**Департамент образования и науки города Москвы**

**Государственное бюджетное общеобразовательное учреждение**

**Г. Москвы**

**«Школа №1501»**

**«Сайт для создания и решения школьных тестов»**

Автор: Лобков Антон Юрьевич

Ученик 10 класса «Б»

ГБОУ города Москвы «Школа №1501»

Научный Руководитель: Анатольев Алексей Владимирович, учитель информатики

**г. Москва**

**2022-2023 учебный год**

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ……………………………………………………………………..3

СТРУКТУРА САЙТА. ДИЗАЙН……………………………………………...5

СИСТЕМА АККАУНТОВ. АВТОРИЗАЦИЯ И ВЫХОД С УСТРОЙСТВ...6

КОНСТРУКТОР ТЕСТОВ……………………………………………………..8

ПРОХОЖДЕНИЕ ТЕСТА. РЕЗУЛЬТАТЫ………………………………….11

СТРУКТУРА ПРОЕКТА. ИНСТРУМЕНТЫ РАЗРАБОТКИ………………13

**ВВЕДЕНИЕ**

В начале я хочу пояснить некоторые термины, которые будут использоваться дальше:

* **Frontend** *(рус. фронтенд) –* то, что видит пользователь, сам сайт, анимации, стили и т. д.
* **Backend** *(рус. бэкенд) –* то, что не видит пользователь, что находится «под капотом» сайта, приложения: взаимодействия с базой данных и т. д.
* **API –** описание способов взаимодействия одной компьютерной программы с другими, в моем случае это взаимодействие при помощи веб-запросов
* **Фреймворк** *(от англ. frame – каркас) –* большая программная библиотека, как правило включающая в себя множество других библиотек, облегчающая разработку и направленная на определенную задачу. Например, для веб-разработки, работы с данными и т. д.

Когда я начала думать над темой для проекта, я сразу понял, что хочу делать сайт. К идее сайта-конструктора тестов я быстро пришел так как уже делал программу на Python для создания тестов. Свой сайт я хотел сделать не только чтобы он просто работал, но и чтобы был красивым, как минимум удобным в использовании.

Как правило сайты состоят из бекэнда и фронтенда. В качестве фреймворка для сервера, я выбрал Flask, так как его я хорошо знал, а учить новый заняло бы много времени. Flask – это Python фреймворк для разработки веб приложений. Создатели позиционируют его как фреймворк для быстрого старта: минимум настроек, почти сразу можно перейти к написанию основной части проекта. Поэтому этот фреймворк часто используют для небольших проектов. В него уже включены библиотеки для работы с базой данных, использования шаблонов страницы и другие вспомогательные.

Изначально весь сайт был написан только на Flask, вся обработка данных и отрисовка сайта была фактически в одном файле, это очень неудобно. Поэтому я прошел курс frontend-разработки и узнал одну из самых популярных и удобных библиотек для этих целей – React. Она позволяет конструировать сайт из блоков (компонентов), и удобно управлять каждым из них. Frontend является отдельным от бекэнда проектом, в моем случае он написан на другом языке – JavaScript. В ИТ-компаниях фронтендом и бэкендом занимаются разные люди. К счастью, к этому моменту я находился на начальном этапе разработке сайта, поэтому переход проекта на новый способ разработки веб-приложения не вызвало проблем, я быстро перешел и продолжил разработку. Все нижеописанные части проекта идут в хронологической последовательности.

**СТРУКТУРА САЙТА. ДИЗАЙН**

Сайт разрабатывался так, что верхняя навигационная панель и подвал со ссылками на социальные сети есть на всех страницах. В навигационной панели содержатся ссылки на главную страницу сайта, конструктор тестов, а также кнопки войти и регистрация, если пользователь не вошел в аккаунт (рис. 1), и ссылка на профиль, если пользователь вошел в аккаунт (рис. 2).

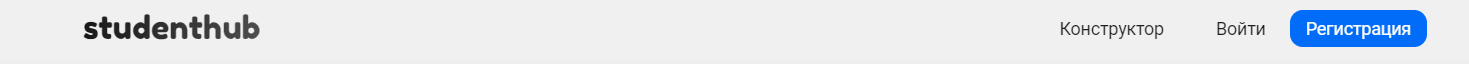


Рисунок 1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2

Сайт имеет адаптацию под телефон, на телефонах все ссылки из навигационной панели сворачиваются в «меню бургер» (рис. 3).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3

Навигационная панель прозрачная, задний фон размывается. В подвале находятся ссылки на мои социальные сети и ссылка на репозиторий GitHub (исходный код проекта). При разработке дизайна я придерживался минимализма, следовал руководству «человеческого интерфейса» от Apple.

**СИСТЕМА АККАУНТОВ. АВТОРИЗАЦИЯ И ВЫХОД С УСТРОЙСТВ**

Разработку своего сайта я начал с системы аккаунтов, авторизации и регистрации. Первым делом настроил работу базы данных, и создал модель (таблицу) с пользователями. После начал делать систему авторизации, я изучил различные методы и выбрал авторизацию с помощью токенов. Работает это работает следующим образом: клиент отправляет запрос на сервер с логином и паролем, сервер проверяет существует ли такой пользователь и правильный пароль, если все верно, то сервер создает токен (уникальный ключ), который действителен 30 дней, после нужно входить заново. Затем сервер отправляет токен обратно клиенту, токен сохраняется в браузере и используется для последующего входа.

При регистрации пользователь может выбрать тип аккаунта ученик или учитель, учителя могут создавать тесты, ученики – нет.

Так как подразумевается, что сайт будет использоваться на школьных компьютерах, то есть вероятность что ученик может забыть выйти из аккаунта на компьютере. Поэтому я сразу решил, что сделаю возможность удаленного выхода с других устройств. Подобное практикуется во многих сервисах, где хранится конфиденциальная информация пользователя, например Google и Telegram (рис. 4 и рис. 5)

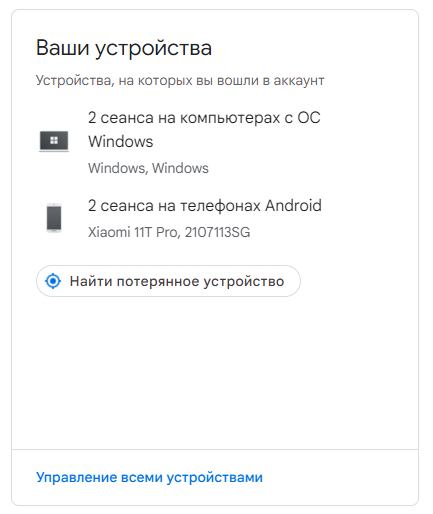


Рисунок 4

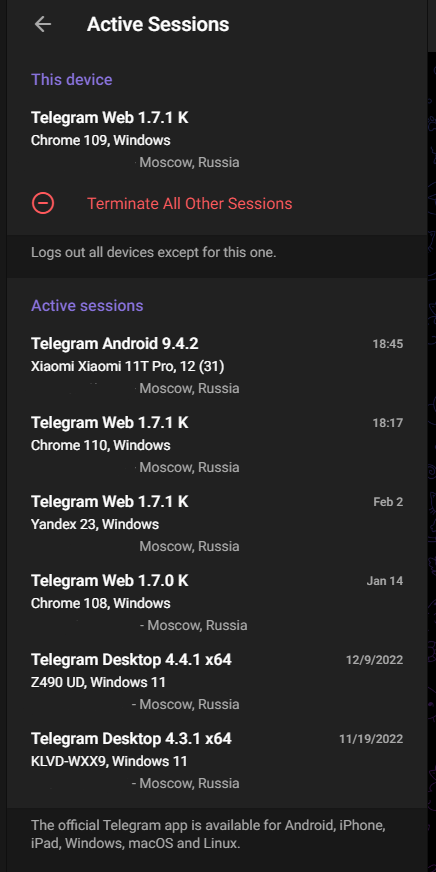


Рисунок 5

На моем сайте не хранится конфиденциальная информация, но мне было интересно сделать такую возможность и считаю ее полезной.

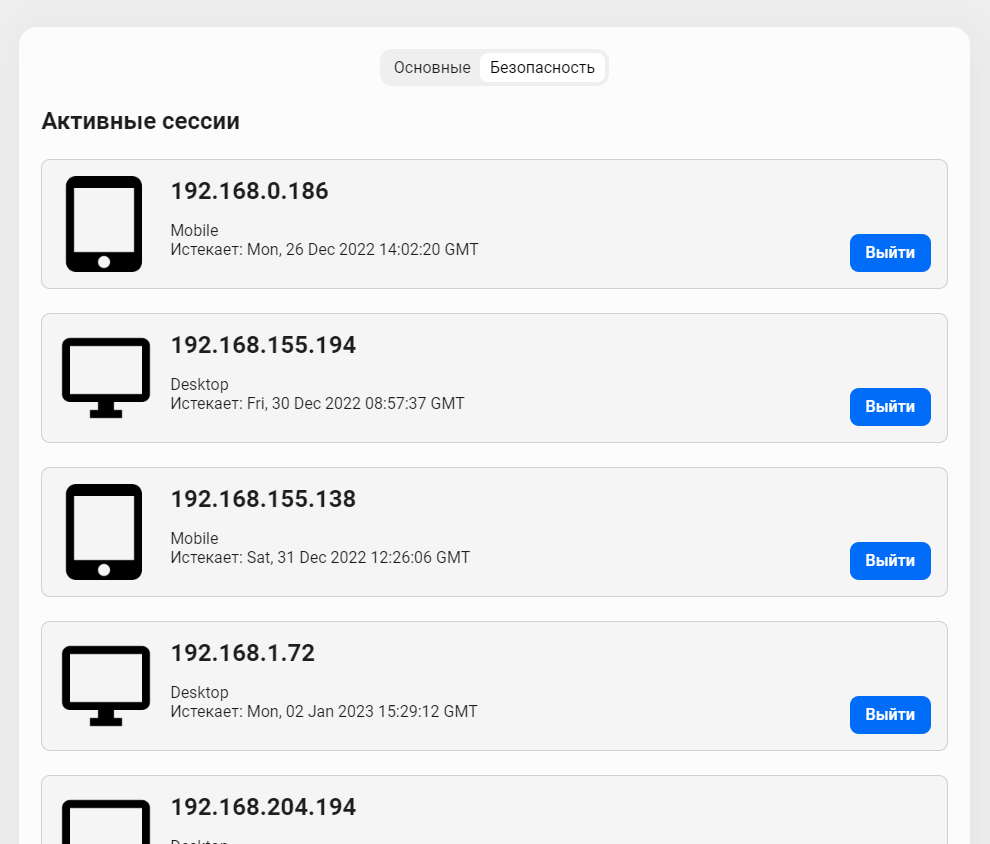


Рисунок 6

**КОНСТРУКТОР ТЕСТОВ**

Конструктор тестов – основная и самая сложная часть проекта. Конструктор занял у меня больше всего времени, в принципе я так и планировал. В самом начале проекта я решил, что отличительной особенностью будет являться то, что в одном задании может быть несколько вопросов, такое я видел только на сайте олимпиады <https://online.olimpiada.ru>. Это может быть полезно в заданиях на вставку слов в текст, а также если есть несколько вопросов к одному условию задачи. Баллы начисляются за каждый вопрос отдельно.

Конструктор состоит из поля ввода условия задачи, добавления вопроса, предпросмотр задания редактирования вопроса и кнопок сохранения и удаления задания (рис. 7). При нажатии на добавление задания предлагается на выбор 4 типа вопросов: ввод, выбор, перестановка и установление соответствий (связка) (рис. 7).

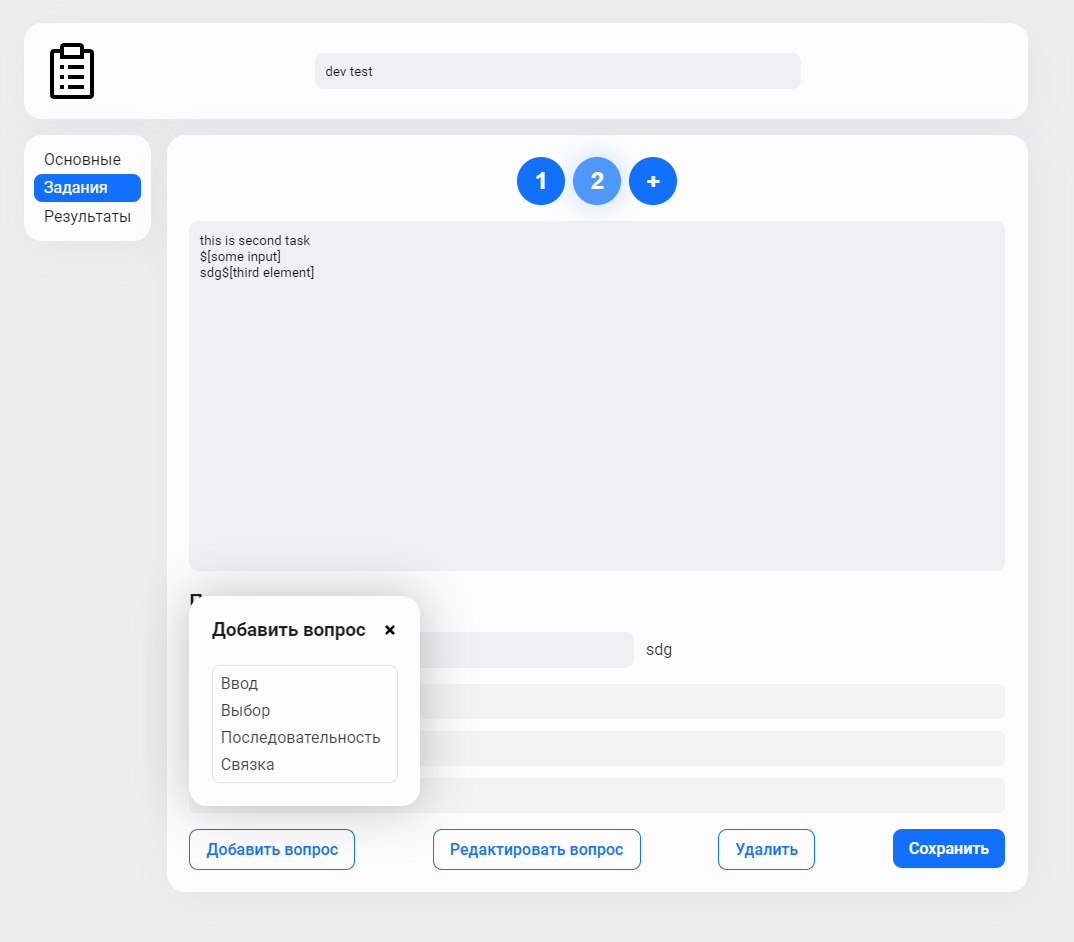


Рисунок 7

Для каждого типа вопросов свои параметры в редакторе вопросов (рис. 8). Для ввода это переключение форматирования число или строка и внутристрочный ввод (вставка слова). Для выбора это добавление элементов и возможность выбрать несколько вариантов ответа. Для перестановки – добавление элементов. Для связки – добавление вариантов выбора и групп куда можно перетащить эти варианты.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8

Два общих параметра — это название вопроса и правильный ответ. Название вопроса носит исключительно технический характер, оно позволяет системе понять куда вставлять задание, у ученика название нигде не отображается.

Вопросы вставляются в текст задания в виде «$[<название вопроса>]». Такие метки можно перемещать по тексту, если ее удалить, то после сохранения вопрос полностью удалится, хотя до сохранения его еще можно будет видеть в списке вопросов.

**ПРОХОЖДЕНИЕ ТЕСТА. РЕЗУЛЬТАТЫ**

Страница теста содержит название и описание теста, если тест уже был пройден, то отображаются лучшее, последнее и все остальные решения (рис. 9). Если у пользователя есть любой незавершенный тест, то он не сможет начать новый, высветиться уведомление, что надо пройти предыдущий.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9

На странице с решением теста помимо заданий есть индикатор решенных и пропущенных заданий, решенные помечаются синим, пропущенные серым, к пропущенным заданиям можно вернуться позже. Переключаться между заданиями нельзя. Доступ к чужим решениям получить нельзя, учитель может просматривать решения учеников.

После прохождения теста ученику будут доступны правильные ответы. Правильные ответы подсвечены зеленым, неправильные – красным. Внизу страницы есть переключатель какие ответы отображать: правильные или ответы ученика (рис. 10)

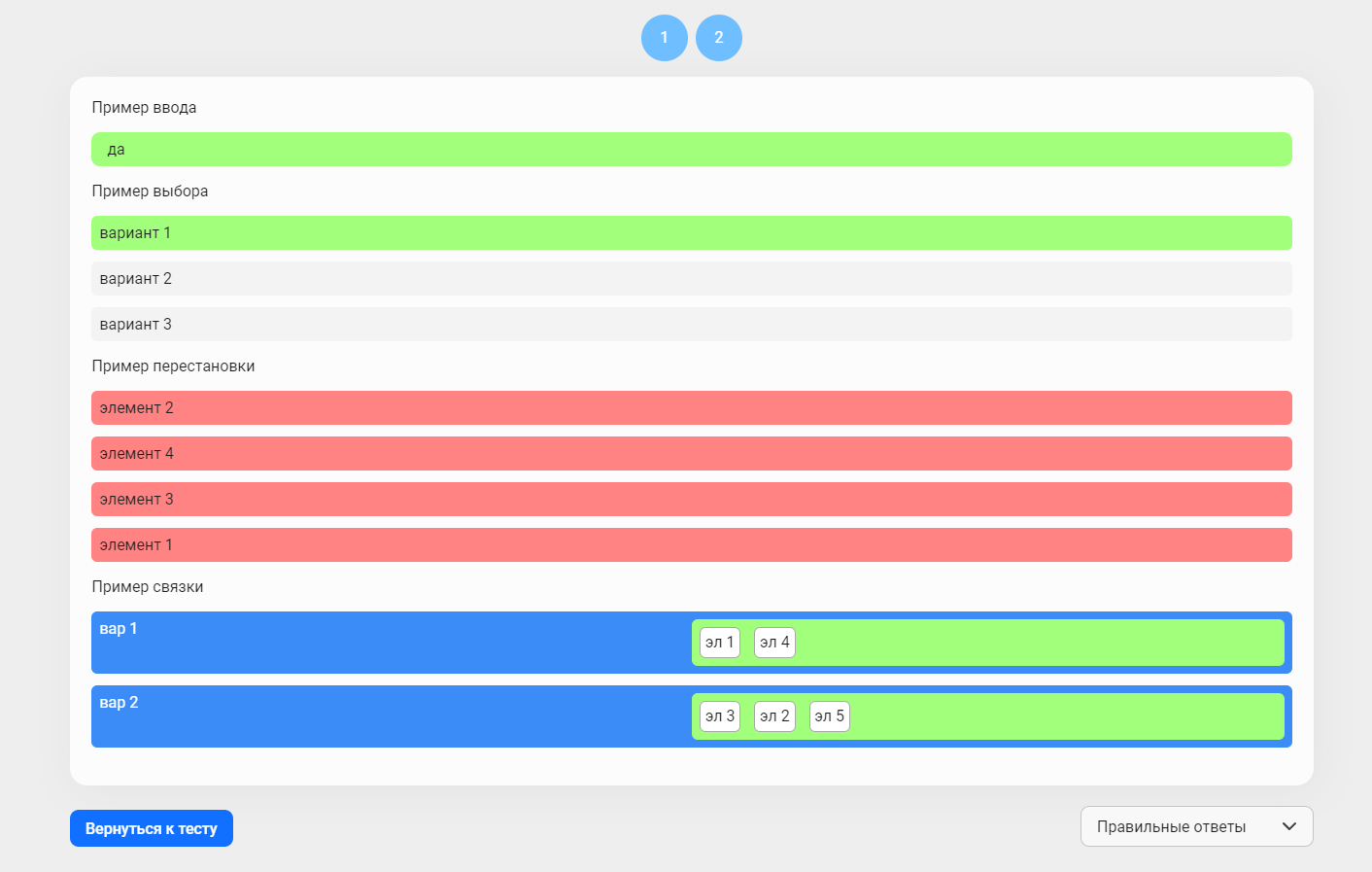


Рисунок 10

У учителя отображаются результаты учеников, по умолчанию показывается только первый результат, при нажатии на ученика показываются все результаты (рис. 11).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11

**СТРУКТУРА ПРОЕКТА. ИНСТРУМЕНТЫ РАЗРАБОТКИ**

Как я уже говорил ранее, мой проект состоит из двух частей: фронтенда и бэкенда. Бэкенд состоит из базы данных, файлов тестов, картинок и кода (рис. 12). Весь код бэкенда написан на Python. Для работы с базами данных я использую SQLite и Python-библиотеку SQLAlchemy. Структуру базы данных я привел на рис. 13.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12

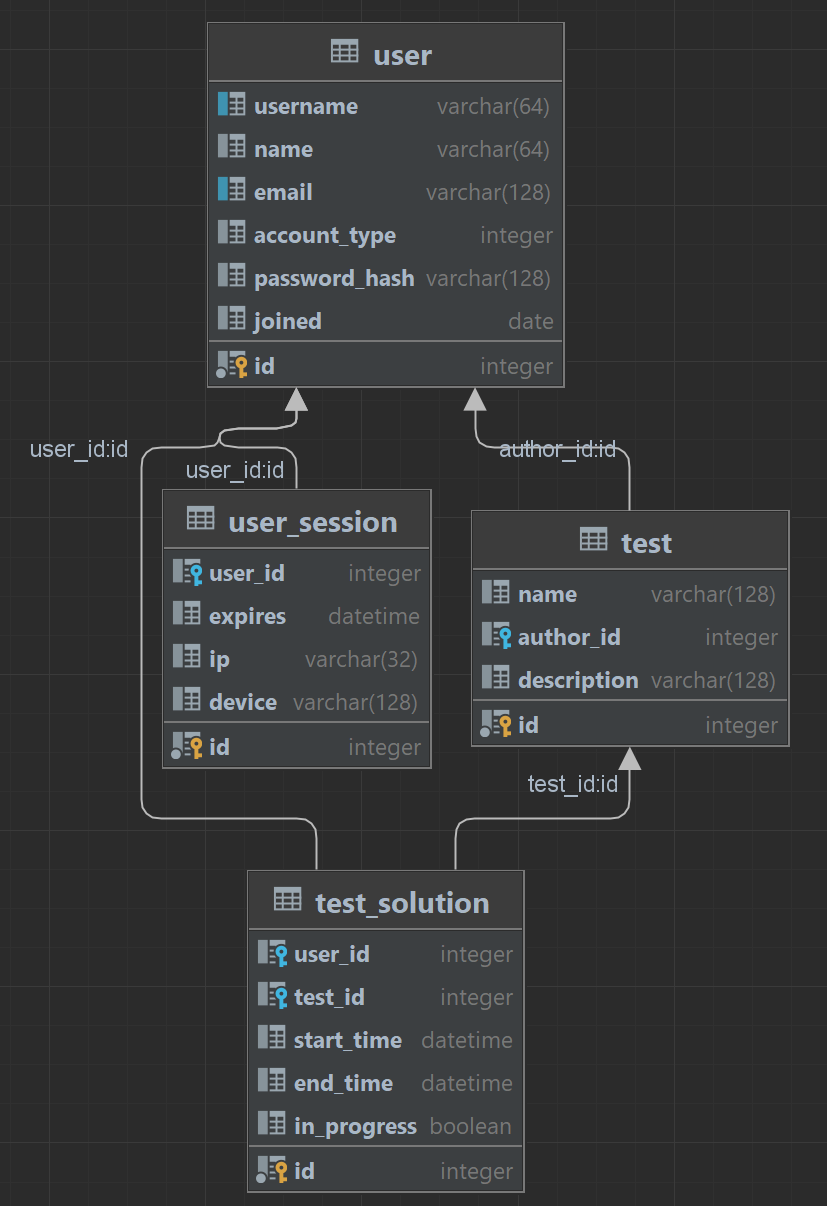


Рисунок 13

В файле routes.py описаны все маршруты для доступа к ресурсам сервера. Основные из них авторизация, получение теста, отправка ответа на задание, редактирование теста. В файле models.py описаны все модели базы данных (рис. 13).

Фронтенд составляющая проекта написана на языке программирования JavaScrip. Фреймворк для отрисовки интерфейсов я использую React. Файловая структура состоит из папки с компонентами и папки с страницами (рис. 14).

В данный момент чтобы зайти на сайт нужно запустить 2 сервера: бэкенд и фронтенд отдельно. Это удобно в разработке, но плохо в финальной версии. Для «упаковки» и оптимизации сайта используется библиотека webpack, у меня она полностью настроена.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14

**ВЫВОДЫ**

Область применения этого проекта четкая и ясна с самого начала, это означает, что он может начать работу сразу после завершения. Мое решений так же может послужить основой для более крупного проекта, например образовательной платформы, сайта с курсами и т.д.

Еще я делал этот проект как замену неудобной и плохо работающей на текущей школьной операционной системе Linux программе MyTestXPro. И считаю этой цели я достиг. Тестовая система в формате сайта означает, что работать она будет на любом устройстве одинаково. Более того веб-приложения намного проще корректировать и ими проще пользоваться. На данный момент все задачи, которые я поставил себе в начале проекта я выполнил.

**ИСПОЛЬЗОВАННЫЕ РЕСУРСЫ**

1. React <https://reactjs.org/>
2. Flask <https://flask.palletsprojects.com/en/2.2.x/>
3. Apple Human Interface Guidelines <https://developer.apple.com/design/human-interface-guidelines/guidelines/overview/>
4. MDN Web Docs <https://developer.mozilla.org/ru/>
5. Авторизация <https://habr.com/ru/post/340146/>
6. JavaScript codestyle <https://github.com/airbnb/javascript/tree/master/react>

Исходный код проекта: <https://github.com/x64penguin/studenthub>