Szkolenie Rails

Skrypt: Aplikacja do zarządzania inteligentnym domem.

1.
- ❏ Utwórz nową aplikację Ruby on Rails.
- ❏ Inicjujemy w niej nowe repozytorium git.
- ❏ Dodaj wszystkie pliki do repozytorium
- ❏ git commit “initial commit”

2. Utwórz nowe repozytorium na bitbuckecie i ustaw je jako zdalne repozytorium ‘origin’.
3. Do Gemfile dodaj następujące gemy: “twitter-bootstrap-rails”, "therubyracer", “less-rails” a następnie uruchom polecenie bundle install.
4. Zainstaluj gem twitter-bootsrap w aplikacji uruchamiając polecenie:

```
rails generate bootstrap:install less
```

5. Wygeneruj responsywny layout, korzystając z generatora gemu twitter-bootstrap-rails:

```
rails g bootstrap:layout application fluid
```

6. Wygeneruj kontroler main z widokiem index, oraz ustaw go jako landing page.
7. Sprawdź czy działa, git commit.
8. Zapoznaj się ze schematem bazy danych dla projektu.
9. Korzystając ze scaffolda wygeneruj kontrolery, widoki oraz modele wraz z migracjami opierając się na schemacie bazy danych (oprócz tabeli Users)
- ❏ **rails g scaffold room name ….**
- ❏ **rails g scaffold category name ….**
- ❏ **rails g scaffold room_cateogory name ….**
- ❏ **rails g scaffold device name type on:boolean room_category:references**

a następnie:
- ❏ **rails g bootstrap:themed Rooms**
- ❏ **rails g bootstrap:themed Categories**
- ❏ **rails g bootstrap:themed RoomCategories**
- ❏ **rails g bootstrap:themed Devices**

10. Uruchom migracje bazy danych
11. Commit i push

12. Dodaj gem devise do Gemfile. Zainstaluj gem poleceniem - bundle i zainstaluj devise w projekcie następującym poleceniem:

```
rails g devise:install
```

Wygeneruj model User poleceniem`: rails g devise User.`

*13. (zadanie dodatkowe) Dodaj klasę css “navbar-inverse” do diva otwierającego navbar, aby zmienić kolor nawigacji.
14. W navbarze dodaj warunkowe linki do obsługi logowania. Zamieść je po prawej stronie, zagnieżdżone w istniejącej klasie

```
<div class="pull-right login-links">
  <% if user_signed_in? %>
 <%= link_to current_user.email, edit_user_registration_path, :title => "Edit Profile"%>
          <%= link_to "Log off", destroy_user_session_path, :method => :delete, :class
 => "label label-important" %>
  <% else %>
     <%= link_to "Log in", new_user_session_path, :method => :get %> |
     <%= link_to "Register", new_user_registration_path, :method => :get %>
  <% end %>
</div>
```

Jeżeli linki są słabo widoczne to uzupełnij odpowiednio plik css main.css.scss dla klasy login-links:
http://goo.gl/NQQoHh

15. Zamień link nazwy aplikacji z html template:
HomeAutomation
na railsowy link_to, wskazujący na root_path.
16. Commit.
17. Stwórz nowy kontroler o nazwie admin z jednym widokiem “index”. Utwórz link do nowego widoku w navbarze (zamiast “Link1”) i nazwij go AdminPanel. Usuń z navbaru “link2” i “link3”

W głównym layoucie wytnij i przenieś wszystkie elementy <li> z sidebara do nowego partiala views/admin/_sidebar.html.erb a w ich miejsce wstaw
<%= render ‘admin/sidebar’ %>
W powyższym partialu _sidebar podepnij link1 i link2 na rooms_path i categories_path i nazwij je odpowiednio.
18. Utwórz kolejny partial _sidebar.html.erb w katalogu views/main:

Zmień logikę wyświetlania sidebaru w głównym layoucie:

```
<% if params[:controller] == 'main' %>
        <%= render 'main/sidebar'%>
<% else %>
        <%= render 'admin/sidebar'%>
 <% end %>
```

http://goo.gl/WPZT9n

19. Zdefiniuj asocjacje w modelach wg schematu bazy danych.

20. W pliku seeds.rb utwórz defaultowe pokoje: Living Room, Bedroom, Kitchen, Hall, Office oraz kategorie: Alarm, Audio, Lighting, Heating
wygeneruj kilka powiązanych urządzeń (devices) z rekordami RoomCategory.

21. Zaimplementuj możliwość przypisywania Kategorii do poszczególnych pokoi w panelu administracyjnym, ponieważ mamy niewiele rekordów do wyboru.
Można to zrobić przy pomocy checkboxów:

views/rooms/_form.html.erb:

```
<%= f.collection_check_boxes :category_ids, Category.all, :id, :name %>
```

(Dla chętnych w domu proszę spróbować zaimplementować javascriptowe autouzupełniane token fields wg railscasta 258 - Token Fields (revisited)

22. Na widoku show dla modelu room dodaj wyświetlanie przypisanych kategorii.

*23. (Zad Dodatkowe przydatne do autoryzacji w przyszłości) Stwórz migrację dodającą pole boolean is_admin z defaultową wartością false:

```
        add_column :users, :is_admin, :boolean, default: false
```

24
Dodaj do seedów swojego użytkownika:

```
User.create!(email: 'twójmail@gmail.com', password: '123', password_confirmation: '123')
```

zmień zakres długości hasła na:

```
 config.password_length = 3..128
```

w pliku config/initializers/devise.rb

25. Wygeneruj kontroler users:

```
        rails g controller users index
```

Jako kolejne linki w sidebarze panelu administracyjnego dodaj:

<%= link_to "Room categories", room_categories_path %>
<%= link_to "Devices", devices_path %>
<%= link_to "Users", users_path %>

Zamień wygenerowane routesy:
routes.rb
na:

```
  scope "/admin" do
    resources :users
    resources :devices
    resources :categories
    resources :room_categories
    resources :rooms
  end
```

users_controller.rb:
Uzupełnij akcje index o zmienną globalną @users
Wypełnij widok/users index wzorując sie na scaffoldzie np: rooms/index
http://goo.gl/Y01N9u

26. Kontroler “main”: utwórz dwie nowe akcje: “room” i “category”, stwórz dla nich widoki w folderze views/main. Przygotuj zmienne dla nawigacji w sidebarze dodając metodę before_action :prepare_sidebar:

```
 private

  def prepare_sidebar
        @rooms = Room.all
        @categories = Category.all
  end
```

Ustaw odpowiednio routesy dla użytkowników aplikacji:
```
  get '/room/:id', to: 'main#room', as: 'main_room'
  get '/category/:id', to: 'main#category', as: 'main_category'
```

27. Main _sidebar:
Do nawigacji użyjemy Toggable tabs z biblioteki bootstrapa:

najpierw przygotujmy helperowe metody, aby przenieść logikę z widoku do heplers/main_helper.rb:
http://goo.gl/H0Ewlx

```
module MainHelper
	def category_class
		action_name == 'category' ? 'active' : ''
	end

	def room_class
		action_name != 'category' ? 'active' : ''
	end

end
```

Zawartość sidebara:
http://goo.gl/6p67hI

```
<ul class="nav nav-tabs" role="tablist">
	<%= content_tag(:li, link_to('Rooms', '#rooms', data: {toggle: 'tab'}), class: room_class)
%>
	<%= content_tag(:li, link_to('Categories', '#categories', data: {toggle: 'tab'}), class:
category_class ) %>
</ul>

<div class="tab-content">

	<%= content_tag(:div, class: "tab-pane " + room_class, id: 'rooms') do -%>
		<% @rooms.each do |room| %>
		<p>	<%= link_to room.name, main_room_path(room)  %> </p>
		<% end %>
  <% end -%>

	<%= content_tag(:div, class: "tab-pane " + category_class, id: 'categories' ) do -%>
		<% @categories.each do |cat| %>
		<p>	<%= link_to cat.name , main_category_path(cat) %> </p>
		<% end %>
	<% end -%>
</div>
```

28. Edytuj scaffoldowe widoki “Room Cateogries” w panelu administracyjnym:
w widoku index:

- ❏ zamiast (id, bądź obiektu) wyświetlaj nazwę pokoju oraz kategorii.
- ❏ usuń możliwość dodawania nowych rekordów “Room Categories”.
- ❏ usuń link destroy, aby nie było możliwe usuwanie tej asocjacji w tym miejscu.

w partialu _form.html.erb:
Wyświelaj nazwę pokoju i kategorii zamiast ich id. Zablokuj możliwość ich edycji.

Zadanie domowe: (nested attributes) W formularzu dodaj pola dla devices należących do room_category wg przykładu nested attributes z poprzednich zajęć.

29. Devices:
w modelu device.rb ustaw stałą types oraz zmień domyśne ustawienie STI z kolumny type na dev_type.

```
TYPES = ['switch']
self.inheritance_column = :dev_type
```

30. Włączanie urządzeń:
Instalujemy gem wg instrukcji:

```
gem "bootstrap-switch-rails"
```

https://github.com/manuelvanrijn/bootstrap-switch-rails

Uzupełniamy widoki main/room.html.erb oraz main/category.html.erb:
http://goo.gl/jzssFc
http://goo.gl/jzssFc

Wyłącz turbolinki
usuwając linię:
//= require turbolinks
z pliku app/assets/javascripts/application.js

W pliku main.js.coffee przy pomocy jQuery zamień checkboxy na bootstrapowe switche:

```
jQuery ->
  $("[name^='switch_']").bootstrapSwitch();
  $("[name^='switch_']").on "switchChange.bootstrapSwitch", (event, state) ->
    console.log this # DOM element
    console.log event # jQuery event
    console.log state # true | false
    console.log "TODO AJAX call to external app and to save state to db."
```

Kolejne kroki dla chętnych:

- ❏ Devise autoryzacja użytkowników:
  - ❏ dostęp do panelu administracyjnego tylko dla adminów.
  - ❏ ukrycie niedostępnych linków.
- ❏ Walidacje modeli, dopracowanie panelu administracyjnego.

- ❏ Zapisywanie w bazie danych zmian włączników utworzonych w poprzednim kroku.
- ❏ Deployment na heroku.
- ❏ Nested attributes.
- ❏ Stylizacja widoków devise bootstrapem.
- ❏ Zabezpieczenie niedozwolonych akcji w routesach.
- ❏ Redesign?
- ❏ Cokolwiek wam przyjdzie do głowy.