiele różnych projektów, przy użyciu platformy mikroprocesorowej, łacza sieciowego oraz aplikacji serwerowej.

Projekt obejmuje zaprojektowanie i stworzenie autonomicznego urządzenia wspomagającego zarządzenie czasem, serwisu internetowego udostępniającymi te same funkcjonalności w tym tworzenie i prezentację zadań oraz stworzenie i wdrożenie metody synchronizacji danych pomiędzy urządzeniem, a serwisem webowym.

Platforma mikroprocesorowa będzie docelowo wykorzystywana w środowisku domowym jako element wyposażenia. Powinna być łatwo dostępna, dawać możliwość kożystania z niej bez użycia klawiatury, myszy i osobnego ekranu.

1.2 Opis problemu

Istnieje wiele systemów, które wspomagają prowadzenie i organizację projektów. Są one tworzone głównie z myślą o wykonawcy zleceń, jak i o organizatorach. Zasadą ich działania jest zbieranie informacji od użytkownika dotyczące niezbędnych do wykonania zadań oraz układanie ich w formie planu - terminarza. W użyciu przypominają rozbudowany kalendarz umożliwiający łatwe dodawanie nowych zadań oraz podpowiadającego efektywne zagospodarowanie czasu.

1.2.1 Wymagania

System przeznaczony jest dla pojedynczych użytkowników, do użytku własnego. Z tej perspektywy ważnymi, ale nie kluczowymi wyznacznikami urządzenia są:

1. zapewnienie maksymalnej niezawodności
2. wytrzymałość na zniszczenia i warunki ekstremalne
3. możliwość wykonania kopii zapasowej

Kluczowymi cechami platformy, ze względu na swoje przeznaczenie i środowisko pracy są:

1. łatwe zarządzanie danymi
2. trwałość i pewność zapisu
3. wysoka dostępność do danych
4. intuicyjność interfejsu aplikacji
5. łatwa konfiguracja
6. płynność działania aplikacji
7. szybka synchronizacja

Wymagania podstawowe stawiane aplikacji:

1. możliwość anulowania operacji
2. możliwość manipulacji danymi (dodawanie/edycja/usuwanie)
3. interfejs w języku polskim

Dostępność tłumaczy się jako możliwość podglądu zadań z poziomu aplikacji internetowej jak i aplikacji na platformach mikroprocesorowych zsynchronizowanych z aplikacją internetową.

1.2.2 Ograniczenia

Ograniczeniem dla projektu są:

1. wymiary

   platforma mikroprocesorowa powinna być możliwie płaska, oraz posiadać duży ekran dotykowy

2. zasilanie

   systemy wbudowane wymagają specjalnego zasilania

3. dostęp do internetu

   internet jest niezbędny do dwukierunkowej synchronizacji danych z serwerem

Ograniczeniem dla projektu w przyszłości mogą być:

1. przepisy regulujące kwestie związane z używaniem urządzeń elektrycznych w określonym środowisku (np. łazienka)

1.3 Opis rozwiązania i wykorzystanych technologii

1.3.1 Aplikacja serwerowa

Do zaimplementowania aplikacji serwerowej wykorzystaliśmy:

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

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

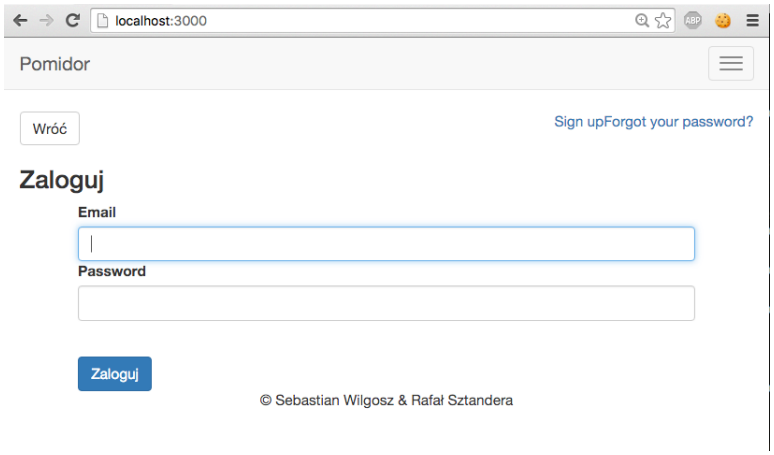
Aby uruchomić środowisko aplikacji, należy mieć zainstalowany język ruby oraz framework Ruby on Rails, wraz ze wszystkimi potrzebnymi wtyczkami. W systemie Ubutntu, aby to osiągnąć, należy najpierw zainstalować manager wersji Ruby (RVM [6]). Kiedy mamy to zrobione, należy uruchomić terminal, jprzejść 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 odpowiedzią 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 uzytkownika 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 - identifikator 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
    }
  }
}
```

3.5 Synchronizacja

Aby zsynchronizować bazy, należy wykonać zapytanie:

```
GET http://tomato-cal.herokuapp.com/sync
```

Wymagane argumenty:

- access_token - token identyfikujący użytkownika
- snapshot - obraz bazy danych na urządzeniu: kolekcja wszystkich list i zadań.

Przykład:

```
curl -H 'Content-Type: application/json' -H 'Accept: application/json' -X POST http
    ://tomato-cal.herokuapp.com/sync?access_token=42fF2zhLai3 -d '
{
  lists: [
    {
      identifier: "rpi_4LK3kll2nza",
      name: "Test list",
      updated_at: "2013-05-30T13:47:41Z",
      tasks: [
        {
```

```
            "name" => "Test task",
            "priority"=> 1,
            "updated_at"=>"2015-11-22T15:47:22.768Z",
            "identifier" => 'rpi_0v9Kds2ngi',
            "x"=> 104,
            "y"=> 45,
            "duration" => 3
          }
        ]
      }
    ]
}'
```

Statusy:

- 200 - ok
- 403 - not authorized

Odpowiedzi:

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

```
{
  lists: [
    {
      id: 1,
      identifier: "rpi_4LK3kll2nza",
      name: "Test list",
      created_at: "2013-05-30T13:47:41Z",
      updated_at: "2013-05-30T13:47:41Z",
      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" => task.identifier,
          "x"=> 104,
          "y"=> 45,
          "duration" => 3,
          "list_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