

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 9

з дисципліни “Основи програмування ”

тема “Завершення розробки і розгортання веб-сервісу.”

|  |  |  |
| --- | --- | --- |
| Виконав  студент II курсу  групи КП-52  Саприкін Артем Олексійович  (*прізвище, ім’я, по батькові*)  варіант №4­­­­­ |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2016 р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2016

**Мета роботи**

Навчитись готувати веб-проект до розгортання на віддаленому серверному хості та відкриття доступу до мережі Інтернет.

**Постановка завдання**

1. Опублікувати розроблений сайт на одній із платформ (наприклад, [Heroku](http://prog-base.appspot.com/www.heroku.com)) так, щоби сайт був доступний у мережі Інтернет.
2. Створити віддалену копію бази даних на одній із платформ у мережі Інтернет (наприклад, [mLab](http://prog-base.appspot.com/mlab.com) для MongoDB).
3. Підготувати скрипти для автоматичного встановлення і початкового налаштування веб-сервера.

**База даних**

|  |
| --- |
| Lab5 |
| CREATE TABLE public.auth\_user  (  id integer NOT NULL DEFAULT nextval('auth\_user\_id\_seq'::regclass),  password character varying(128) COLLATE pg\_catalog."default" NOT NULL,  last\_login timestamp with time zone,  is\_superuser boolean NOT NULL,  username character varying(150) COLLATE pg\_catalog."default" NOT NULL,  first\_name character varying(30) COLLATE pg\_catalog."default" NOT NULL,  last\_name character varying(30) COLLATE pg\_catalog."default" NOT NULL,  email character varying(254) COLLATE pg\_catalog."default" NOT NULL,  is\_staff boolean NOT NULL,  is\_active boolean NOT NULL,  date\_joined timestamp with time zone NOT NULL,  CONSTRAINT auth\_user\_pkey PRIMARY KEY (id),  CONSTRAINT auth\_user\_username\_key UNIQUE (username)  ) |

**Тексти коду**

|  |
| --- |
| directors.html |
| {% **extends 'base\_nav.html'** %} {% **load staticfiles** %} {% **load bootstrap3** %}  {% **block head** %}  <**script src="**{% **static 'js/mustache.min.js'** %}**"**></**script**>  <**script src="**{% **static 'js/pagination.min.js'** %}**"**></**script**>  <**script id="table\_template" type="text/mustache"** >  {% verbatim %}  <table class="table-hover table">  <tr>  <th>Firsname</th>  <th>Lastname</th>  <th>Birthdate</th>  <th>Year</th>  <th>Score</th>  <th>Avatar</th>  {{#can\_delete}}  <th>Options</th>  {{/can\_delete}}  </tr>  {{#dirs}}  <tr class="clickable-row" data-href="/directors/{{ id }}/">  <td>{{ name }}</td>  <td>{{ surname }}</td>  <td>{{ birthdate }}</td>  <td>{{ year }}</td>  <td>{{ score }}</td>  <td><img src="{{ avatar }}" height="64"></td>  {{#can\_delete}}  <td>  <form method="post" action="{{ id }}/" name="delete\_form">  <input type="submit" value="DELETE" class="btn-danger" id="id\_delete\_btn">  </form>  </td>  {{/can\_delete}}  </tr>  {{/dirs}}  </table>  {% endverbatim %}  </**script**>  <**script src="**{% **static 'js/search.js'** %}**"**></**script**> {% **endblock** %}   {% **block content** %}  <**div class="row"**>  <**div class="col-sm-3"**>  <**form role="form" method="get"**>  {% **bootstrap\_form form** %}  {% **buttons** %}  <**button type="submit" class="btn btn-primary" id="id\_search\_btn"**>  {% **bootstrap\_icon "star"** %} Search  </**button**>  {% **endbuttons** %}  </**form**>  </**div**>  </**div**>  <**div id="template\_div"**>   </**div**>   <**div class="paginationjs" id="pagination"**></**div**>  <**script src="**{% **static 'js/delete\_confirm.js'** %}**"**></**script**>  <**script src="**{% **static 'js/click\_row.js'** %}**"**></**script**> {% **endblock** %} |
| Search.js |
| */\*\*  \* Created by Artem on 12/3/2016.  \*/* **function** *addScripts*() {  **$**(**".clickable-row"**).click(**function**(e) {  **if**(e.target.**id** != **"id\_delete\_btn"**)  **window**.**document**.**location** = **$**(**this**).data(**"href"**);  **return true**;  });   **$**(**"[name='delete\_form']"**).submit(**function**(e) {  **c** = confirm(**'Are you sure?'**);  **if**(!**c**) {  e.preventDefault();  **return false**;  }  **else** {  **return true**;  }  });  }  **function** *instantSearch*() {  **var** ajaxRequest;  **console**.log(**'instant search'**);  ajaxRequest = **new** XMLHttpRequest();  ajaxRequest.**onreadystatechange** = **function**() {  **if**(ajaxRequest.**readyState** == 4 && ajaxRequest.**status** == 200) {  **var** template = **$**(**"#table\_template"**).html();  **var** all = **JSON**.parse(ajaxRequest.**responseText**);  **can\_delete** = all[**'can\_delete'**];  **if**(!all[**'dirs'**].**length**) {  **if**(!**$**(**'#id\_search\_error'**).**length**)  **$**(**'#id\_name'**).after(**'<span id="id\_search\_error" class="error text-warning">Nothing found</span>'**)  **$**(**'#template\_div'**).html(**''**);  **$**(**'#pagination'**).hide()  }  **else** {  **$**(**'#id\_search\_error'**).remove();    **var** pagination\_selector = **$**(**'#pagination'**);  pagination\_selector.show();  pagination\_selector.pagination({  **dataSource**: all[**'dirs'**],  **pageSize**: 5,  **autoHidePrevious**: **true**,  **autoHideNext**: **true**,  **ulClassName**: **'pagination'**,  callback: **function**(data, pagination) {  **var** dict = {  **'dirs'**: data,  **'can\_delete'**: **can\_delete** };  **html** = ***Mustache***.render(template, dict);  **$**(**'#template\_div'**).html(**html**);  *addScripts*();  }  })  }  }  };  ajaxRequest.open(**'GET'**, **'/get-directors/?name='**+**$**(**'#id\_name'**).val());  ajaxRequest.send() }  **$**(**document**).ready(**function**() {  *instantSearch*();  **$**(**'#id\_name'**).on(**'input'**, *instantSearch*);  **$**(**'#id\_settings\_btn'**).on(**'input'**, settings); }); |

|  |
| --- |
| Click\_row.js |
| */\*\*  \* Created by Artem on 11/26/2016.  \*/ /\*jQuery(document).ready(function($) {  $(".clickable-row").click(function(e) {  if(e.target.id != "id\_delete\_btn")  window.document.location = $(this).data("href");  return true;  }); });\*/* **function** *addEvent*(node, type, callback) {  **if**(node.addEventListener) {  node.addEventListener(type, **function**(e) {  callback(e, e.target);  }, **false**);  }  **else if** (node.attachEvent) {  node.attachEvent(**'on'** + type, **function**(e) {  callback(e, e.**srcElement**);  }, **false**);  } }  **clickable\_elements** = **document**.getElementsByClassName(**'clickable-row'**);  **for** (**var *i*** = 0; ***i*** < **clickable\_elements**.**length**; ***i***++) {  *addEvent*(**clickable\_elements**[***i***], **'click'**, **function** (e) {  **if** (e.target.**id** != **"id\_delete\_btn"**) {  **window**.**document**.**location** = **"/"** } }); } |

|  |
| --- |
| Delete\_confirm.js |
| */\*\*  \* Created by Artem on 11/30/2016.  \*/* **function** *addEvent*(node, type, callback) {  **if**(node.addEventListener) {  node.addEventListener(type, **function**(e) {  **return** callback(e, e.target)  }, **false**)  }  **else if**(node.attachEvent) {  node.attachEvent(**'on'** + type, **function**(e) {  **return** callback(e, e.**srcElement**)  }, **false**)  } }  **delete\_forms** = **document**.getElementsByName(**'delete\_form'**); **for** (**var *i*** = 0; ***i*** < **delete\_forms**.**length**; ***i***++) {  *addEvent*(**delete\_forms**[***i***], **'submit'**, **function**(e, target) {  **c** = confirm(**'Are you sure?'**);  **if**(!**c**) {  e.preventDefault();  **return false**;  }  **else** {  **return true**;  }  }); } |

|  |
| --- |
| Form\_validation.js |
| */\*\*  \* Created by Artem on 11/29/2016.  \*/* **var *elements*** = [  **document**.getElementsByTagName(**"input"**),  **document**.getElementsByTagName(**"textarea"**) ];  **function** *addEvent*(node, type, callback) {  **if**(node.addEventListener) {  node.addEventListener(type, **function**(e) {  callback(e, e.target);  }, **false**);  }  **else if** (node.attachEvent) {  node.attachEvent(**'on'** + type, **function**(e) {  callback(e, e.**srcElement**);  }, **false**);  } }   **function** *getErrorMessage*(field) {  **pattern** = field.getAttribute(**"pattern"**);  **value** = field.**value**;   **if** (field.**required** && !**value**)  **return** field.getAttribute(**"name"**)+**" is required"**;  **if** (!field.**required** && !**value**)  **return ""**;  **if** (**pattern** && !(**new** RegExp(**pattern**).test(**value**)))  **return "Invalid format"**;   **return ""**; }  **var *has\_invalid*** = **false**; **for** (**var *i*** = 0; ***i*** < ***elements***.**length**; ***i***++) {  **for** (**var *j*** = 0; ***j*** < ***elements***[***i***].**length**; ***j***++)  **if**(*getErrorMessage*(***elements***[***i***][***j***])) {  ***has\_invalid*** = **true**;  }  } **if**(!***has\_invalid***)  **document**.getElementById(**"id\_create\_btn"**).removeAttribute(**"disabled"**)   **function** *instantValidation*(field) {  **error\_msg** = *getErrorMessage*(field);  **if**(!**error\_msg**) {  field.parentElement.**classList**.remove(**"has-error"**);  **error** = **document**.getElementById(**"id\_"**+field.getAttribute(**"name"**)+**"\_error"**);  **if**(**error**) {  **error**.**parentNode**.removeChild(**error**);  }  }  **else** {  **error\_element** = **document**.getElementById(**"id\_"**+field.getAttribute(**"name"**)+**"\_error"**);  **if** (!**error\_element**) {  **var** divBlock = **document**.createElement(**'span'**);  divBlock.**innerHTML** = **"<span>"**+**error\_msg**+**"</span>"**;  divBlock.**className** = **"error text-danger"**;  divBlock.**id** = **"id\_"**+field.getAttribute(**"name"**)+**"\_error"**;  }  **else** {  **error\_element**.**innerHTML** = **"<span>"**+**error\_msg**+**"</span>"**;  }   field.parentElement.appendChild(divBlock);  field.parentElement.**classList**.add(**"has-error"**);  **document**.getElementById(**"id\_create\_btn"**).setAttribute(**"disabled"**, **"disabled"**);  }  **var** has\_invalid = **false**;  **for** (**var** i = 0; i < ***elements***.**length**; i++) {  **for** (**var** j = 0; j < ***elements***[i].**length**; j++)  **if**(*getErrorMessage*(***elements***[i][j])) {  has\_invalid = **true**;  }  }  **if**(!has\_invalid)  **document**.getElementById(**"id\_create\_btn"**).removeAttribute(**"disabled"**) }   **for** (**var *i*** = 0; ***i*** < ***elements***.**length**; ***i***++) {  **for** (**var *j*** = 0; ***j*** < ***elements***[***i***].**length**; ***j***++)  *addEvent*(***elements***[***i***][***j***], **'input'**, **function**(e, target) {  *instantValidation*(target);  }) } |

|  |
| --- |
| Views.py |
| **from** django.http **import** HttpResponse, HttpResponseRedirect **from** django.shortcuts **import** render **from** .forms **import** DirectorSearchForm, DirectorCreationForm **from** .models **import** Director **from** django.core.paginator **import** Paginator, EmptyPage, PageNotAnInteger **import** math **from** django.views **import** View **from** django.core.exceptions **import** PermissionDenied **from** django.contrib.auth.decorators **import** login\_required, user\_passes\_test **from** django.utils.decorators **import** method\_decorator **from** django.views.decorators.csrf **import** csrf\_exempt **from** django.contrib.auth.models **import** User   @method\_decorator(login\_required(login\_url=**'/login/'**), name=**'dispatch'**) **class** CreateView(View):  **def** get(self, request):  dir\_form = DirectorCreationForm()  args = dict()  args[**'form'**] = dir\_form  **return** render(request, **'create\_director.html'**, args)    **def** post(self, request):  dir\_form = DirectorCreationForm(request.POST, request.FILES)  **if** dir\_form.is\_valid():  dir\_form.save()  **return** HttpResponse(**'Director added successfully'**)  **else**:  **return** render(request, **'create\_director.html'**, {**'form'**: dir\_form})   @method\_decorator(login\_required(login\_url=**'/login/'**), name=**'dispatch'**) **class** DirectorsView(View):  **def** get(self, request):  args = dict()  **if 'name' in** request.GET:  directors\_list = Director.objects.filter(name=request.GET[**'name'**])  args[**'name'**] = request.GET[**'name'**]  **else**:  directors\_list = Director.objects.all()   settings = int(request.session.get(**'settings'**, **False**))  **if not** settings:  settings = math.sqrt(len(directors\_list))   paginator = Paginator(directors\_list, max(1, settings))   **try**:  directors = paginator.page(request.GET.get(**'page'**))  **except** PageNotAnInteger:  directors = paginator.page(1)  **except** EmptyPage:  directors = paginator.page(paginator.num\_pages)  args[**'dirs'**] = directors  args[**'form'**] = DirectorSearchForm()  **return** render(request, **'directors.html'**, args)    **def** post(self, request):  **if 'settings' in** request.POST:  settings = request.POST[**'settings'**]  request.session[**'settings'**] = settings  **return** HttpResponseRedirect(**'/directors/'**)   @method\_decorator(csrf\_exempt, name=**'dispatch'**) @method\_decorator(login\_required(login\_url=**'/login/'**), name=**'dispatch'**) **class** DirectorView(View):  **def** get(self, request, id):  args = dict()  **try**:  dir = Director.objects.get(id=id)  args[**'dir'**] = dir  **return** render(request, **'director.html'**, args)  **except**:  **return** HttpResponse(**'director doesnt exist'**)    **def** post(self, request, id):  **if** request.user.has\_perm(**'directors.can\_delete'**):  **try**:  dir = Director.objects.get(id=id)  name = dir.name  dir.delete()  **return** HttpResponse(**'Director %s has been deleted'** % name)  **except**:  **return** HttpResponse(**'director doesnt exist'**)  **else**:  **raise** PermissionDenied   **def** user\_check(user):  **return** user.is\_superuser   @user\_passes\_test(user\_check) **def** my\_admin(request):  args = dict()  args[**'users'**] = User.objects.all()  **return** render(request, **'my\_admin.html'**, args) |

REST API

GET /api/directors – повертає список усіх об’єктів

POST /api/directors – створює нову сутність

OPTIONS /api/directors – повертає опції

GET /api/directors/id – повертає інформацію про заданого директора

Параметри

Name=name

Surname=surname

Year=year

Фільтрує результати

PUT /api/directors/id - модифікує заданого директора

DELETE /api/directors/id – видаляє задану сутність

OPTIONS /api/directors/id – повертає опції

Налаштування для Heroku

Requirements.txt

dj-database-url==0.4.1

Django==1.10.4

django-bootstrap3==7.1.0

djangorestframework==3.5.3

gunicorn==19.6.0

Pillow==3.4.2

psycopg2==2.6.2

whitenoise==3.2.2

Procfile

web: gunicorn lab9.wsgi --log-file -

**Висновки**

Під час виконання даної лабораторної роботи я навчився розгортанню веб сервісів.