

Отчет по лабораторной работе № 5 по курсу
"Разработка Интернет-Приложений"

Выполнил:
Студент группы
ИУ5-55Б
Хижняков Вадим Максимович

Москва, МГТУ – 2021

Задание:

В этой лабораторной работе Вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также Вам нужно будет дополнить свои классы предметной области, связав их с созданной БД. После этого Вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей.

1. Создайте сценарий с подключением к БД и несколькими запросами, примеры рассмотрены в методических указаниях.
2. Реализуйте модели Вашей предметной области из предыдущей ЛР (минимум две модели, т.е. две таблицы).
3. Создайте представления и шаблоны Django для отображения списка данных по каждой из сущностей

Для выполнения лабораторной работы была выбрана база данных Postgres. Чтобы избежать проблем с локальной установкой и настройкой базы данных приложение запускается в Docker контейнере.

Поскольку в качестве клиента использовалось React приложение, написанное в рамках ЛР4, оно также было докеризировано.

Текст программы:
docker-compose.yml

```
version: "3.9"

services:
  ripdb:
    image: postgres
    container_name: ripdb
    volumes:
      - ./data/ripdb:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=root
      - POSTGRES_DB=root
  api:
    build:
      context: ./server
      dockerfile: Dockerfile
    command: "sh entripoint.sh"
    volumes:
      - "./server:/code"
    ports:
      - "8000:8000"
    environment:
      - POSTGRES_NAME=root
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=root
    depends_on:
      - ripdb
  client:
    container_name: client
    build:
      context: ./client
      dockerfile: Dockerfile
    ports:
      - "5000:80"
    environment:
      - NODE_ENV=production
    depends_on:
      - api
```

docker-compose.dev.yml

```
version: "3.9"

services:
  client:
    build:
      dockerfile: Dockerfile.dev
    volumes:
      - "./client:/usr/src/app"
    ports:
      - "3000:3000"
    environment:
```

```
- NODE_ENV=development
```

server/Dockerfile

```
# syntax=docker/dockerfile:1
FROM python:3 as build-deps
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
WORKDIR /code
COPY requirements.txt .
RUN pip install -r requirements.txt
```

settings.py

```
DATABASES = {
    'default': {
        'ENGINE': env('POSTGRES_ENGINE', default='django.db.backends.sqlite3'),
        'NAME': env('POSTGRES_NAME', default=(os.path.join(BASE_DIR, 'db.sqlite3'))),
        'USER': env('POSTGRES_USER'),
        'PASSWORD': env('POSTGRES_PASSWORD'),
        'HOST': env('POSTGRES_HOST', default='localhost'),
        'PORT': env('POSTGRES_PORT', default='5432'),
    }
}
```

scenario.py

```
import psycopg2

db = psycopg2.connect(
    host=env('POSTGRES_HOST', default='localhost'),
    user=env('POSTGRES_USER'),
    dbname=env('POSTGRES_PASSWORD'),
)

c = db.cursor()
c.execute('INSERT INTO "Lab4_Manufacturer" (name) VALUES (%s);', ('UAZ'))
db.commit()
c.close()
db.close()
```

models.py

```
from django.db import models

class Manufacturer(models.Model):
    name = models.CharField(max_length=200)
```

```

def __str__(self):
    return self.name

class Car(models.Model):
    name = models.CharField(max_length=200, verbose_name="Name")
    price = models.PositiveIntegerField(verbose_name="Price")
    description = models.TextField(verbose_name="Description")
    main_image = models.ImageField(
        upload_to='car_images/',
        blank=True
    )
    manufacturer = models.ForeignKey(
        'Manufacturer',
        on_delete=models.CASCADE,
    )
    score = models.PositiveSmallIntegerField(verbose_name="Score")

def __str__(self):
    return self.name

```

Код клиента для лабораторной работы №5 был написан с помощью библиотеки React. В качестве аналогичной работы будет изложен код РК2, написанный с использованием шаблонов django с применением css фреймворка bootstrap5

form.html

```

<div class="mb-3">
    <label for="bookName" class="form-label">Book Name</label>
    <input type="text" name="name" value="{{ book.name }}" required="required" class="form-control" id="bookName" />
</div>

<div class="mb-3">
    <label for="bookCost" class="form-label">Book Cost</label>
    <input
        type="number"
        min="0"
        name="cost"
        value="{{ book.cost }}"
        required="required"
        class="form-control"
        id="bookCost"
    />
</div>

<div class="mb-3">

```

```

<label for="bookStore" class="form-label">Select Store</label>
<select name="store" class="form-select" aria-label="Default select example" id="bookStore">
  {% for store in stores %}
    <option value="{{ store.pk }}">{{ store.name }}</option>
  {% endfor %}
</select>
</div>

```

card.html

```

{% load static %}

<div class="card mb-3">
  <div class="row g-0">
    <div class="col-md-4">
      
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">{{ book.name }}</h5>
        <p class="card-text">{{ book.store.name }}</p>
        <p class="card-text">
          This is a wider card with supporting text below as a natural lead-in to additional content. This
          content is a little bit longer.
        </p>
        <p class="card-text"><small class="text-muted">{{ book.cost }}</small></p>
        <a href="{% url 'delete_book' book.pk %}" class="btn btn-danger"> Delete </a>
        <a href="{% url 'update_book' book.pk %}" class="btn btn-primary"> Update </a>
      </div>
    </div>
  </div>
</div>

```

list.html

```

{% extends 'inc/base.html' %} {% load static %} {% block content %}
<div class="container">
  <h1 class="text-center mt-4">All Books</h1>
</div>
<!-- Page Content -->
<div class="d-flex flex-column">
  {% for book in books %} {% include 'book/book_card.html' with book=book %} {% endfor %}
</div>
{% endblock content %}

```

create.html

```

{% extends 'inc/base.html' %} {% load static %} {% block content %}
<div class="container">
  <h1 class="text-center mt-4">New Book</h1>

```

```

</div>
<!-- Page Content -->
<div>
  <form action="{% url 'create_book' %}" method="POST" class="form">
    {% csrf_token %} {% include 'book/book_form.html' with book=book %}
    <button type="submit" class="btn btn-primary">Create</button>
  </form>
</div>
{% endblock content %}

```

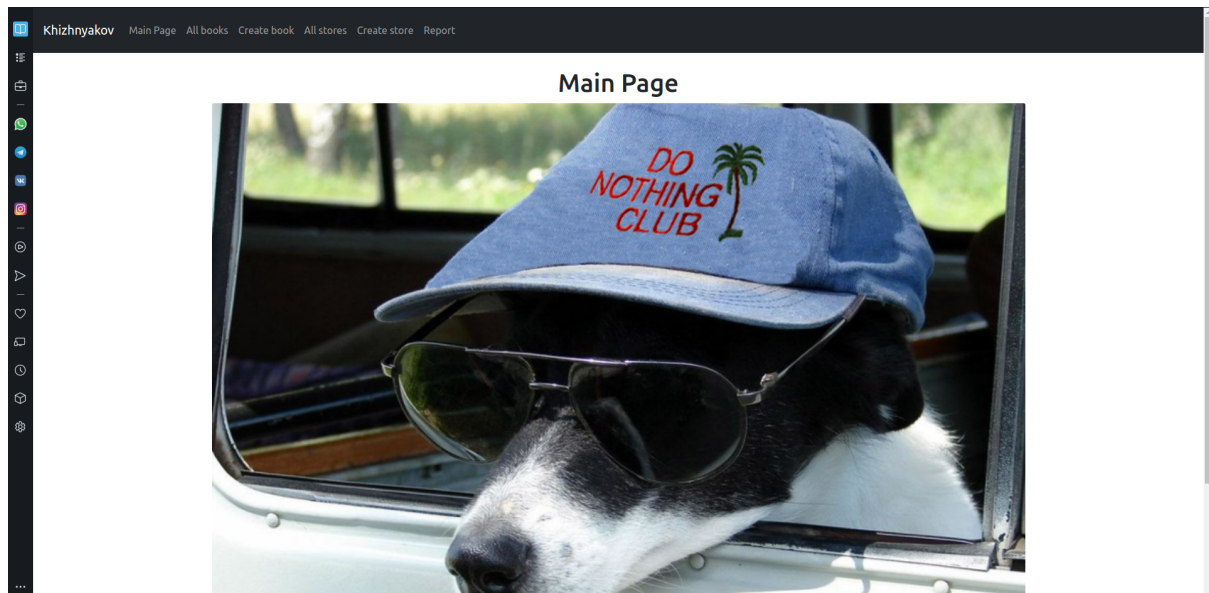
update.html

```




{% extends 'inc/base.html' %} {% load static %} {% block content %}
<div class="container">
  <h1 class="text-center mt-4">Update Book</h1>
</div>
<!-- Page Content -->
<div>
  <form action="{% url 'update_book' book.pk %}" method="POST" class="form">
    {% csrf_token %} {% include 'book/book_form.html' with book=book %}
    <button type="submit" class="btn btn-primary">Save</button>
  </form>
</div>
{% endblock content %}

```

Результат работы программы:



All Books

	<div>latte</div> <div>bloom-n-brew</div> <div>This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.</div> <div>123</div> <div><div>Delete</div><div>Update</div></div>
	<div>capuchino</div> <div>brrrew</div> <div>This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.</div> <div>123123</div> <div><div>Delete</div><div>Update</div></div>
	<div>work pls</div> <div>brrrew</div>

New Book

Book Name

Book Cost

Select Store

brrrew

▼

Create

Update Book

Book Name

Book Cost

Select Store

brrrew

▼

Save