Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет

Лабораторная работа № 7

По курсу «Разработка интернет приложений»

ИСПОЛНИТЕЛЬ:
Группа ИУ5-55Б
Петренко С.С.

"16" декабря 2020 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

_____

"__"_____2020 г.

Москва 2020

1. **Задание**

   Необходимо разработать систему для проведения викторин по российским городам.

   Работа была разделена на несколько частей:

   - Проработка дизайна
   - Разработка фронтенда
   - Разработка бекенда
   - Деплой и дальнейший мониторинг

2. **Разработка фронтенда**

   На стороне фронтенда в проекте используется FLUX архитектура и PWA

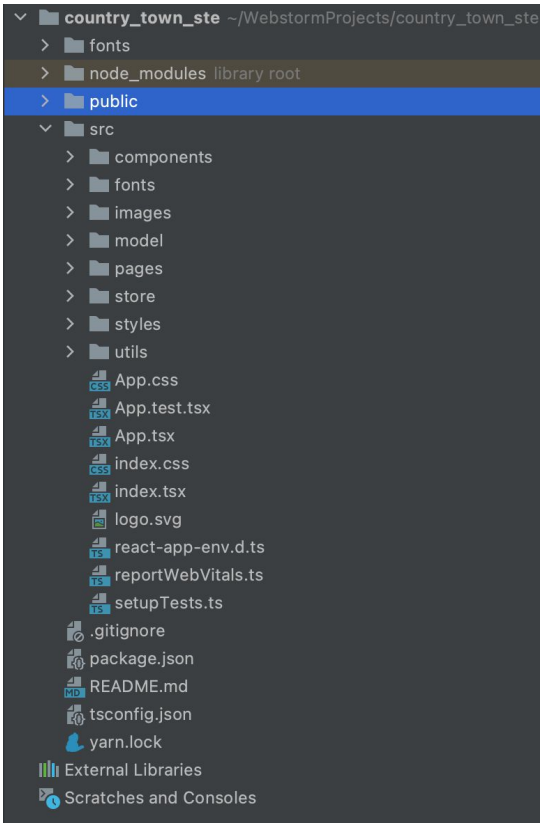
   Этапы разработки фронтенда

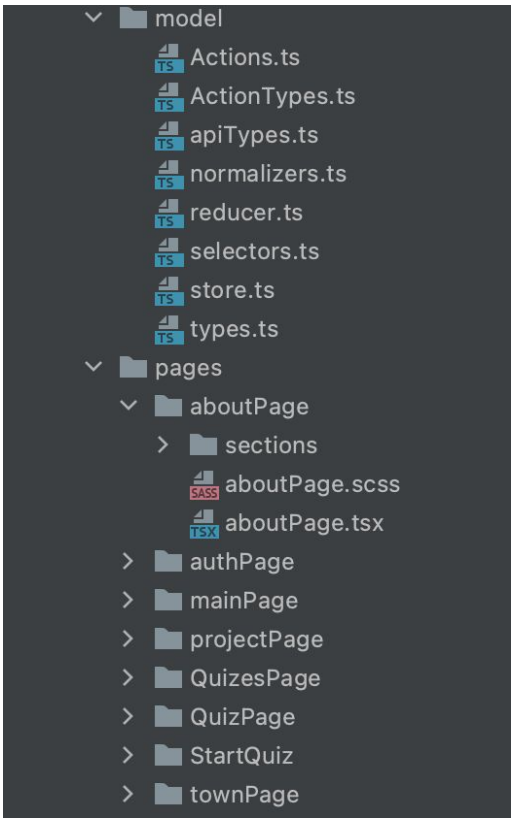
   - Сетап проекта.
   - Проектирование макетов страниц в figma
   - Проектирование базовых элементов интерфейса
   - Проектирование хранилища веб-приложения.
   - Проектирование роутинга в приложении.
   - Верстка страниц и компонентов.
   - Применение стилей для страниц.

   Технологии, используемые в фронтенде:

   - TypeScript — расширенная версия JavaScript со статической типизацией
   - React - Создание пользовательских интерфейсов
   - Redux - Хранилище
   - HTML - Разметка
   - SCSS - Стили

Структура проекта:





3. Разработка бекенда

На стороне бэкенда в проекте используется MVP архитектура.

Этапы разработки бекенда:

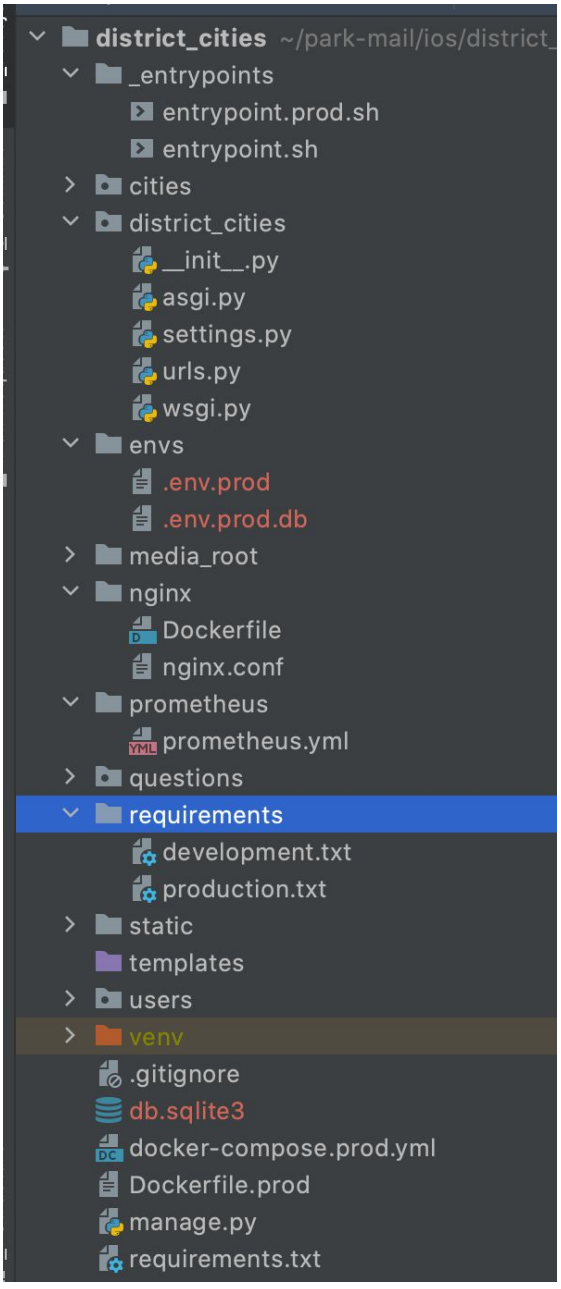
- Проектирование базы данных
- Написание моделей в Django
- Написание сериалайзеров для моделей с помощью Django Rest Framework
- Создание портала для администратора
- Написание view, а также добавление необходимой бизнес-логики в модели

Технологии, используемые в бекенде:

- Django
- Django Rest Framework(DRF)
- PIL(для работы с картинками)
- Django-json-widget(виджет для редактирования json в admin панели)

- Django-prometheus(мониторинг)
- Postgresql(база данных)

Структура проекта:



4. Деплой и мониторинг

После написания бекенда и фронтенда необходимо произвести деплой и настроить удобный мониторинг.

Во время деплоя были использованы следующие технологии

- Nginx для отдачи фронтенда и проксирования запроса на бекенд
- Node_exporter для отдачи информации о состоянии сервера

- UWSGI Gunicorn позволяет запустить несколько процессов с Django для ускорения работы сайта
- Prometheus собирает и хранит все данные о мониторинге из Django и Node_exporter
- Grafana используется для составления дашбордов о работе сайта на основе данных из Prometheus
- Docker-compose позволяет удобно запускать все сервисы приложения одной командой

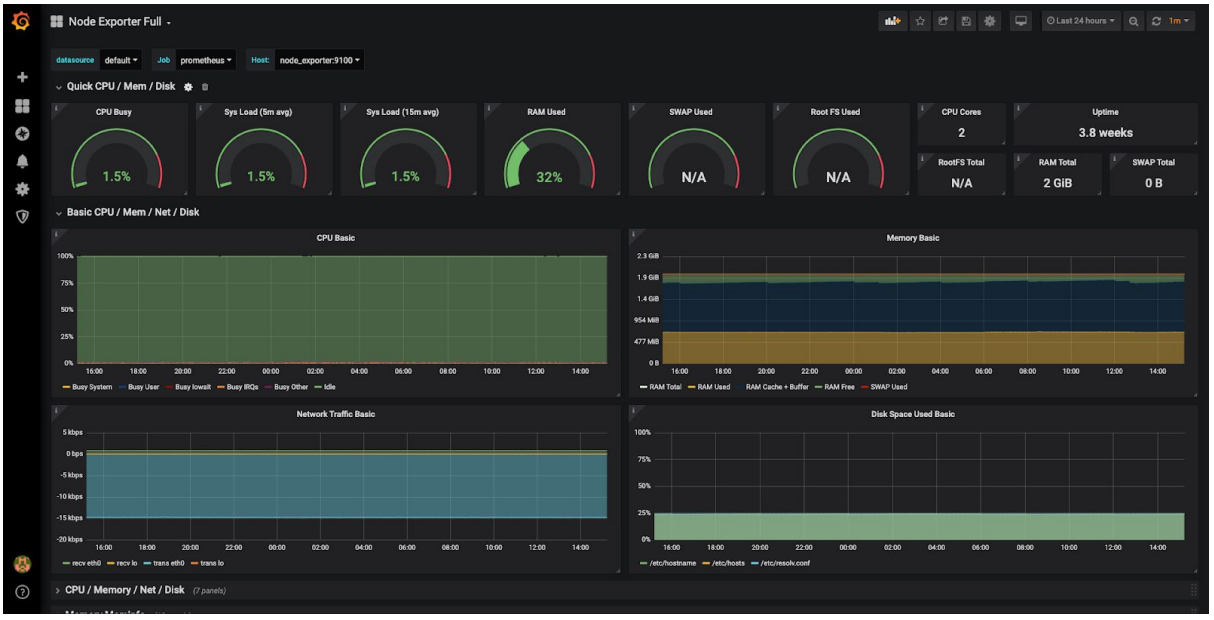
После деплоя и настройки приложения на сервере мы выпустили сертификат для подключения по https и настроили его автоматический перевыпуск с помощью certbot.

Конфигурация docker-compose файла:

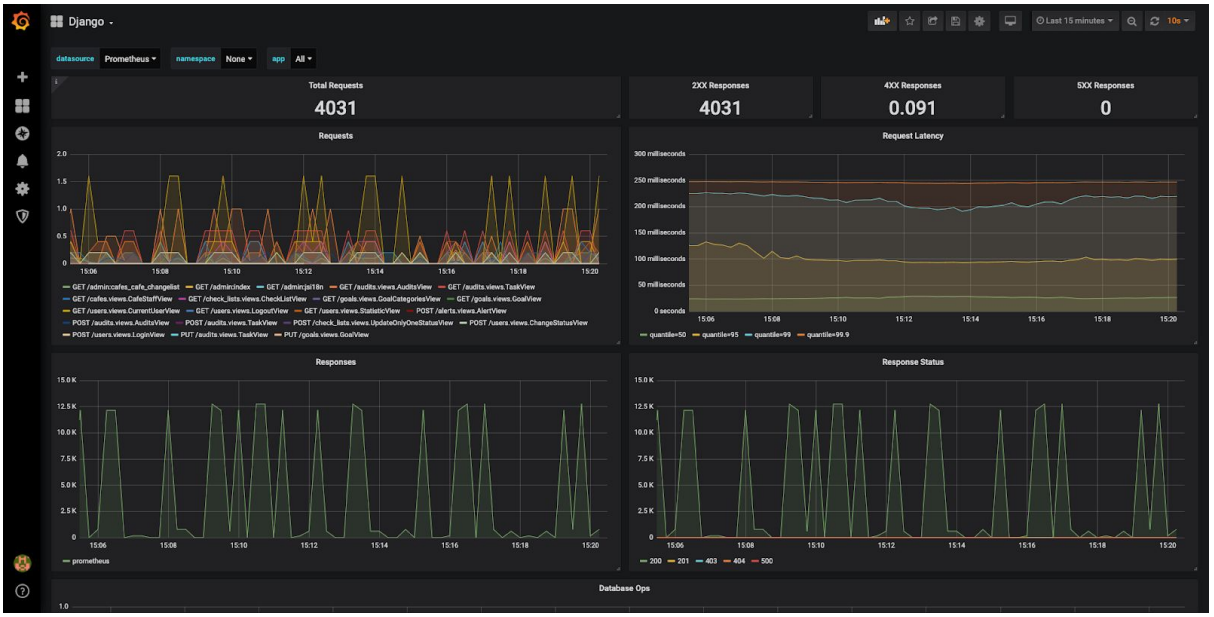
```
version: '3.3'

services:
 prometheus:
   image: prom/prometheus:v2.14.0
   volumes:
     - ./prometheus/:/etc/prometheus/
   restart: always

 grafana:
   image: grafana/grafana:6.5.2
   ports:
     - 3060:3000
   restart: always

 nginx:
   build: ./nginx
   volumes:
     - static_volume:/usr/src/district_cities/static
     - static_volume:/usr/src/district_cities/media_root
   ports:
     - 127.0.0.1:9090:80
   depends_on:
     - web
   restart: on-failure

 web:
   build:
     context: .
     dockerfile: ./Dockerfile.prod
   command: gunicorn district_cities.wsgi --bind 0.0.0.0:8000
--worker-connections=1000 --workers=3
   volumes:
     - static_volume:/usr/src/district_cities/static
     - static_volume:/usr/src/district_cities/media_root
   ports:
     - 127.0.0.1:8080:8000
```

```
    env_file:
      - envs/.env.prod
    depends_on:
      - db
    restart: on-failure

  db:
    image: postgres:12.0-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    env_file:
      - envs/.env.prod.db
    restart: on-failure

  node_exporter:
    image: prom/node-exporter
    ports:
      - 127.0.0.1:9100:9100
    restart: always

volumes:
  postgres_data:
  static_volume:
  frontend_volume:
```

Скриншоты дашборда мониторинга системы:
Состояние сервера(данные node)

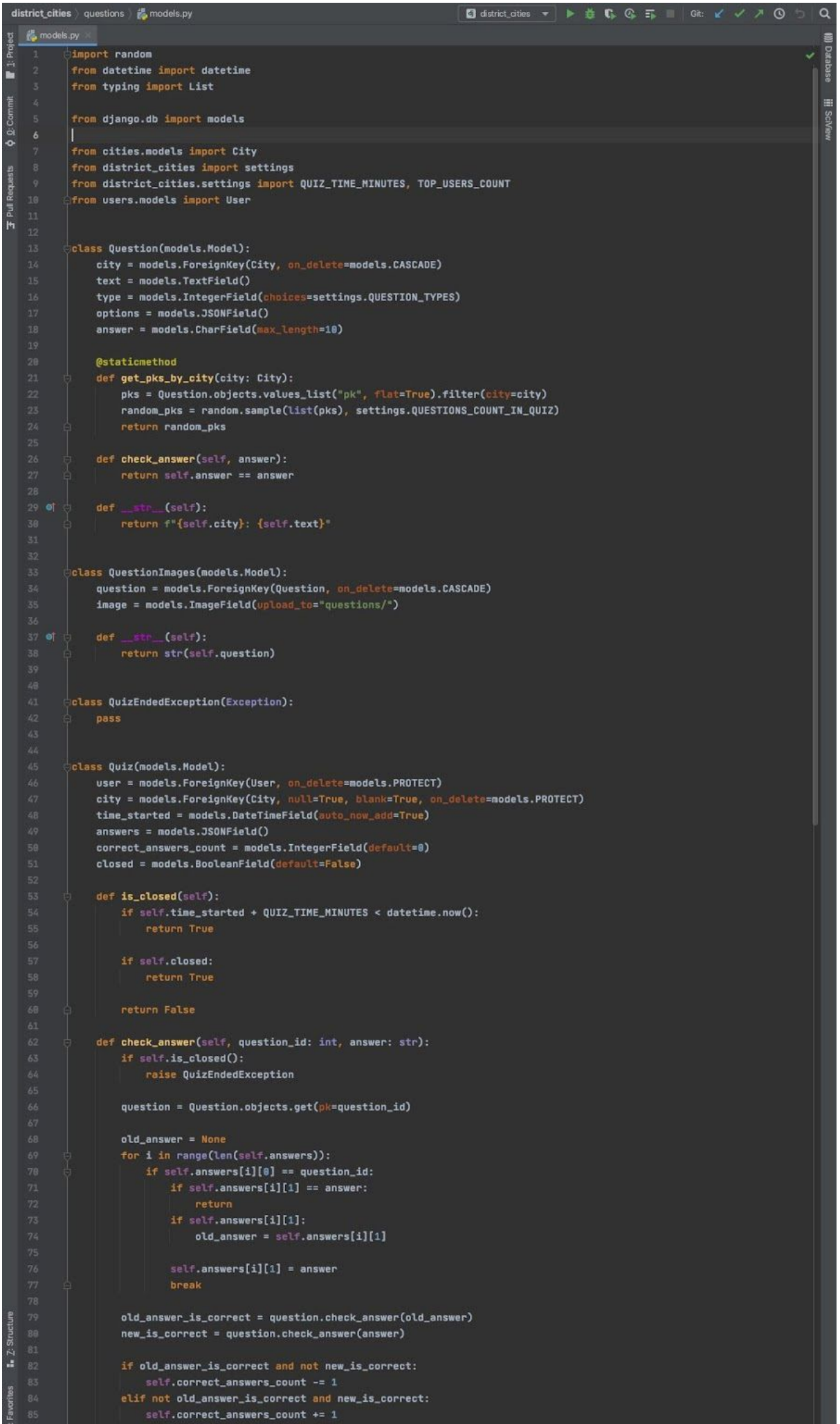


Состояние Django(данные django)

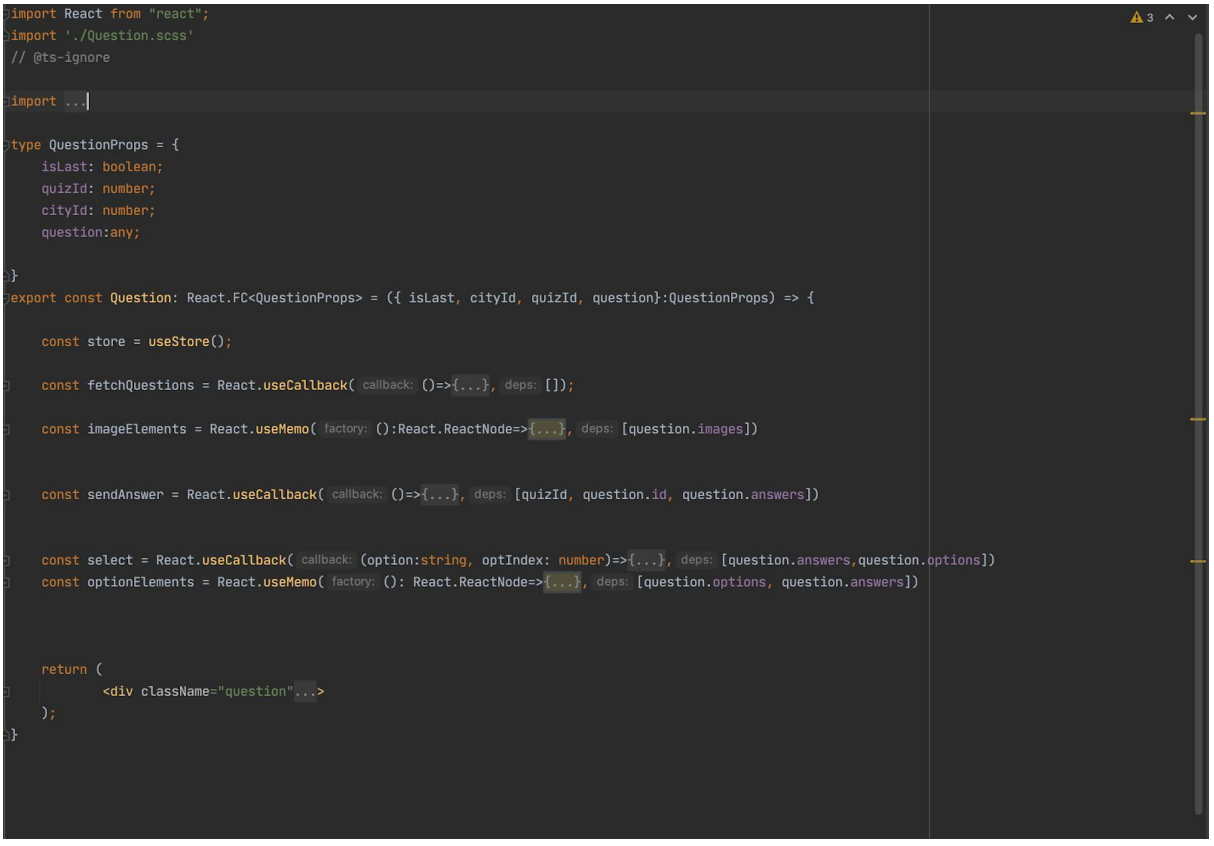


Основные View приложения

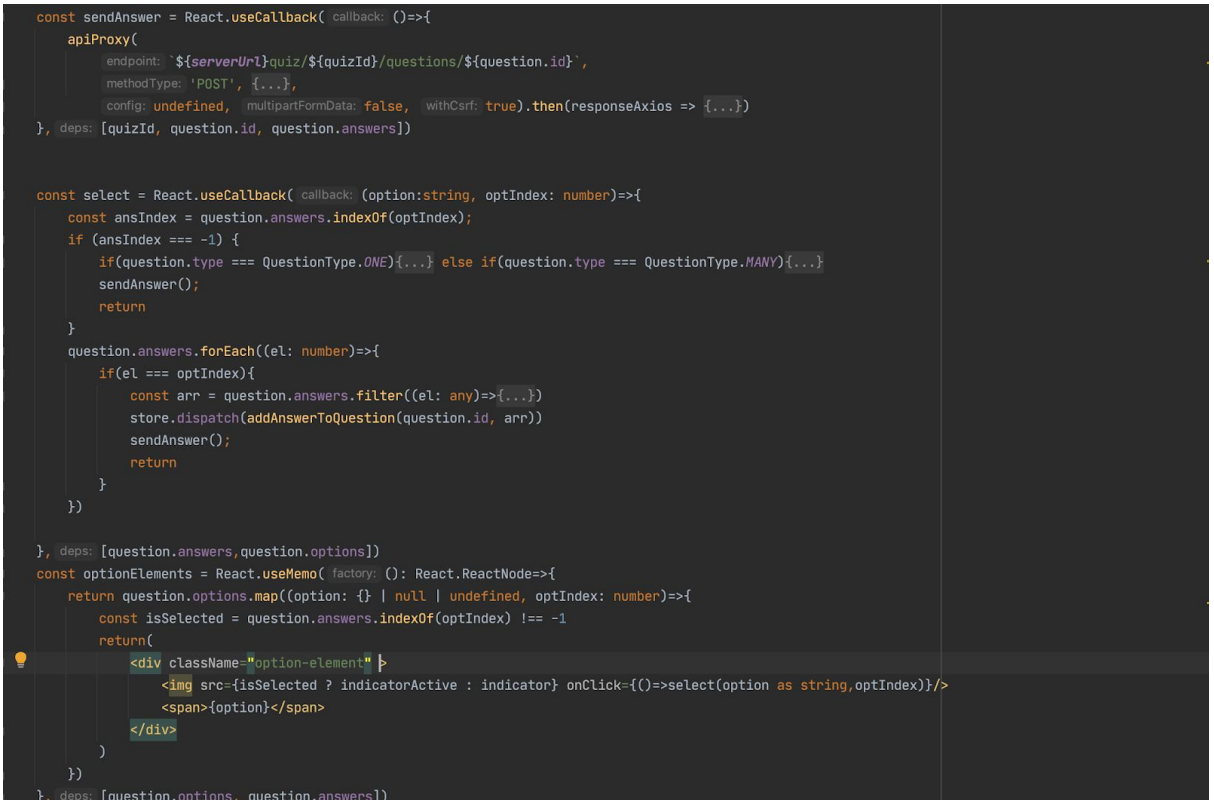

district_cities 〉 questions 〉 models.py
district_cities
Git:
models.py
1: Project
0: Commit
Pull Requests
Database
SciView
Z: Structure
Favorites
for i in range(len(self.answers)):
if self.answers[i][0] == question_id:
if self.answers[i][1] == answer:
return
if self.answers[i][1]:
old_answer = self.answers[i][1]
self.answers[i][1] = answer
break
old_answer_is_correct = question.check_answer(old_answer)
new_is_correct = question.check_answer(answer)
if old_answer_is_correct and not new_is_correct:
self.correct_answers_count -= 1
elif not old_answer_is_correct and new_is_correct:
self.correct_answers_count += 1
self.save()
def stop(self) -> bool:
for answer in self.answers:
if not answer[1]:
return False
self.closed = True
self.save()
return True
@staticmethod
def get_running_quiz(user: User, city: City):
"""
Function checks is there any stated sessions for user and city
"""
now = datetime.now()
quiz: List[Quiz] = Quiz.objects.filter(
user=user, city=city, time_started__lt=now, closed=False,
)
for q in quiz:
if q.time_started + QUIZ_TIME_MINUTES > now:
return q
return None
@staticmethod
def start_new(user: User, city: City):
quiz = Quiz.get_running_quiz(user, city)
if quiz is not None:
return quiz
quiz = Quiz(user=user, city=city)
questions_ids = Question.get_pks_by_city(city)
answers = []
for question_id in questions_ids:
answers.append([question_id, None])
quiz.answers = answers
quiz.save()
return quiz
@staticmethod
def get_top(city_id=None):
if city_id is None:
quiz_top: List[Quiz] = Quiz.objects.all().order_by("-correct_answers_count")[
:TOP_USERS_COUNT
]
else:
quiz_top: List[Quiz] = Quiz.objects.filter(city__pk=city_id).order_by(
"-correct_answers_count"
)[:TOP_USERS_COUNT]
print(quiz_top, "")
users = []
for quiz_session in quiz_top:
users.append([quiz_session.user, quiz_session])
return users
def __str__(self):
return f"{self.user}: {self.city}"


district_cities 〉 questions 〉 models.py
district_cities
Git:
models.py
1: Project
9: Commit
Pull Requests
Database
SciView
Z: Structure
Favorites
import random
from datetime import datetime
from typing import List
from django.db import models
from cities.models import City
from district_cities import settings
from district_cities.settings import QUIZ_TIME_MINUTES, TOP_USERS_COUNT
from users.models import User
class Question(models.Model):
city = models.ForeignKey(City, on_delete=models.CASCADE)
text = models.TextField()
type = models.IntegerField(choices=settings.QUESTION_TYPES)
options = models.JSONField()
answer = models.CharField(max_length=10)
@staticmethod
def get_pks_by_city(city: City):
pks = Question.objects.values_list("pk", flat=True).filter(city=city)
random_pks = random.sample(list(pks), settings.QUESTIONS_COUNT_IN_QUIZ)
return random_pks
def check_answer(self, answer):
return self.answer == answer
def __str__(self):
return f"{self.city}: {self.text}"
class QuestionImages(models.Model):
question = models.ForeignKey(Question, on_delete=models.CASCADE)
image = models.ImageField(upload_to="questions/")
def __str__(self):
return str(self.question)
class QuizEndedException(Exception):
pass
class Quiz(models.Model):
user = models.ForeignKey(User, on_delete=models.PROTECT)
city = models.ForeignKey(City, null=True, blank=True, on_delete=models.PROTECT)
time_started = models.DateTimeField(auto_now_add=True)
answers = models.JSONField()
correct_answers_count = models.IntegerField(default=0)
closed = models.BooleanField(default=False)
def is_closed(self):
if self.time_started + QUIZ_TIME_MINUTES < datetime.now():
return True
if self.closed:
return True
return False
def check_answer(self, question_id: int, answer: str):
if self.is_closed():
raise QuizEndedException
question = Question.objects.get(pk=question_id)
old_answer = None
for i in range(len(self.answers)):
if self.answers[i][0] == question_id:
if self.answers[i][1] == answer:
return
if self.answers[i][1]:
old_answer = self.answers[i][1]
self.answers[i][1] = answer
break
old_answer_is_correct = question.check_answer(old_answer)
new_is_correct = question.check_answer(answer)
if old_answer_is_correct and not new_is_correct:
self.correct_answers_count -= 1
elif not old_answer_is_correct and new_is_correct:
self.correct_answers_count += 1

Реализация компонента на примере “Вопроса викторины”



```tsx
import React from "react";
import './Question.scss'
// @ts-ignore

import ...

type QuestionProps = {
    isLast: boolean;
    quizId: number;
    cityId: number;
    question:any;

}
export const Question: React.FC<QuestionProps> = ({ isLast, cityId, quizId, question}:QuestionProps) => {

    const store = useStore();

    const fetchQuestions = React.useCallback( callback: ()=>{...}, deps: []);

    const imageElements = React.useMemo( factory: ():React.ReactNode=>{...}, deps: [question.images])


    const sendAnswer = React.useCallback( callback: ()=>{...}, deps: [quizId, question.id, question.answers])


    const select = React.useCallback( callback: (option:string, optIndex: number)=>{...}, deps: [question.answers,question.options])
    const optionElements = React.useMemo( factory: (): React.ReactNode=>{...}, deps: [question.options, question.answers])



    return (
            <div className="question"...>
    );
}
```



```tsx
    const sendAnswer = React.useCallback( callback: ()=>{
        apiProxy(
             endpoint: `${serverUrl}quiz/${quizId}/questions/${question.id}`,
             methodType: 'POST', {...},
             config: undefined,  multipartFormData: false,  withCsrf: true).then(responseAxios => {...})
    }, deps: [quizId, question.id, question.answers])


    const select = React.useCallback( callback: (option:string, optIndex: number)=>{
        const ansIndex = question.answers.indexOf(optIndex);
        if (ansIndex === -1) {
            if(question.type === QuestionType.ONE){...} else if(question.type === QuestionType.MANY){...}
            sendAnswer();
            return
        }
        question.answers.forEach((el: number)=>{
            if(el === optIndex){
                const arr = question.answers.filter((el: any)=>{...})
                store.dispatch(addAnswerToQuestion(question.id, arr))
                sendAnswer();
                return
            }
        })

    }, deps: [question.answers,question.options])
    const optionElements = React.useMemo( factory: (): React.ReactNode=>{
        return question.options.map((option: {} | null | undefined, optIndex: number)=>{
            const isSelected = question.answers.indexOf(optIndex) !== -1
            return(
                <div className="option-element" >
                    <img src={isSelected ? indicatorActive : indicator} onClick={()=>select(option as string,optIndex)}/>
                    <span>{option}</span>
                </div>
            )
        })
    }, deps: [question.options, question.answers])
```

Ссылка на проект

http://russian-town.ru