

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Web-разработка

Отчет

Лабораторная работа №2

Выполнил:

Егоров Мичил

Группа

K33401

Проверил:

Говоров А. И.

Санкт-Петербург

2021 г.

Задача

Овладеть практическими навыками и умениями реализации web-сервисов средствами Django 2.2.

Проект: StudyDo

Платформа для совместного обучения и подготовки к ЕГЭ для школьников. Существующие онлайн-платформы не закрывают одну из важнейших потребностей ребёнка – потребность в правильном окружении, в единомышленниках. В различных сообществах многие старшеклассники ищут себе товарищей для совместной подготовки к экзаменам.

Необходимо реализовать следующий функционал:

- Авторизация и аутентификация с выбором школы и подтверждением аккаунта.
- Создание тематической комнаты с возможностью отключения микрофона у участников.
- WebRTC связь между участниками.
- Динамическая доска для рисования.
- Общий чат внутри комнаты.
- Возможность подгружать задания из перечня ЕГЭ по разным предметам.

Ход работы

Сначала отрисуем связь между сущностями (Рис. 1).

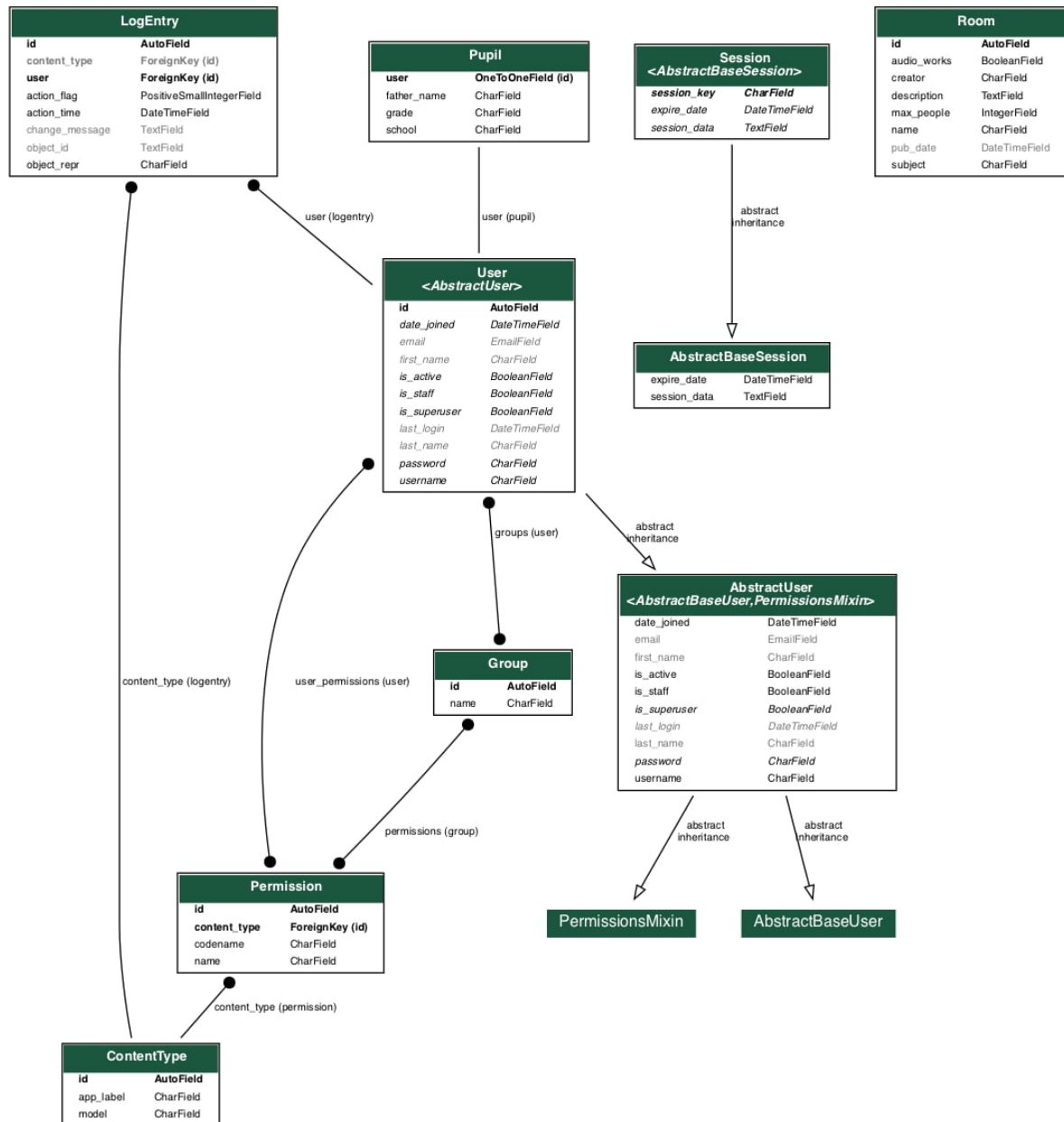


Рисунок 1. Связь между сущностями.

Создадим виртуальное окружение *studydo* и установим нужные библиотеки (список находится в файле `bin/requirements.txt`)

```
$ python -m venv venv  
$ pip install -r bin/requirements.txt
```

Создадим проект

```
$ django-admin startproject studydo
```

Запустим redis для корректной работы Django-channels

```
$ docker run -p 6379:6379 -d redis:5
```

А также приложения `main` и `room`, где будем разрабатывать соответствующие функциональности.

Роуты urls.py

Всего будет три разных категории пути – админская панель, комнаты и все остальное (Рис. 2). Так же подключим роуты для статических и медиа данных. Для комнаты будут пути для создания комнаты, подключения и получения их списка (Рис. 3). Так же, помимо этого, определим пути для авторизации и лендинга (Рис. 4). Наше приложение будет содержать точки для соединения по веб-сокетах: для получения задач внутри комнаты, для доски и чата (Рис. 5).

```
from django.conf.urls.static import static
from django.contrib import admin
from django.conf import settings
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('room/', include('room.urls')),
    path('', include('main.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL, document_root = settings.STATIC_ROOT)
```

Рисунок 2. Роутеры проекта.

```
from django.urls import path

from room import views

urlpatterns = [
    path('list', views.room_list, name='room-list'),
    path('create', views.create_room, name='create-room'),
    path('<int:id>', views.room, name='room'),
]
```

Рисунок 3. Роутеры приложения "room".

```

from django.urls import path

from main import views

urlpatterns = [
    path('', views.index, name='index'),
    path('accounts/login/', views.signin, name='login'),
    path('accounts/register/', views.register, name='register'),
    path('logout/', views.logout_view, name='logout'),
]

```

Рисунок 4. Роутеры приложения "main".

```

from django.urls import re_path
from django.conf import settings

from . import consumers

websocket_urlpatterns = [
    re_path(r'^ws/chat/(?P<room_name>\w+)/$', consumers.ChatConsumer.as_asgi()),
    re_path(r'^ws/tasks/(?P<room_name>\w+)/$', consumers.TasksConsumer.as_asgi()),
    re_path(r'^ws/board/(?P<room_name>\w+)/$', consumers.BoardConsumer.as_asgi()),
]

```

Рисунок 5. Роуты для веб-сокетов.

Контроллеры views.py и отображения

Для авторизации нужно будет ввести личные данные, школу и текущий класс, так же согласиться на обработку данных (Рис. 7). При регистрации проверим логин на уникальность, потом обработаем данные из формы и далее пользователь окажется на странице со спискам комнат (Рис. 8).

Войти

127.0.0.1:8000/accounts/register/

StudyDo Главная

Регистрация

Уже зарегистрированы? [Войти](#)

Фамилия

Имя

Отчество

Логин

Пароль

Школа

Класс

☐ Соглашаюсь на обработку данных

[Зарегистрироваться](#)

Рисунок 6. Страница регистрации.

```
def register(request):
    if request.method == "POST":
        username = request.POST['username']

        user = User.objects.filter(username=username)

        if len(user) == 0:
            pupil = Pupil.from_post(request.POST)
            login(request, pupil.user)

            return redirect(reverse_lazy('room-list'))
        else:
            error_text = 'Пользователь с таким логином уже существует'

    return render(request, 'register/register.html', locals())
```

Рисунок 7. Контроллер регистрации.

Для отображения списка комнат воспользуемся пагинацией и фильтрацией (Рис. 8). На одной странице будем показывать 10 комнат (Рис. 9). Чтобы посмотреть на комнаты потребуется авторизация.

```
@login_required
def room_list(request):
    subject = request.GET.get('subject', '')
    subject_rooms = Room.objects.filter(subject__icontains=subject).order_by('-pub_date')

    paginator = Paginator(subject_rooms, 10)
    page_number = request.GET.get('page', 1)
    room_list = paginator.get_page(page_number)
    subjects = Room.SUBJECTS

    return render(request, 'room/room-list.html', locals())
```

Рисунок 8. Контроллер для получения списка комнат.

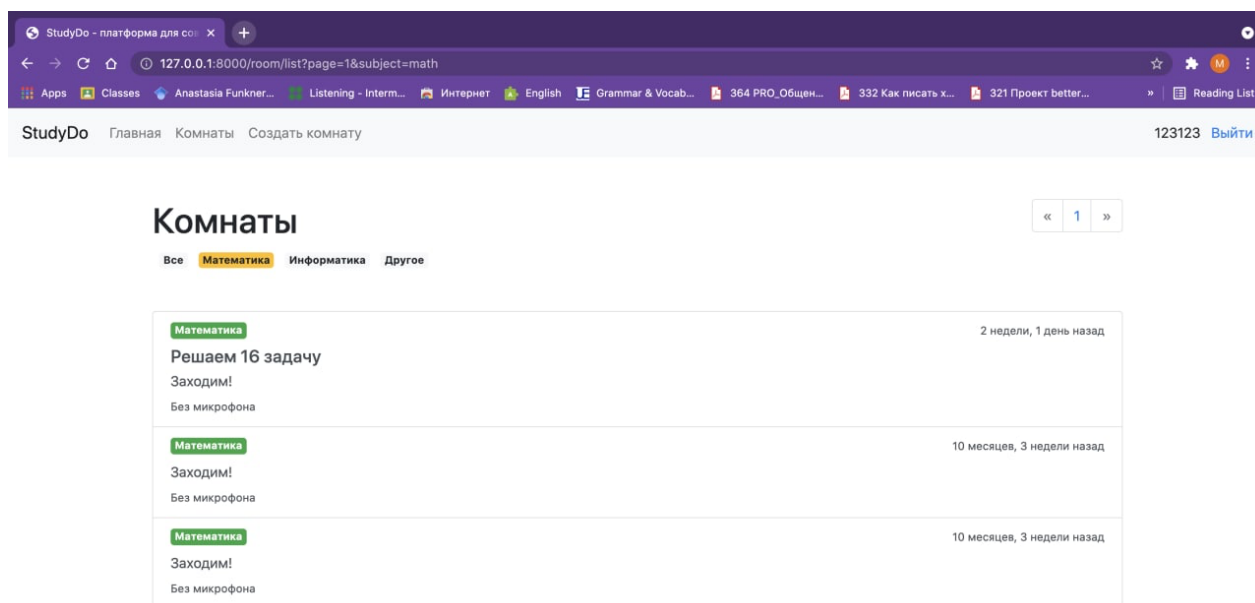


Рисунок 9. Страница списка комнат.

При создании комнаты нужно ввести название комнаты, выбрать предмет, описать зачем комната создается и при желании разрешить использование микрофона внутри комнаты (Рис. 10). Контроллер очевидный.

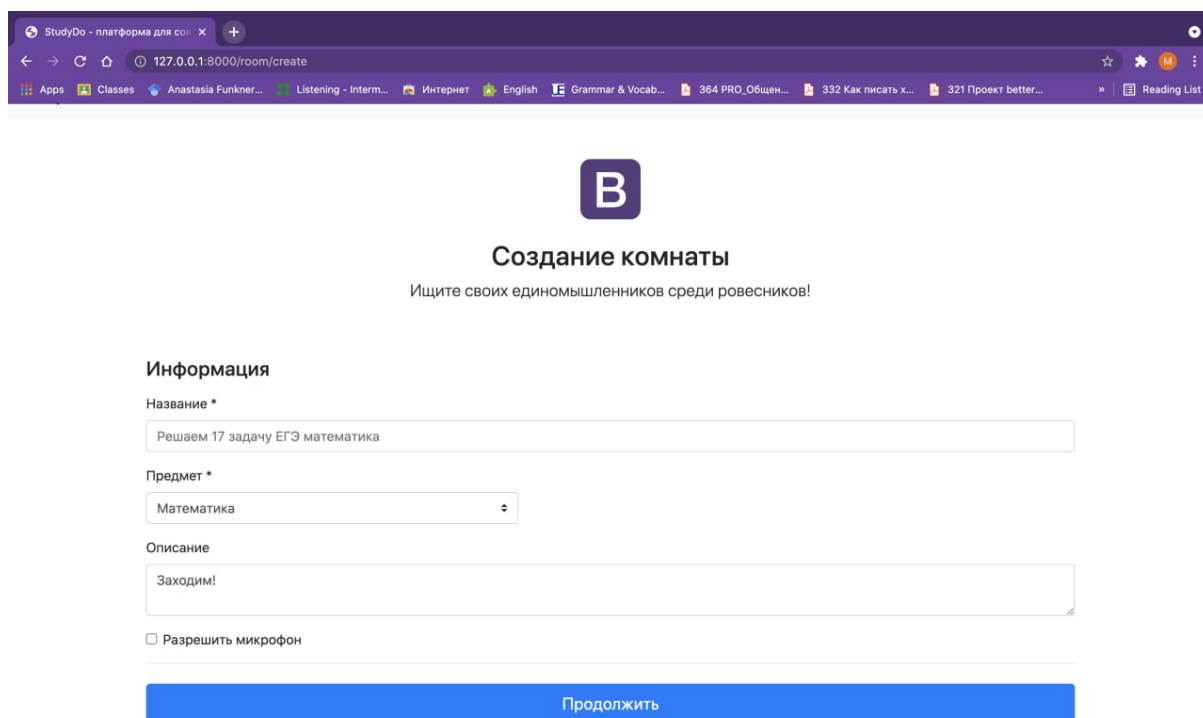


Рисунок 10. Создание комнаты.

Внутри комнаты можно включить/выключить видео, порисовать на доске, пообщаться с участниками комнаты и загрузить задачи (Рис. 11)

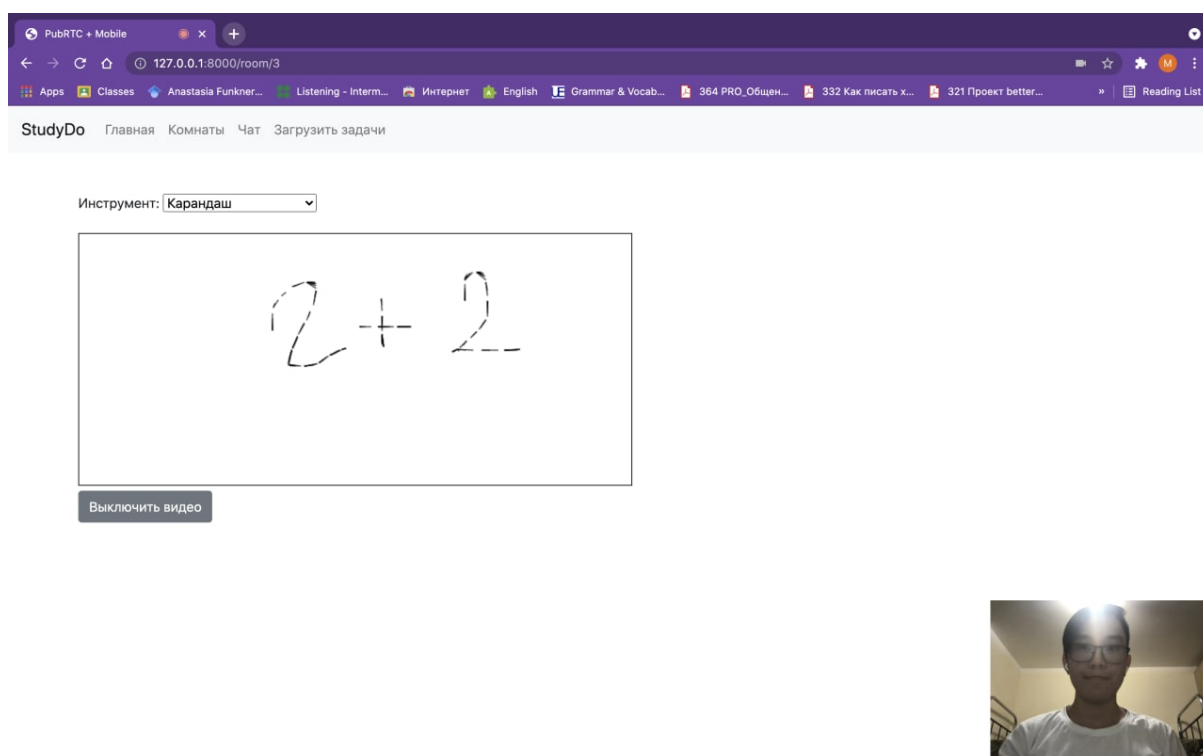


Рисунок 11. Интерактивная комната.

Получения списка задач происходит рандомно по выбранным категориям: биология, математика, информатика или русский язык. Варианты парсятся с сайта sdamgia.ru. Метод, выбирающий задачу показан на рисунке 12.

```
def get_tasks(self, subject, grade):
    test_ids = self.SUBJECTS_GRADES_IDS[subject][grade]

    url = 'https://%s-ege.sdamgia.ru/test?id=%s&nt=True&pub=False'
    r = requests.get(url % (subject, random.choice(test_ids)))
    soup = bs4.BeautifulSoup(r.content)
    c = soup.find_all('div', {'class': 'prob_view'})[1]

    html = ''

    for c in soup.find_all('div', {'class': 'prob_view'}):
        html += str(c).replace('src="/', 'src="https://ege.sdamgia.ru/')

    return html
```

Рисунок 12. Метод, возвращающий задачу из банка sdamgia.ru.

Мышка, при передвижении по доске отправляет координаты во веб-сокеты для остальных участников (Рис. 13-14). Почти аналогично работает чат.

```

class BoardConsumer(WebSocketConsumer):
    def connect(self):
        pass

    def disconnect(self, close_code):
        # Leave room group
        pass

    # Receive message from WebSocket
    def receive(self, text_data):
        text_data_json = json.loads(text_data)

        # Send message to room group
        async_to_sync(self.channel_layer.group_send)(
            self.room_group_name,
            {
                'type': 'chat_message',
                'x': text_data_json['x'],
                'y': text_data_json['y'],
                'x0': text_data_json['x0'],
                'y0': text_data_json['y0'],
            }
        )

    def chat_message(self, event):
        # Send message to WebSocket
        self.send(text_data=json.dumps({
            'x': event['x'],
            'y': event['y'],
            'x0': event['x0'],
            'y0': event['y0'],
        })))

```

Рисунок 13. Получение и отправка точек внутри доски.

```

function drawPencil(x0, y0, x, y, is_send=false){
  if(Math.abs(y-y0) + Math.abs(x - x0) > 20) return;

  context.beginPath();
  context.moveTo(x0, y0);
  context.lineTo(x, y);
  context.stroke();
  context.closePath();
  context.save();

  const w = canvas.width;
  const h = canvas.height;

  if(is_send && x0 && y0){
    boardSocket.send(JSON.stringify({
      'type': 'pencil',
      'x0': x0 / w,
      'y0': y0 / h,
      'x': x / w,
      'y': y / h,
    }));
  }
}

```

Рисунок 14. Отправка точек при рисунке на доске.

Вывод

Реализовали web-сервис для совместных занятий StudyDo с помощью фреймворка Django и Django-Channels. Определили модели, роуты для страниц и их контроллеры, придерживались определения MVC.