Федеральное государственное автономное образовательное учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

**Факультет информатики, математики и компьютерных наук**

**КУРСОВАЯ РАБОТА**

Разработка социальной сети на Python

по направлению подготовки 09.03.04 "Программная инженерия"

образовательная программа «Программная инженерия»

Выполнил:

Студент группы 19ПИ-2

Фатин Максим Романович

Руководитель:

Кащеев Николай Иванович

Доцент, кандидат технических наук

Нижний Новгород 2021

**Содержание**

1. **Титульный лист…………………………………………………………...1**
2. **Содержание………………………………………………………………...2**
3. **Введение……………………………………………………………………3**
4. **Постановка задачи……………………………………………………..…3**
5. **Обзор существующих решений…………………………………………4**
6. **Описание решения………………………………………………...……...6**
7. **Заключение…………………………………………………………….…26**
8. **Литература……………………………………………………………….27**
9. **Репозиторий………………………………………………………………27**
10. **Введение**

В современном мире одним из лучших способов получения знаний является наставничество. Такой подход имеет следующие преимущество перед самостоятельно изучением:

* Актуальная и полезная информация, которая точно будет полезна в практике
* Ментор составит для вас индивидуальную программу обучения
* Изучение предмета со специалистом на практике, а не только в теории

Для выбора наставника на сегодняшний день есть множество различных сервисов, готовых помочь с подбором наставника, но у большинства есть значительные недостатки. Основной из них – не полная информация, предоставляемая сайтом о преподавателе, чтобы ученик не мог связаться с ним, за пределами данного сайта (сайтам это важно так как большинство из них взымает комиссию с преподавателей за каждого нового ученика). Данное ограничение не позволяет полностью оценить навыки репетитора, а в последствии, сотрудничество может не привести заказчика к нужному результату по какой-либо причине.

Цель работы:

Разработать веб – приложение для поиска репетитора

Задачи:

1. Разработка frontend и backend части приложения
2. Тестирование приложения
3. **Постановка задачи**

**4.1 Требование к реализуемому приложению**

Разработать программу на языке Python с web-интерфейсом с целью создания социальной сети для поиска преподавателя. Формат социальной сети будет решать проблему недостаточной коммуникации преподавателя и ученика перед началом сотрудничества и поможет двум сторонам лучше узнать друг друга.

Приложение должно иметь следующий функционал:

1. Возможность смотреть последние посты преподавателей (общая лента новостей и по переделённой теме)
2. Возможность подписываться на публикации выбранных людей
3. Регистрация / авторизация на сайте
4. Возможность бронировать занятие с преподавателем по выбранному предмету
5. Возможность отменять занятия
6. Возможность писать комментарии для постов
7. Реализовать базовый API для дальнейшей интеграции

Данная социальная сеть будет построена на основе идеи блога. Преподаватель может размещать в профиле посты о своих работах, успехах учеников, рассказ о своём опыте, методике преподавания и личных достижениях.

Зарегистрированный пользователь имеет возможность оформить подписку на выбранного автора, а также забронировать занятие, если пользователь имеет статус “Учитель”.

Стать учителем можно только через обращение к администрации сайта, чтобы проверить квалификацию кандидата и в случаи успеха предоставить соответствующие права на сайте.

Коммуникация на сайте производится по средствам комментирования постов, а также письма на почту (на сайте почта используется как подпись под каждым постом).

1. **Обзор существующих решений**

**5.1 Профи (profi.ru)**

Данное приложение пользуется наибольшим спросом среди русскоязычной аудитории и имеет большую базу преподавателей по различным направлениям подготовки.

**Положительные стороны сервиcа:**

* Сайт предоставляет возможность подбирать преподавателя менеджеру и клиенту не нужно самому долго изучать анкеты преподавателей.
* На сайте имеется множество профессионалов, которые работаю как в онлайн, так и в офлайн.
* Преподаватели сами заинтересованы в отклике на заказ и клиенту лишь остаётся выбрать лучшее предложение.
* Сайт не имеет специализации на одном предмете и имеет обширную базу выбора предоставляемой услуги.

**Стороны требующие доработки:**

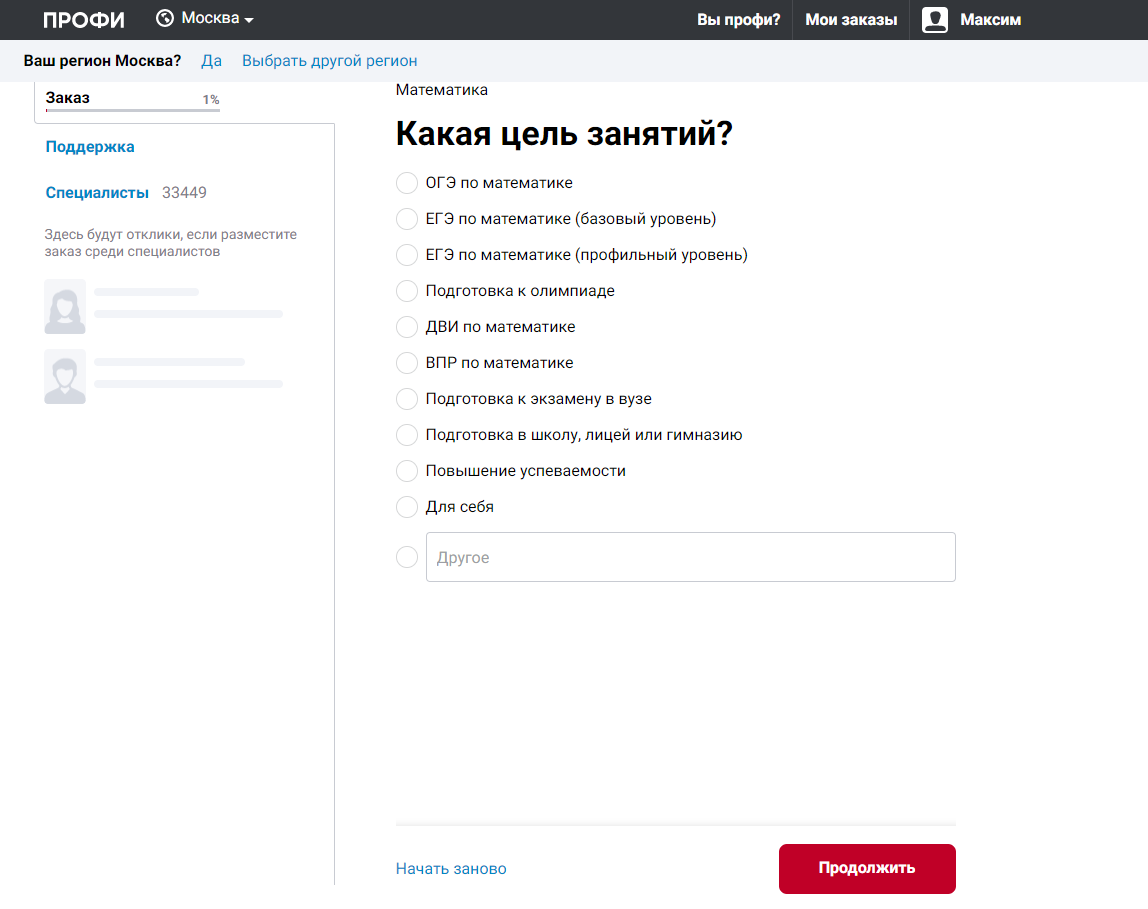
* Сервис не предоставляет удобного процесса оплаты и просмотра предстоящих занятий
* На сайте строгие требования по общению с клиентом, поэтому у преподавателей нет возможности разместить свои публикации с ссылкой на источник.
* Не очень приятный интерфейс. (Рисунок 1)

Рисунок 1

**5.2 Italki**

Сервис для подбора преподавателя для обучения иностранному языку. Отличительно особенность является наличие преподавателей со всего мира, что позволяет заниматься с носителем языка.

**Положительные стороны сервиcа:**

* Сервис содержит удобную систему оплаты занятий.
* Платформа очень дружелюбна к новым преподавателям и готова продвигать их.
* Платформа требует загрузки видео от преподавателя, что делает её более живой и позволяет хорошо узнать преподавателя до начала занятия.
* Присутствует расписание, в котором можно посмотреть забронированные занятия или же отменить их.
* Есть возможность общения с преподавателем на сервисе по средствам переписки.

**Стороны требующие доработки:**

* Нет менеджера, что обязует пользователя самому разбираться с устройством сервиса и подбирать нужного преподавателя придётся самому.
* Сервис специализируется только на изучении языков и не подойдёт для подбора преподавателя в другой области.
* Ученик должен искать преподавателя, а не преподаватели предлагаю свои услуги (это делает сервис менее навязчивым, но требует много времени на подбор нужного преподавателя).
* Взымается комиссия с каждой покупки.

**5.3 GetMentor (https://getmentor.dev/)**

Данный сервис не очень популярен, но имеет достаточно мало аналогов так как полностью бесплатный как для менторов, так и для учеников. Сервис специализируется для нахождения ментора по программированию.

**Положительные стороны сервиcа:**

* Не взымается комиссия за обмен контактами с ментором.
* Очень много высококвалифицированных специалистов из больших компаний.
* Некоторые менторы готовы помогать бесплатно.

**Стороны требующие доработки:**

* Только самостоятельный подбор менторов.
* Специализированный сайт только для поиска ментора-программиста.
* Долгое взаимодействие с ментором (Ментор может просто проигнорировать ваше обращение к нему, и платформа сообщит об этом только через некоторое время).
* Отсутствует система бронирования и оплаты услуг.

**5.4 Вывод**

Проанализировав различные сервисы подбора преподавателей, я cакцентирую внимание на следующих деталях в реализации своего приложения:

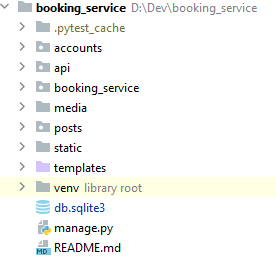
* Система бронирования и просмотр предстоящих занятий
* Общение преподавателя и ученика

1. **Описание решения**

**6.1 Стек технологий**

Для создания приложения я решил использовать язык программирования Python. В приложении используются следующие фреймворки для back-end части приложения: Django и Django REST framework. Для front-end части будут использоваться Bootstrap и jQuery.

* 1. **Структура проекта**



* Posts – приложение, отвечающее за посты пользователей и их коммуникацию в рамках одного поста, а так же за подписки пользователей.
* Api – приложение, служащее для интеграции с другими сервисами
* Accounts – приложение, отвечающее за аккаунт пользователя и сервис бронирования занятий.
* Booking\_service – приложение, содержащее основные настройки для проекта
* Static – папка, содержащая статические элементы страниц (такие как CSS и JavaScript)
* Templates – папка, содержащая все html файлы
* Media – папка, содержащая кеш приложения и все имеющиеся изображения для постов.

**6.3 Общий принцип работы приложений**

Urls.py

URL – адрес

URL – адрес

База Данных

views.py

HTML

Рисунок 2

На втором рисунке проиллюстрирована общая схема работы приложения. Точкой входа в приложения является url. Как только пользователь переходит на сайт, данные поступают в обработку (*urls.py*), где система определяет какую функцию необходимо вызвать (*views.py*). В свою очередь функция может брать или записывать данный в базу данных и в случае успешного завершения функции, отрисовывать пользователю страницу с нужными данными.

**6.4 База данных**

Рассмотрим в первую очередь устройство базы данных. Всего имеется 2 файла для создания базы данных на основе Django – ORM (данный инструмент очень удобен и не требует самому писать SQL запросы) все эти файлы называются *models.py* и располагаются внутри следующих приложений: *accounts*, *posts*. Рассмотрим каждое приложение более подробно

1. Приложение Posts отвечает за реализацию публикаций, группы, подписки и комментарии и содержит следующие таблицы: *Group, Post, Follow, Comment*.

* Модель Group отвечает за принадлежность каждого поста к какой либо группе (#it, #english и так далее). Данные группы нужны для сортировки постов по интересам пользователей. Рассмотрим устройство базы данных для модели Group:
  + Title – Заголовок группы, поле типа CharField с максимальной длинной - 200
  + Slug – Уникальное короткое имя группы, для отображения в url, поле типа SlugField
  + Description – поле типа TextField для описания
* Модель Post отвечает за создание поста и имеет следующие поля:
  + Group – поле типа ForeignKey, связанное моделью Group, по умолчанию имеет тип NULL и является не обязательным
  + Text – поле типа TextField и содержит текст поста
  + Pud\_date – поле типа DateTimeField, отвечает за даты публикации
  + Author – поле типа ForeignKey и связано с моделью User
  + Image – поле типа ImageField, которые ссылается на картинку при наличии её в посте
  + Так же модель имеет Meta class, позволяющий сортировать посты по дате публикации
* Модель Follow отвечает за подписки пользователя, чтобы отображать посты выбранных авторов на странице подписок. Модель имеет следующие поля:
  + Author – поле типа ForeignKey и связано с моделью User
  + User – поле типа ForeignKey и связано с моделью User
  + Так же модель имеет Meta class, гарантирующий уникальность записи по полям Author, User
* Модель Comment отвечает за комментарии пользователей и отображением их под нужным постом. Модель имеет следующие поля:
  + Post - поле типа ForeignKey, связанное моделью Post и указывает на пост, к которому отностится комментарий
  + Author – поле типа ForeignKey и связано с моделью User
  + Text – поле типа TextField и содержит текст комментария
  + Created – поле типа DateTimeField, и нужно для определения даты публикаци
  + Так же можель имеет Meta class, позволяющий сортировать комментарии по дате их написания

1. Приложение Accounts отвечает за основную информацию о пользователе, запланированных уроках и дисциплинах, преподаваемых учителями. Данный файл содержит следующие таблицы: *User, Lesson, DisciplineDescription*.

* Модель User служит для хранения основных данных о пользователе и содержит следующие поля:
  + Email – поле с уникальным значением и имеет тип EmailField
  + Name, Surname – поля типа CharField с ограничением длинны в 255 символов, не могут быть пустыми и обязательны к заполнению
  + Break\_time – поле типа IntegerField и указывает на врем перерыва между уроками
  + Is\_admin, Is\_stuff – базовые, обязательные поля при переопределении встроенной модели User
  + Role – поле типа CharField, имеющую опцию choices = UserRoles.choices (STUDENT, TEACHER, ADMIN)
  + Timestamp – поле типа DateTimeField и указывает дату регистрации
* Модель Lesson служит для хранения информации о проведённых и предстоящих занятиях имеет следующие поля:
  + Duration – поле типа IntegerField, указывающую длину занятия (длительность занятия должна быть кратна 15 и измеряется в минутах)
  + Is\_cancaled – поле типа BooleanField
  + Is\_payed – поле типа BooleanField
  + Student\_pk – поле типа ForeignKey, связанное моделью User и указывает на ученика
  + Teacker\_pk – поле типа ForeignKey, связанное моделью User и указывает на учителя
  + Start\_time – поле типа FateTimeField и указывает на время начала урока
  + Name – поле типа CharField с ограничением в 50 символов и содержит имя дисциплины, по которой будет проводится занятие
  + Price – поле типа IntegerField и указывает на цену занятия
  + Так же модель имеет Meta class, сортирующий записи по времени начала занятий
* Модель DisciplineDescription отвечает за хранение информации о имени дисциплины у конкретного учителя, цены занятия и его времени и содержит следующие поля:
  + Teacher\_pk – поле типа ForeignKey, связанное моделью User и указывает на учителя
  + Name – поле типа CharField и указывает на имя дисциплины
  + Duration – поле типа IntegerField и указывает на длительность занятия
  + Price – поле типа IntegerField, указывающее на цену занятия

**6.5 Основные urls**

Ниже представлен список основных страниц, присутствующих в приложении.

1. Регистрация
2. Авторизация
3. Все посты
4. Создание поста
5. Просмотр избранных авторов
6. Группы
7. Бронирование занятия
8. Расписание занятий

Рассмотрим каждую страницу отдельно

* **Регистрация**

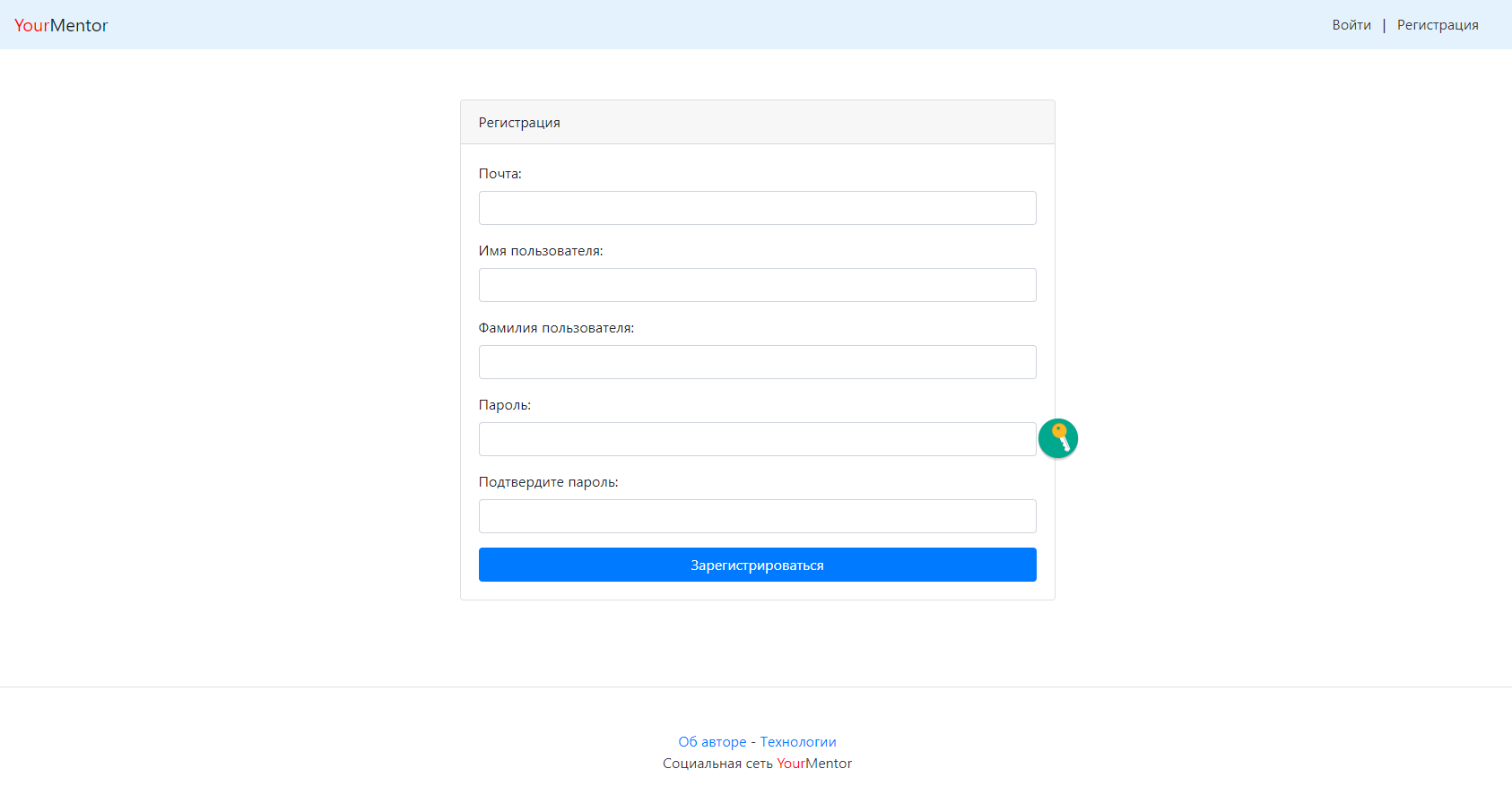
1. **Возможности не авторизованного пользователя:**

Просмотр общей ленты новостей, просмотр профилей других пользователей.

1. **Возможности авторизованного пользователя:**

Регистрация позволяет пользователю бронировать занятия у пользователя сети, если тот имеет статус “учитель”, позволяет оставлять комментарии и подписываться на понравившихся авторов. Остальная часть функционала совпадает с неавторизованным пользователем.

1. **Дизайн**



1. **Валидация данных заполнения формы регистации**

Перед регистрацией пользователя обязательно проходит проверка предоставленных данных. Проверка состоит из следующих стадий:

* Корректность e-mail адреса
* Проверка, что все поля заполнены
* Проверка простоты пароля
* Проверка совпадения паролей

В случае, если одно из требований не выполнено, высветятся подсказки (Рисунок 3)

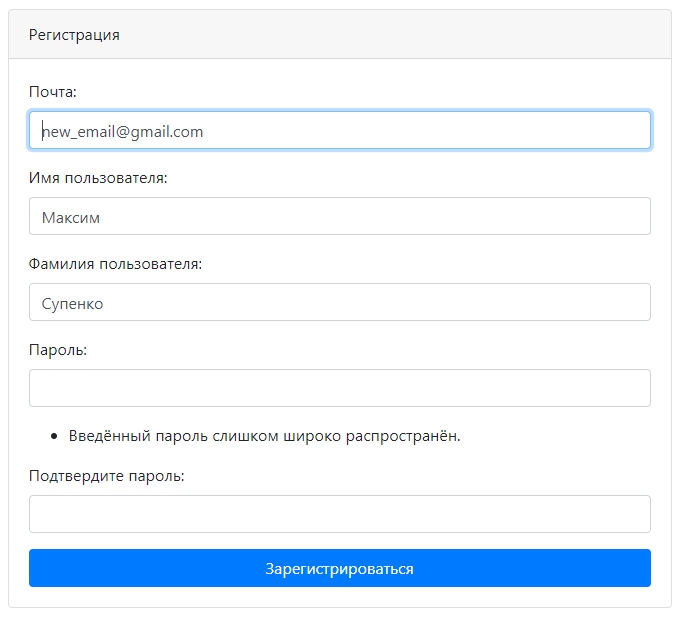


Рисунок 3

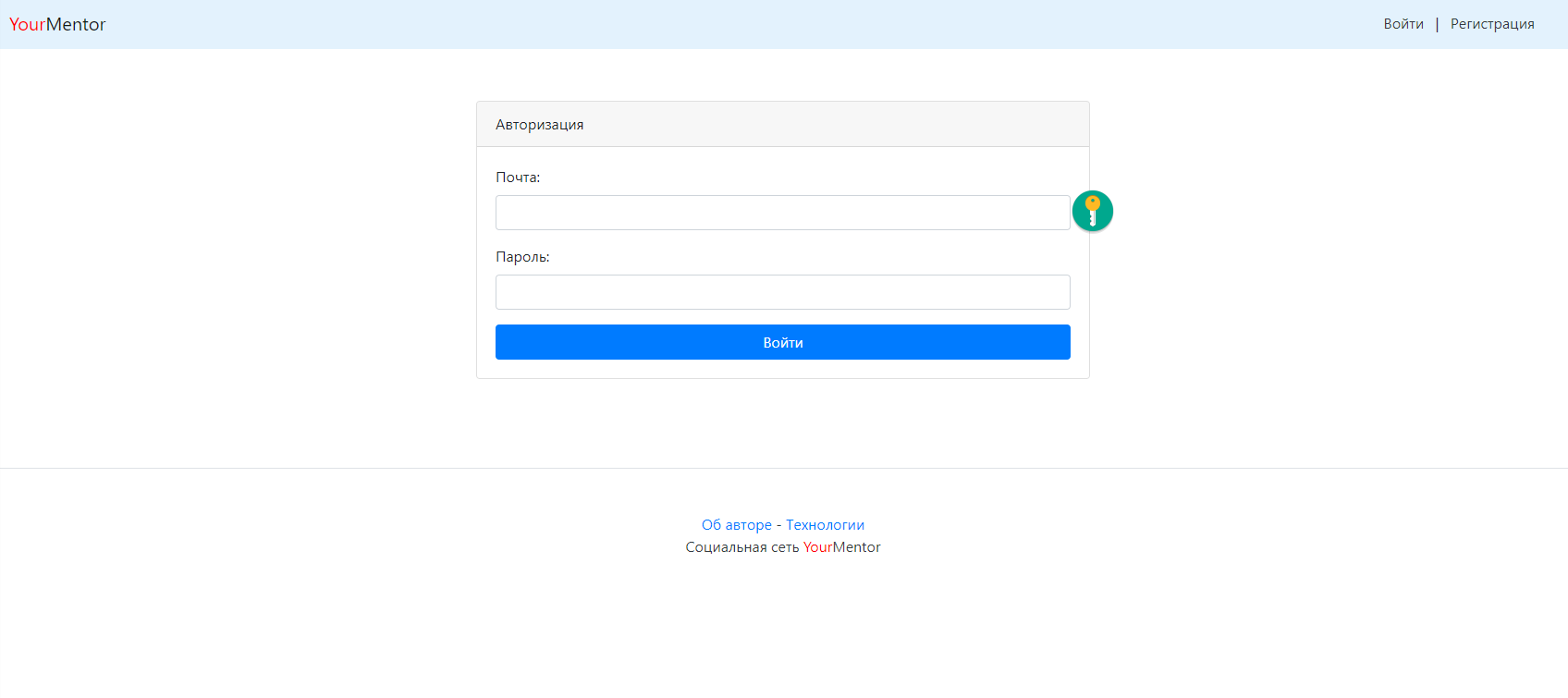
1. **Код**

Back-end часть регистрации находится в файле *views.py* в приложении *accounts.* Логика регистрации основана на view-классе *SignUp* и благодаря наследованию от *CreateView* реализует основную логику самостоятельно.

Front-end часть расположена в папке *templates* и называется *register.html*

* **Авторизация**

1. **Дизайн**



1. **Код**

Back-end часть регистрации находится в файле *views.py* в приложении *accounts.* Логика регистрации основана на view-функции *user\_login.*

Front-end часть расположена в папке *templates* и называется *login.html*

* **Просмотр постов**

1. **Основной функционал и назначение**

На данной странице сайта отображаются последние посты, выложенные авторами, что служит прямой рекламой преподавателей. Если ментор заинтересован в новых учениках, то ему нужно как можно чаще выкладывать образовательный контент, для привлечения новых учеников.

1. **Основные сценарии взаимодействия**

Данная страница имеет большое количество сценариев дальнейшего использования. Ниже рассмотрены основные из них:

* Переход на страницу избранных авторов (Рисунок 4 – пункт 1)
* Переход в группу (Рисунок 4 – пункт 2)
* Переход на страницу редактирования (только для постов, которые относятся к текущему автору) (Рисунок 4 – пункт 3)
* Переход на страницу автора (Рисунок 4 – пункт 4)
* Переход на отдельную страницу поста для добавления или чтения комментариев (Рисунок 4 – пункт 5)

1. **Дизайн**

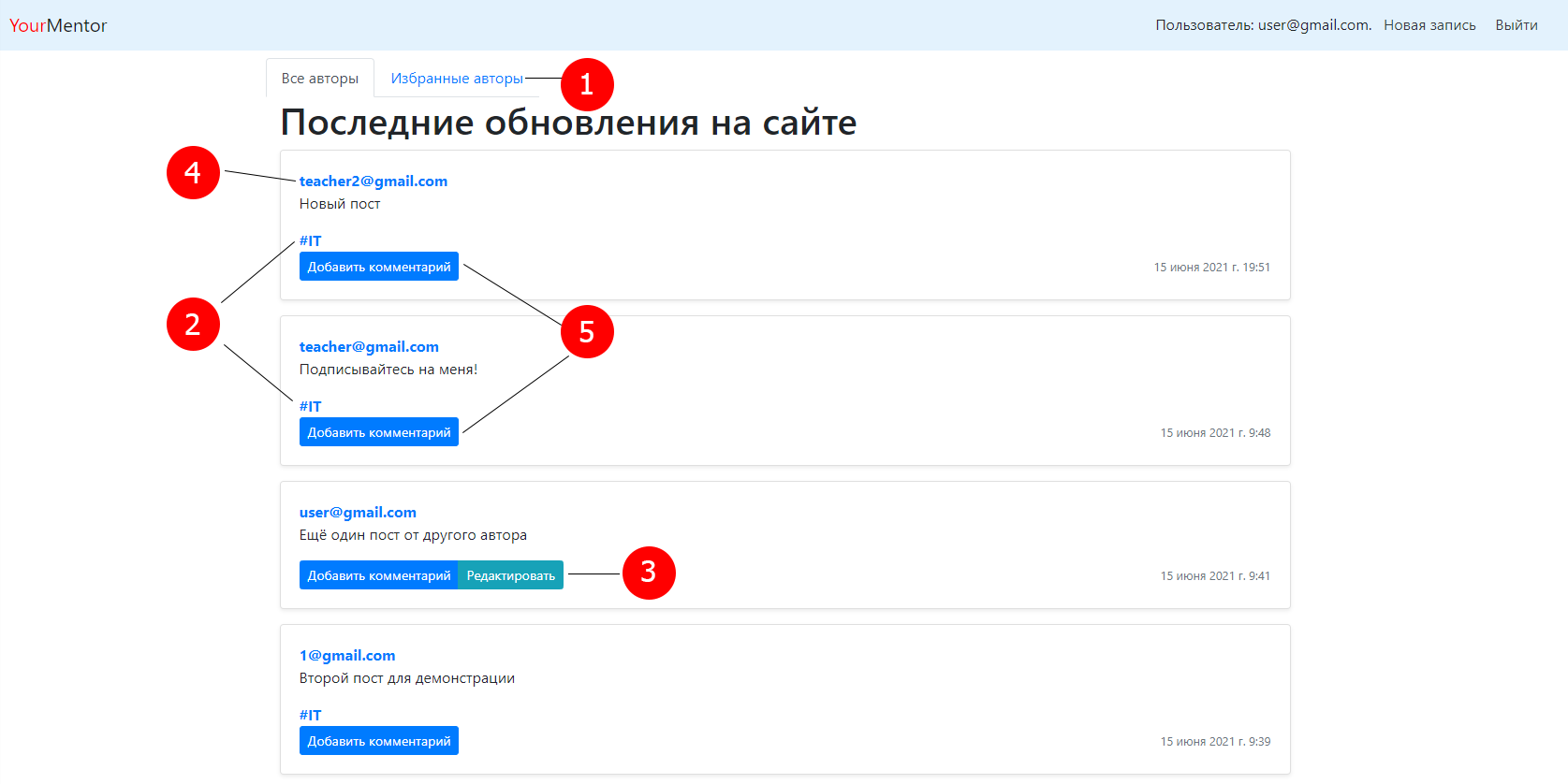
****

Рисунок 4

1. **Код**

Логика основной страницы располагается в файле views.py в приложении posts и называется index, шаблон для данной странице находится в папке templates и называется index.html. В качестве контекста, функция возвращает paginator (объект, позволяющий переключаться между страницами, если много постов) и страницу, которую отрисовывает в данный момент. Paginator представлен на рисунке 5 и появляется, когда кол- во посты превышает 10 штук.

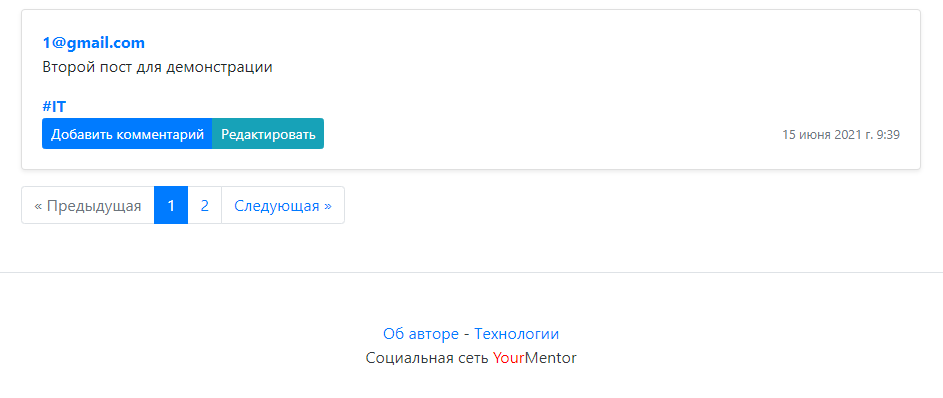


Рисунок 5

* **Создание поста**

1. **Основной функционал и назначение**

Основной целью поста является реклама своих знаний, результатов учеников и личных достижений.

Публикация поста разрешена всем пользователям, прошедших авторизацию. Для учеников посты – это способ показать свои успехи и обозначить цели на ближайшее будущее.

1. **Основные сценарии использования и алгоритм взаимодействия**

Проверка данных на правильность заполнения

Заполнение полей формы

Переход на главную страницу и публикация поста

Логика реализована следующим образом:

* Если пользователь не заполнил обязательные поля, то пост не будет опубликован, а пользователь получит уведомление. (Рисунок 6)

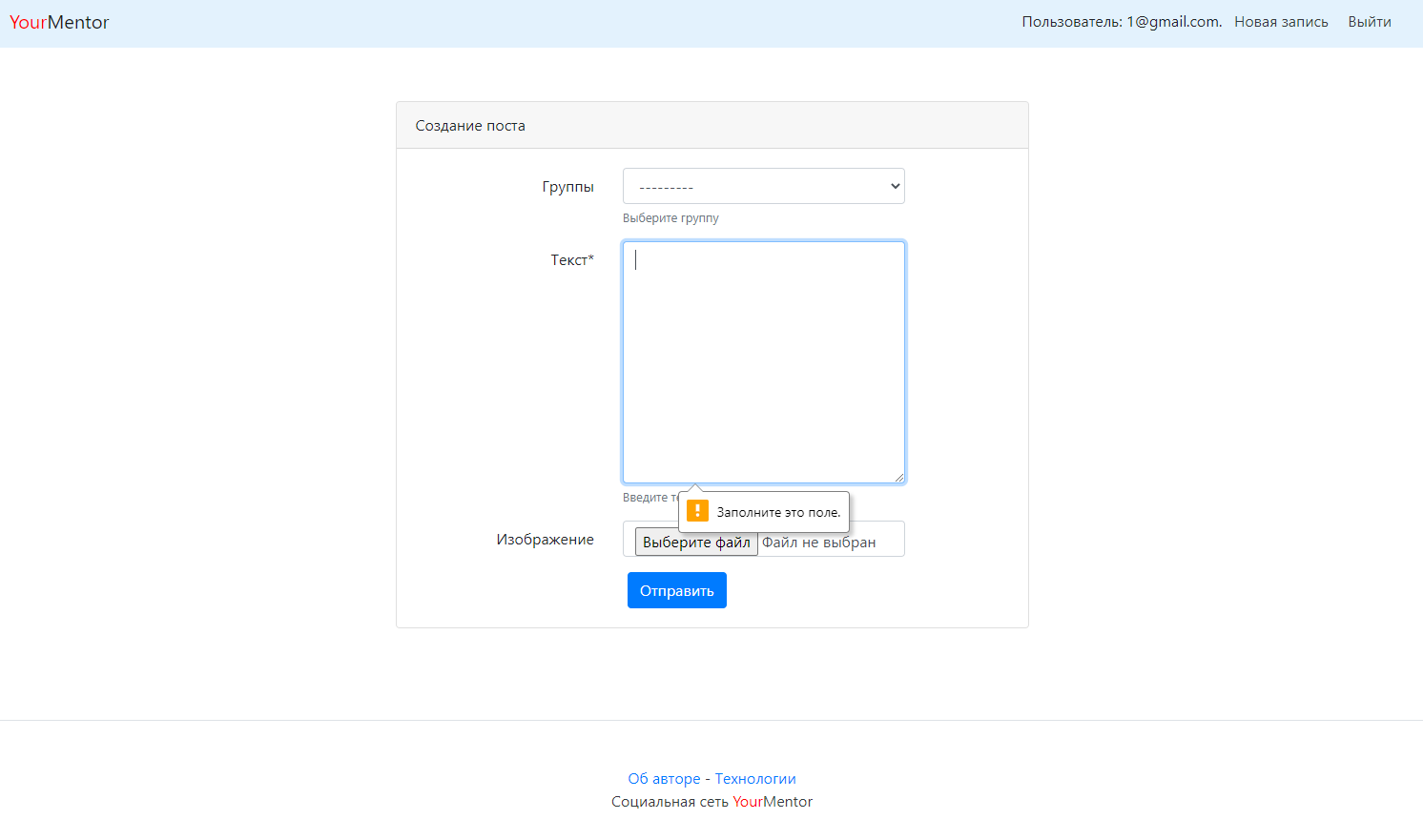


Рисунок 6

1. **Код**

Код располагается в файле views.py в приложении posts и находится в функции new\_post. Для реализации данной функции так же присутствует декоратор @login\_required, позволяющий перейти на данную страницу только авторизированным пользователям.

Front-end часть располагается в папке templates в файле new.html.

* **Просмотр избранных авторов**

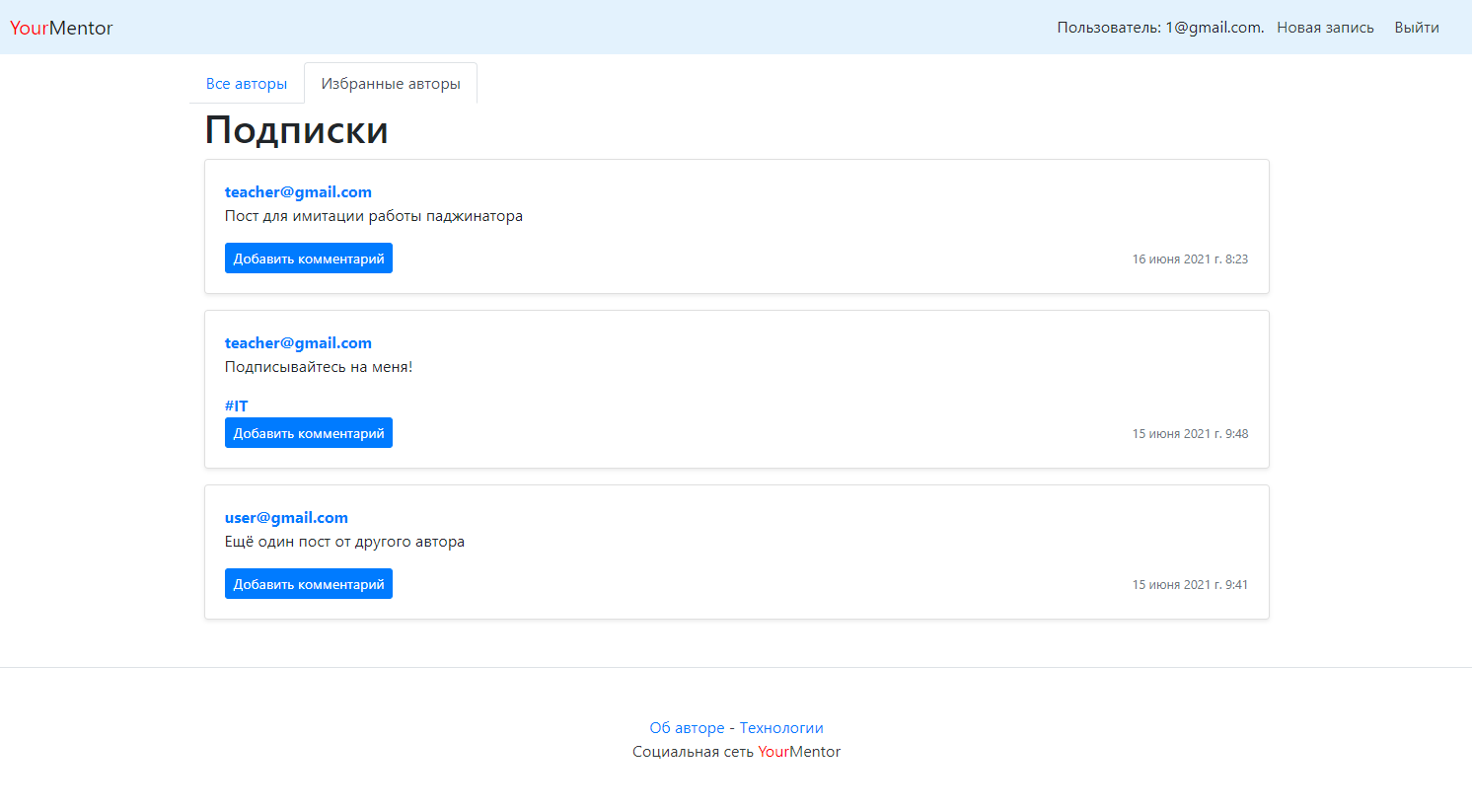
1. **Основной функционал и назначение**

Данная страница отображает посты авторов, на которых вы подписаны, при отмене подписки, посты соответствующего автора будут скрыты.

Данная страница, как и “основная” (Рисунок 4), позволяет добавлять и просматривать комментарии, переходить в группы по хештегу и переходить в профиль автора по клику на соответствующий e-mail.

Сценарии использования также полностью совпадают с “основной” страницей (Рисунок 4).

1. **Дизайн**



1. **Код**

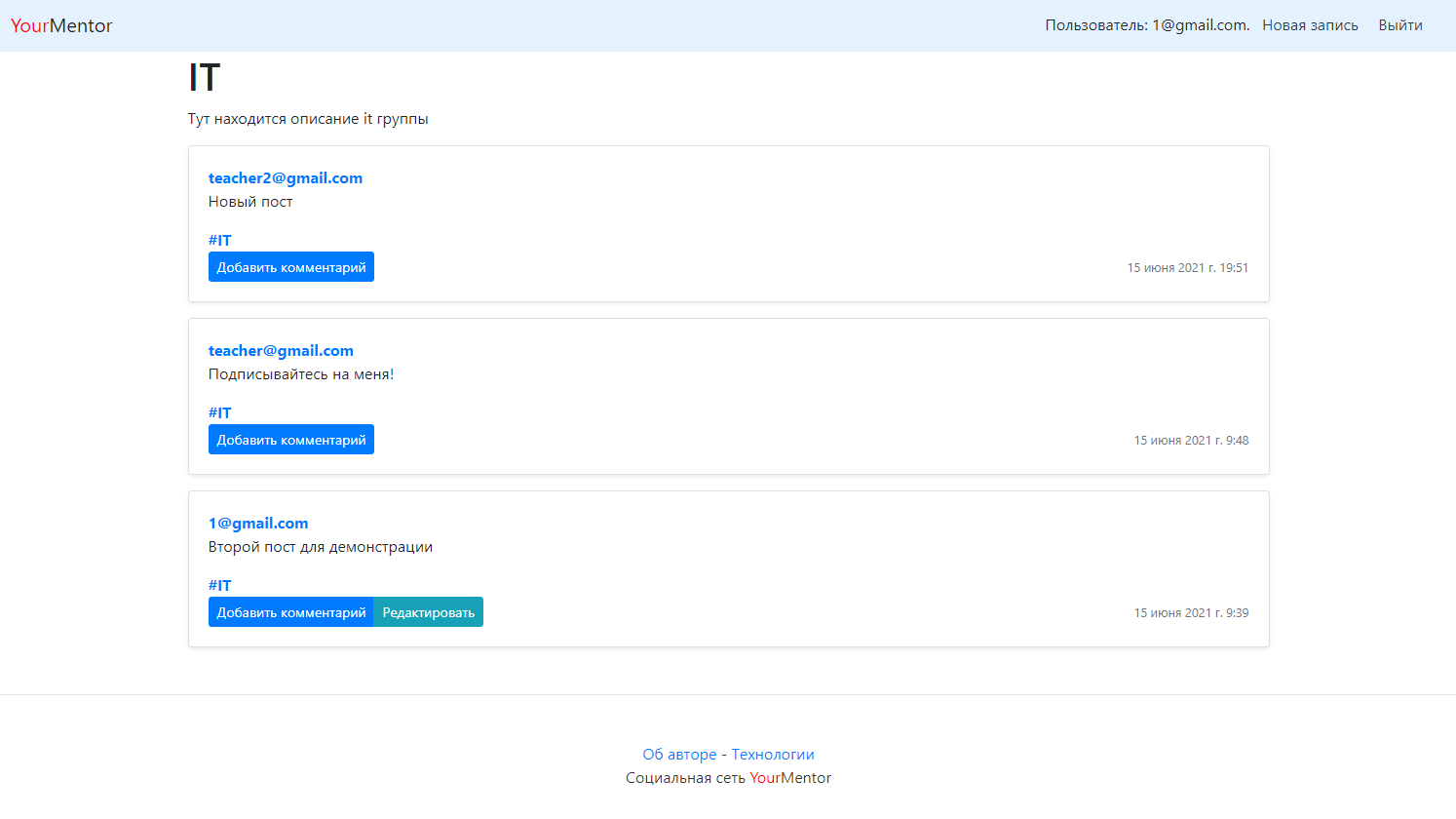
Код располагается в файле views.py в приложении posts и находится в функции follow\_index. Front-end часть располагается в папке templates в файле follow.html. Для данной страницы, так же реализован paginator.

* **Группы**

1. **Основной функционал и назначение**

Данная страница отображает всё посты, имеющие группу при создании (они имеют хештег – Рисунок 4, пункт 2). В группе собраны посты, непосредственно относящиеся только к тебе группы, чтобы пользователи имели возможность читать посты на заданные темы. Добавление групп производит администратор, через admin-панель.

1. **Дизайн**



1. **Код**

Код располагается в файле views.py в приложении posts и находится в функции group\_posts. Front-end часть располагается в папке templates в файле group.html.

* **Бронирование занятия**

1. **Основной функционал и назначение**

Данная страница отвечает за бронирование занятия. Бронирование происходит в 3 этапа

* Выбор дисциплины, по которой хочет заниматься ученик (Рисунок 7)
* Выбор времени и цены занятия (Рисунок 8)
* Выбор времени (Рисунок 9)

Логика данной страницы основана на AJAX запросах и не требует обновления страницы при переходе от одного пункта к другому.

1. **Дизайн**

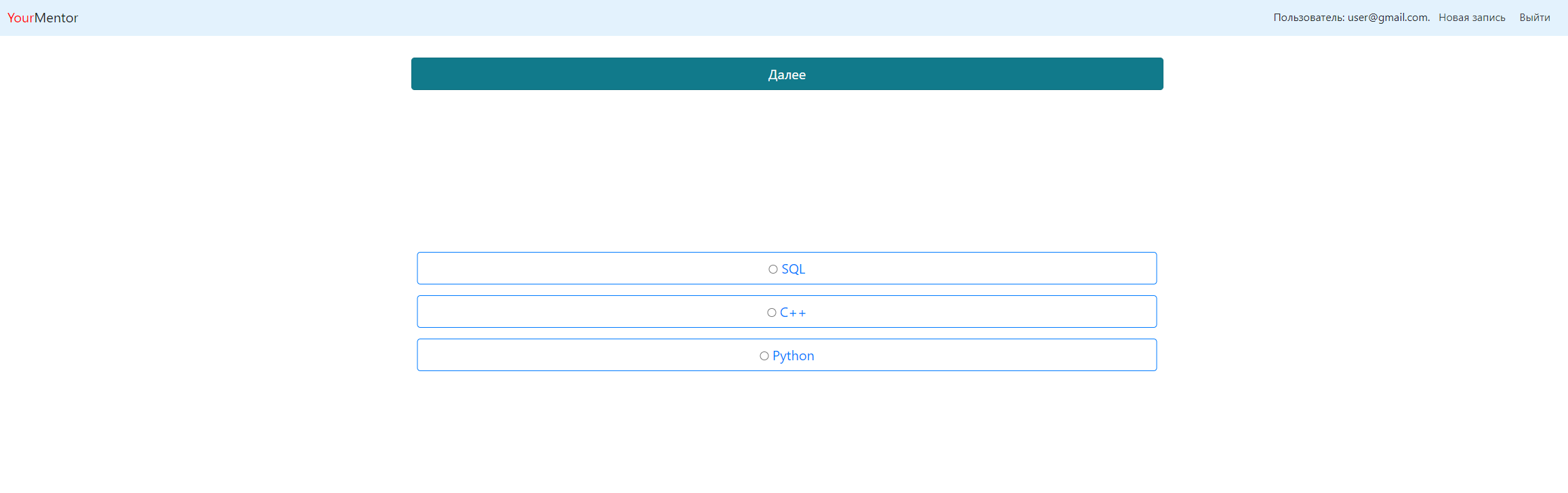


Рисунок 7

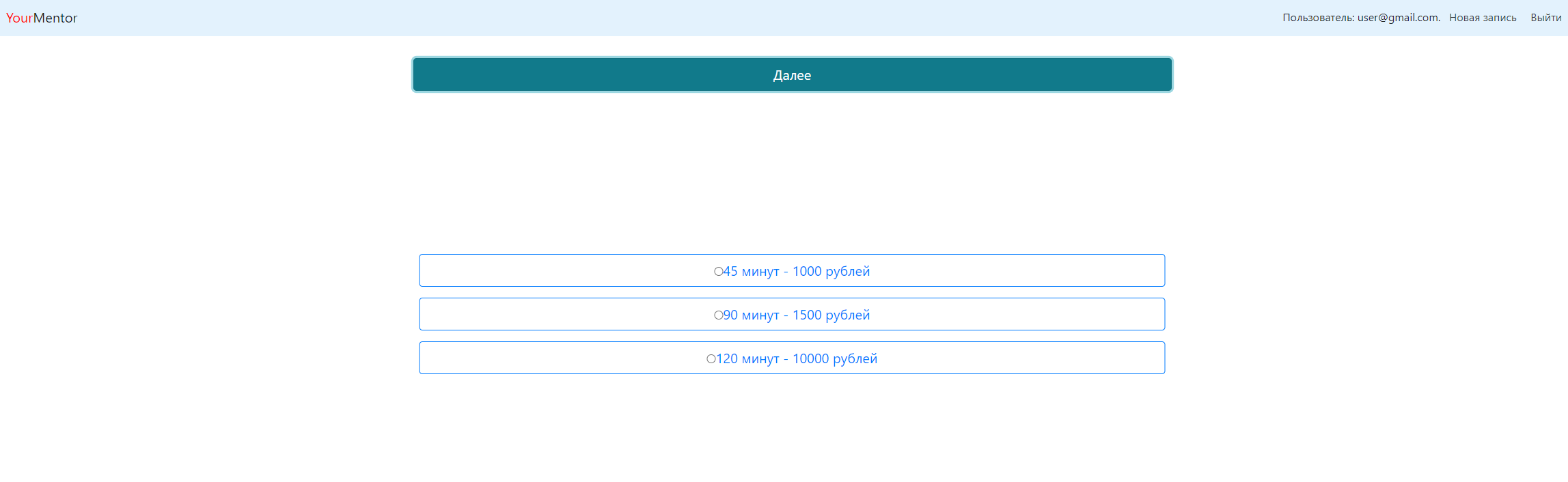


Рисунок 8

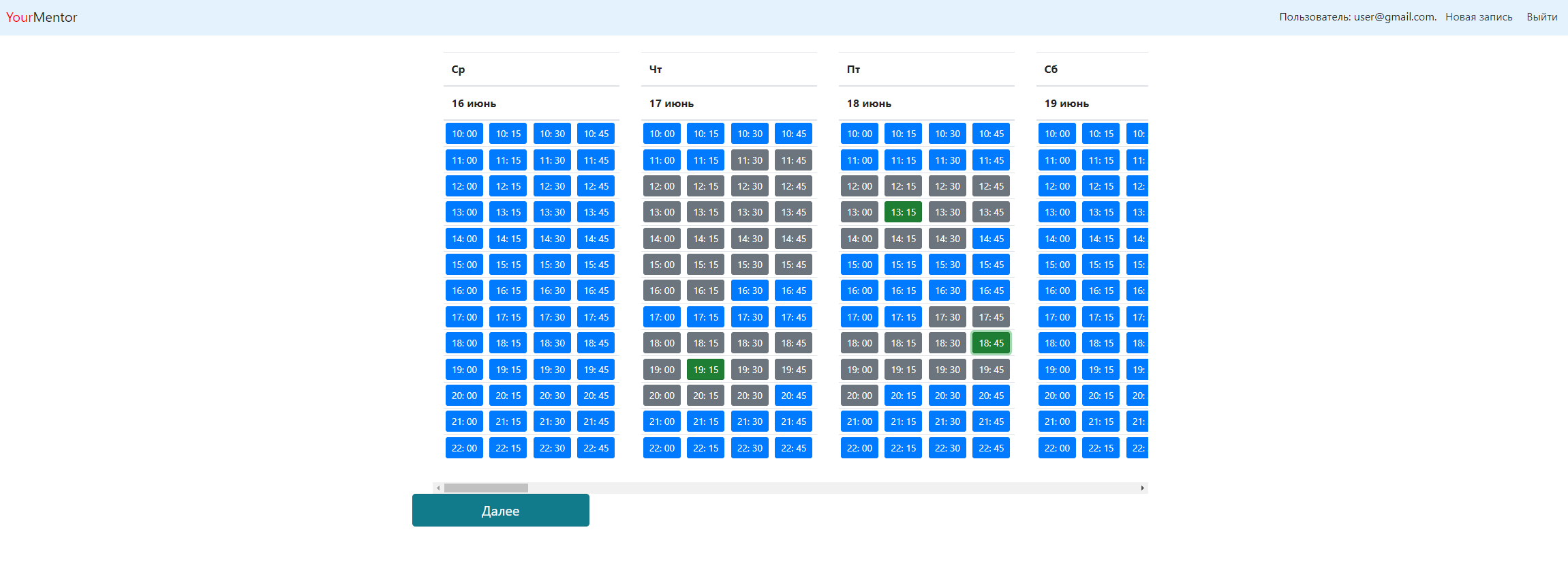


Рисунок 9

1. **Код**

Рассмотри общую схему взаимодействия кода на Python(Back-end) и Java Script (Front-end) (Рисунок 10). Пояснения к рисунку 10:

* После слова Python указано название функции в файле views.py
* После слова JS (Java Script) указано название функции в фале selected\_discipline.html.
* После слова Url указан относительный адрес, по которому производится POST или GET запрос.

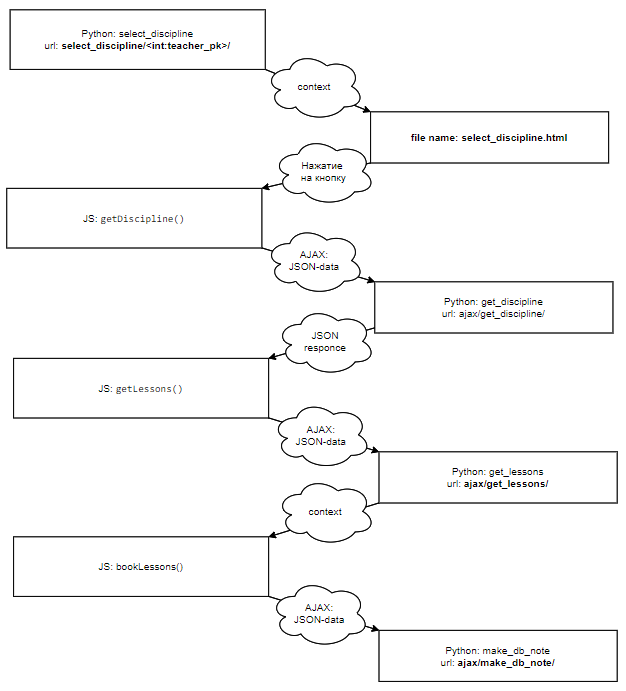


Рисунок 10

Рассмотрим каждую функцию в отдельности.

* Функция select\_discipline (Python)

Отвечает за получение всех дисциплин, которые есть у преподавателя. Если же пользователь, обращающийся к странице не имеет статус ученика или другой пользователь не является преподавателем, то вернётся 404 ошибка. Функция возвращает страницу, полученную с помощью Django-шаблонизатора.

* Функция getDiscipline (Java Script)

Отвечает за выбор нажатия на одну из кнопок. Если не была выбрана дисциплина и пользователь захочет перейти к следующему этапу бронирования, появится alert-диалог (Рисунок 11), содержащий предупреждение и переход к следующему шагу не будет осуществлён. При выборе дисциплины, результатом работы функции - переход по url “ajax/get\_discipline”, который в свою очередь обратится к get\_discipline в Python.

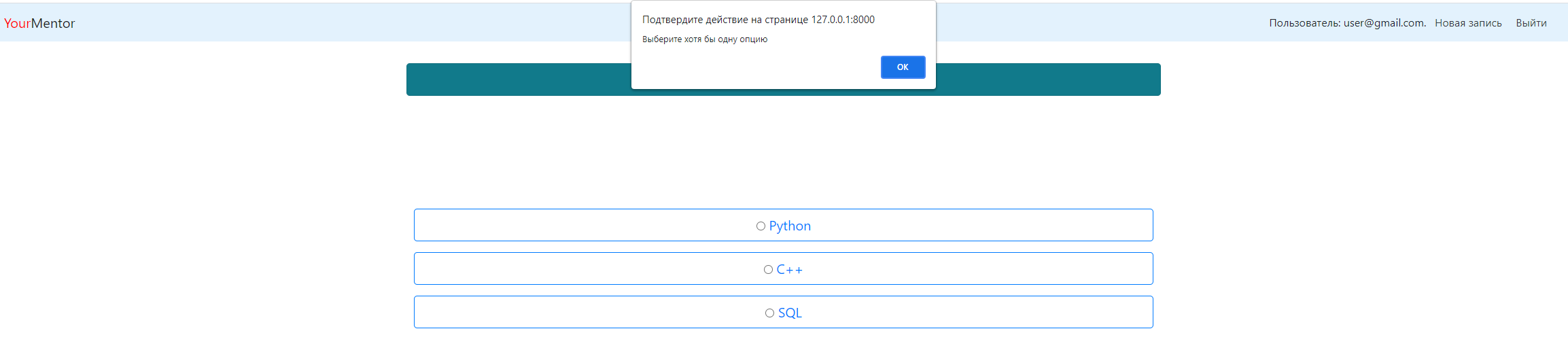


Рисунок 11

* Функция get\_discipline (Python)

Генерирует новый html код для отображения на страницы, через f-строки. Я решил использовать данный метод вместо передачи по шаблону и вынесению html кода в отдельный файл, потому что размер генерируемого шаблона достаточно мал и не занимает много места в коде. Результатом работы функции является сгенерированный шаблон, переданный через JSON обратно в Java Script.

* Функция getLessons (Java Script)

Так как страница является динамической, то при получении информации о выбранной дисциплине, функция скрывает данный контент, и, определив выбранную цену и время занятия выполняет AJAX- запрос на страницу “ajax/get\_lessons/”.

* Функция get\_lessons(Python)

Специализируется на обработке ранее полученных данных (так же передаются и дополнительные данные – такие как дни недели, даты и месяца ближайших 30 дней) и передачи из в Django-шаблонизатор. Результатом работы, является возвращение нового html кода в Java Script.

* Функция bookLessons (Java Script)

Обрабатывает нажатия на выбранные дни для бронирования занятий и отправляет полученные данные, с помощью AJAX-запроса на страницу “ajax/make\_db\_note”, которая в свою очередь переходит в функцию make\_db\_note в Python.

* Функция make\_db\_note (Python)

Отвечает за бронирование выбранных занятий по средством занесения соответствующих данных в базу. После успешного добавления данных, пользователь оказывается на главной странице.

* **Расписание занятий**

1. **Основной функционал и назначение**

Страница нужна для просмотра предстоящих занятий, она будет полезна как для преподавателей, так и студентов. На странице можно смотреть предстоящие занятия и в случае надобности отменять их.

1. **Дизайн**

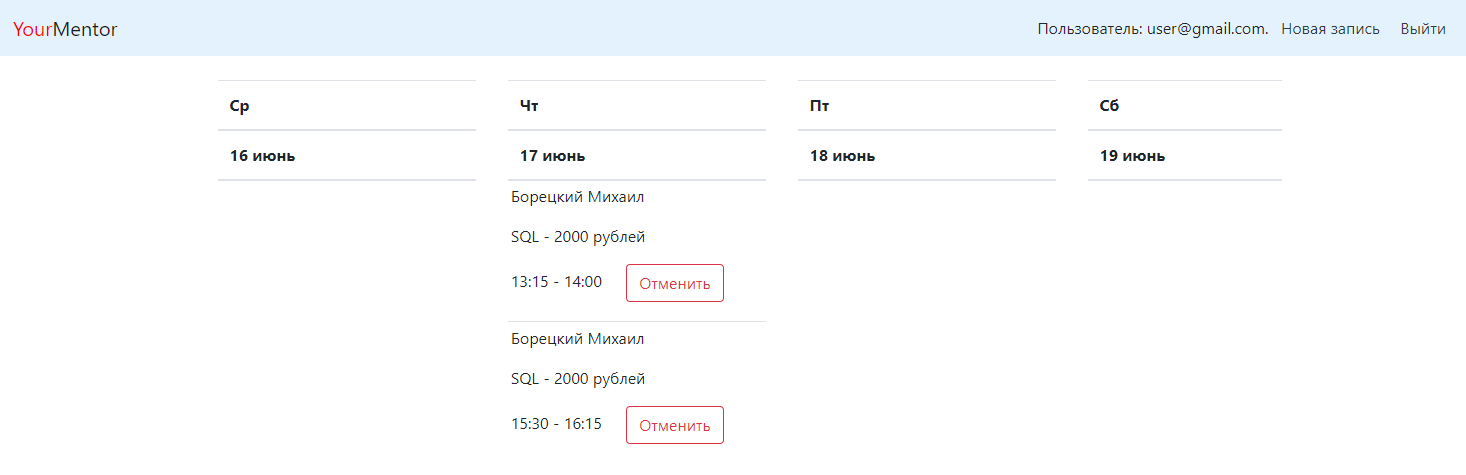


Рисунок 12

1. **Код**

Прежде чем перейти к коду, рассмотрим общую схему работы данной страницы.

Схема запросов представлена на рисунке 13:

Context

url: timtime\_table/

Python: get\_timetable (view)

AJAX: JSON-data

JS: cancel button click

url: ajax/cancel\_lesson/

Python: cancel\_lesson (view)

Рисунок 13

Рассмотрим функцию get\_timetable (Рисунок 14):

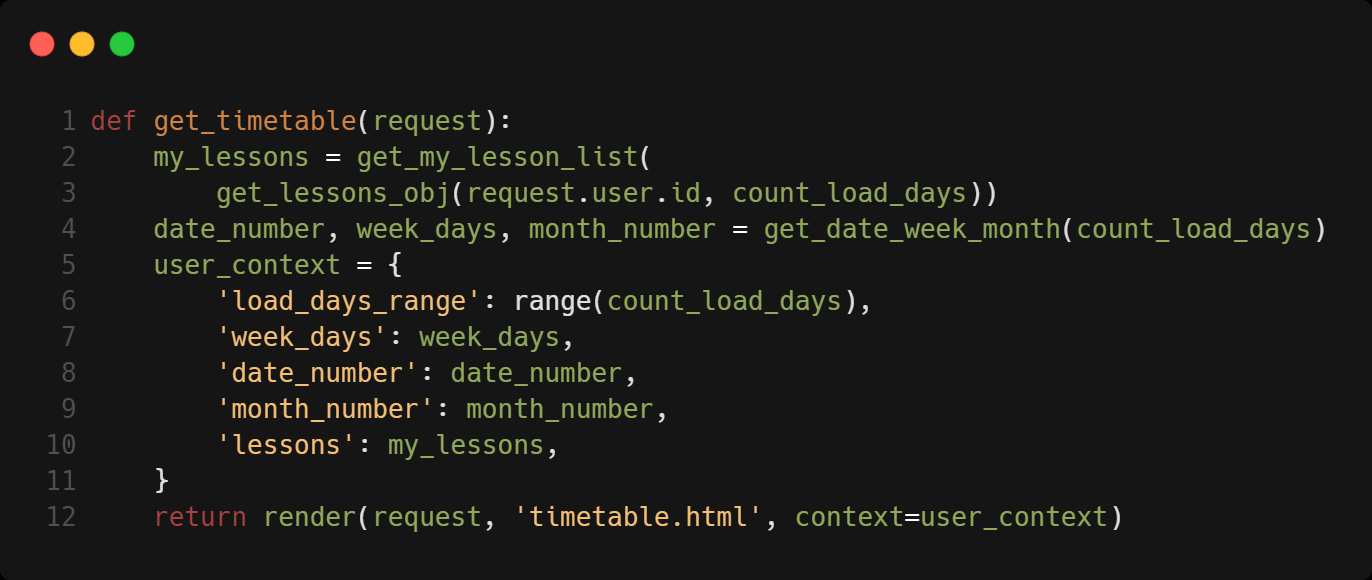


Рисунок 14

Функция передаёт в шаблон следующие данные: кол-во загружаемых дней, день недели, месяц, дату и все не отменённые уроки, где пользователем является учителем или учеником, таким образом, просмотр расписания удобен для всех типов пользователей и не требует отдельных проверок.

Теперь рассмотрим код в Java Script (Рисунок 15):



Рисунок 15

Для каждой кнопки отменить я использую data-id, чтобы потом в коде (Рисунок 16) определить кнопку, которая была нажата.

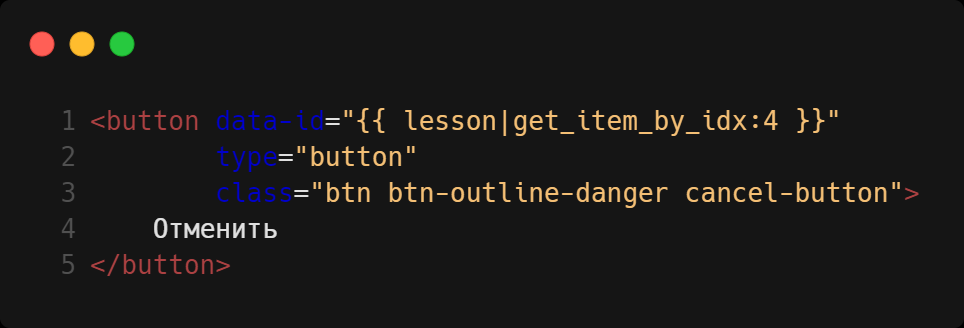


Рисунок 16

Так же для присвоения data-id я использую свой фильтр, который позволяет получать значение из массива по индексу. В 4 индексе двумерного массива lesson содержится id данного урока в базе данных и таким образом получается следующая схема:

1. Присваиваю каждой кнопке id урока, на который она ссылается
2. При обработке нажатия, получаю данный id и отправляю AJAX запрос с этими данными
3. Принимаю запрос в view функции и меняю у урока, по полученному id поле is\_cancaled и устанавливаю ему значение True. (И более данный урок не будет отображаться в расписании)
4. Удаляю элемент из DOM дерева
5. **Заключение**

**Результат**

При работе над данным проектом я приобрёл следующие навыки:

* Приобрёл навыки работы с Django-ORM и шаблонизатором Django
* Познакомился с понятием API и реализовал на практике полученные знания
* Реализовал динамические страницы по средствам AJAX-запросов
* Научился стерилизовать данные

**Планы на будущее**

На данный момент, для данного проекта существует множество путей развития и в первую очередь – изменение дизайна. Bootstrap предоставляет хорошую адаптивную вёрстку, но стилей не хватает. На данный момент существует новый вариант дизайна (Рисунок 17) для страницы бронирования занятий. Так же требует доработки тестирование функционала по средствам автоматических тестов и проработка API приложений.

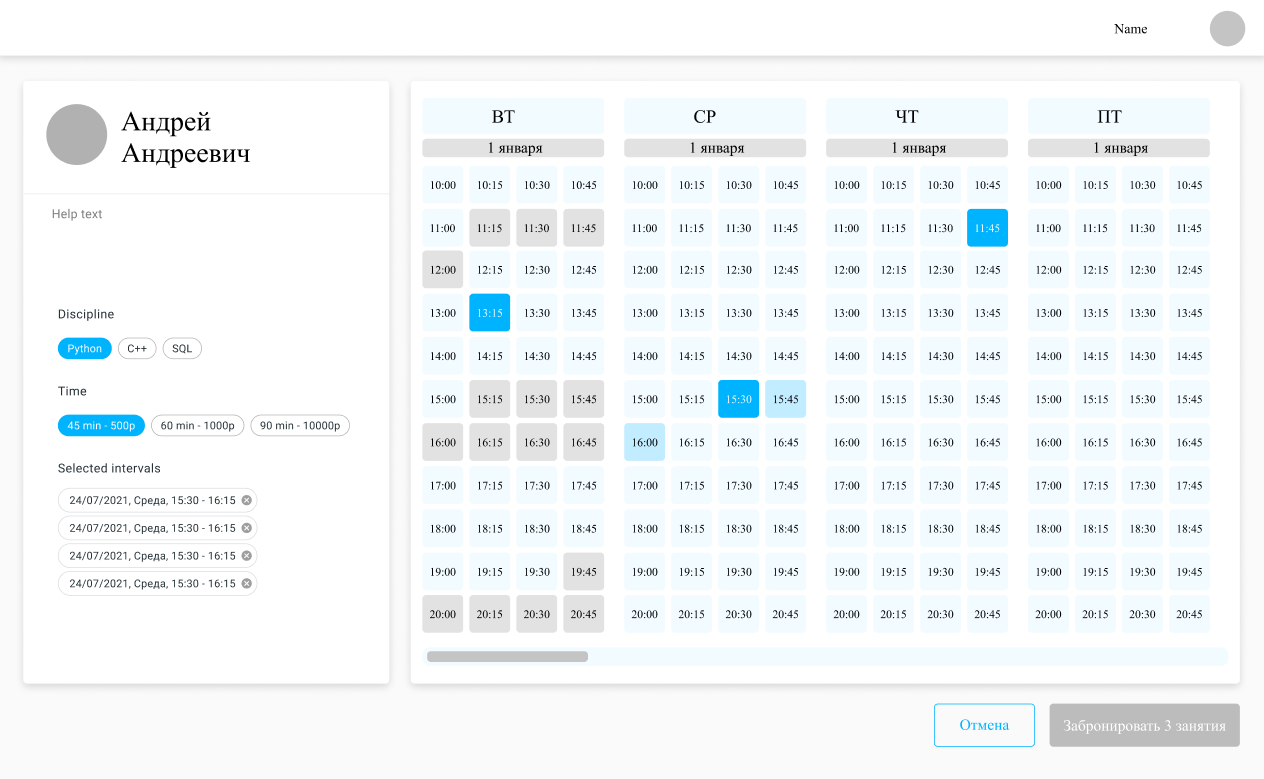


Рисунок 17

1. **Литература**
2. <https://praktikum.yandex.ru/profile/backend-developer/>
3. <https://docs.djangoproject.com/en/3.2/>
4. <https://ilyachch.gitbook.io/django-rest-framework-russian-documentation/overview/readme>
5. <https://www.django-rest-framework.org/topics/documenting-your-api/>
6. <https://habr.com/ru/company/yandex_praktikum/blog/561696/>
7. **Github - репозиторий**

<https://github.com/skprpi/booking_service>