

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Казанский (Приволжский) федеральный университет»
Институт вычислительной математики и информационных технологий

Кафедра системного анализа и информационных технологий

Направление подготовки: 02.03.02 — Фундаментальная информатика и
информационные технологии

Профиль: Системный анализ и информационные технологии

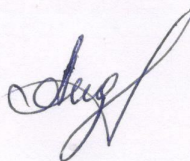
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
WEB-ПРИЛОЖЕНИЕ ДЛЯ ОРГАНИЗАЦИИ ОБУЧЕНИЯ
И ПРОВЕРКИ ЗНАНИЙ ОСНОВ МУЗЫКАЛЬНОЙ ТЕОРИИ

Обучающийся 4 курса
группы 09-031




(Хамитова А.Р.)

Руководитель
канд. физ.-мат. наук, доцент



(Андрианова А.А.)

Заведующий кафедрой системного анализа
и информационных технологий,
канд. физ.-мат. наук, доцент



(Васильев А.В.)

Казань – 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
1. Анализ предметной области	5
2. Функциональные требования к приложению	7
3. Архитектура проекта	9
4. Описание базы данных	11
5. Реализация серверной части приложения.....	19
6. Реализация клиентской части приложения.....	22
7. Реализация нотного редактора	25
8. Демонстрация работы приложения и тестирование	28
9. Анализ полученных результатов	38
ЗАКЛЮЧЕНИЕ	40
СПИСОК ЛИТЕРАТУРЫ.....	44
ПРИЛОЖЕНИЕ	45

ВВЕДЕНИЕ

В настоящее время, как и всегда, музыка является популярным искусством, она звучит в каждом кафе, присутствует во всех фильмах и сериалах, а также ни один праздник не обходится без нее. Однако несмотря на то, что музыка всегда рядом, далеко не все ее понимают, а ведь с пониманием она становится еще прекраснее.

В музыкальных школах обучают музыкальной грамоте, однако, если ученик не смог присутствовать на занятии, не так просто восполнить пропущенный материал, поскольку необходимо много media материалов.

В век популярности информационных технологий и online-сервисов идеальной была бы возможность улучшить свои музыкальные знания дистанционно, однако платформ для самостоятельного обучения не так много. На просторах интернета были найдены курсы по музыкальной теории, однако это конкретные курсы, которые можно лишь купить и пройти, и возможности добавить свой курс для распространения ученикам выявлено не было.

Реализованное web-приложение призвано помочь самостоятельно изучить музыкальную грамоту, а также помочь учителям музыкальных школ ввести интерактивный аспект в процесс обучения.

Целью выпускной квалификационной работы является спроектированное и реализованное web-приложение, которое призвано помочь в организации обучения основам музыкальной теории, а также в проверке усваиваемости материала.

Для достижения поставленной цели были назначены следующие задачи:

- 1) исследование музыкальной предметной области,
- 2) разработка макета,
- 3) проектирование и реализация базы данных,
- 4) проектирование и реализация нотного редактора,
- 5) проектирование и реализация редактора материала урока,
- 6) проектирование клиентской и серверной частей приложения,
- 7) выбор языков программирования,

- 8) реализация серверной части приложения,
- 9) реализация клиентской части приложения,
- 10) отладка и тестирование реализованного web-приложения,
- 11) анализ полученного результата.

Для получения объективных результатов необходимо провести тестирование [1], включающее тестирование функциональности, обработки некорректно введенных данных и альтернативных цепочек взаимодействия с приложением, а также проверку корректной работы приложения в разных браузерах.

Для тестирования функциональности необходимо составить несколько полноценных курсов, которые будут содержать все необходимые элементы и материалы.

1. Анализ предметной области

В сети есть множество систем для организации обучения, рассмотрим некоторые популярные из них.

Moodle.com – сервис для организации курсов, который доступен не только для организаций, но и для частных лиц. Достоинствами являются универсальность и распространенность, однако имеются и минусы, такие как:

- перегруженность: в moodle доступно множество различных элементов, и при необходимости организовать курс, в котором не требуется разнообразность данных, ориентирование во всех параметрах может быть затруднено;

- проблемы с локализацией: в сервисе доступно множество языков, однако не все являются равнозначными, то есть для некоторых языков в переводе доступна лишь часть интерфейса.

Stepik.org предоставляет пользователям различные настройки и варианты использования. Достоинствами этой платформы являются доступность и разнообразность, однако есть и минусы, такие как:

- ограниченный функционал в бесплатной версии,
- перегруженность разнообразными тематиками.

Тематика проекта – музыкальная теория, которая требует больше медиа и нотных материалов, нежели текста, а также особые типы заданий. В наиболее популярных образовательных платформах нет возможности добавить музыкальный материал.

Специализированные музыкальные курсы были найдены, например, на платформах skillbox.ru, pimaschool.ru, а также на локальных сайтах музыкальных школ, однако это определенные курсы, разработанные командой или музыкальной школой, и возможности учителю свободно составить и распространить курс ученикам выявлено не было.

Web-приложение для организации обучения и проверки знаний основ музыкальной теории призвано разнообразить процесс обучения в

музыкальной школе и сделать его более увлекательным, помочь начинающим музыкантам сделать свои первые шаги в области этого прекрасного искусства, а опытным музыкантам улучшить свои знания и помочь менее опытным музыкантам их приобрести, нацелено на свободный доступ к созданию и распространению музыкальных курсов и должно содержать функциональные возможности, необходимые для этой предметной области.

2. Функциональные требования к приложению

На основании анализа предметной области были определены следующие функциональные требования:

- возможность регистрации и авторизации;
- возможность создания, настройки и распространения музыкальных курсов;
- возможность создания в курсе уроков и их настройка;
- возможность добавления текстового и медиа материалов к уроку;
- возможность редактирования и добавления нотного материала к уроку;
- возможность создания тестирования и прикрепления его к уроку;
- возможность добавления отзыва к курсу и просмотра отзывов других пользователей;
- возможность распространения и нахождения курса по уникальному коду;
- возможность прохождения курсов, созданных другими пользователями;
- возможность отслеживания прогресса прохождения курсов.

Диаграмма вариантов использования изображена на рисунке 1.

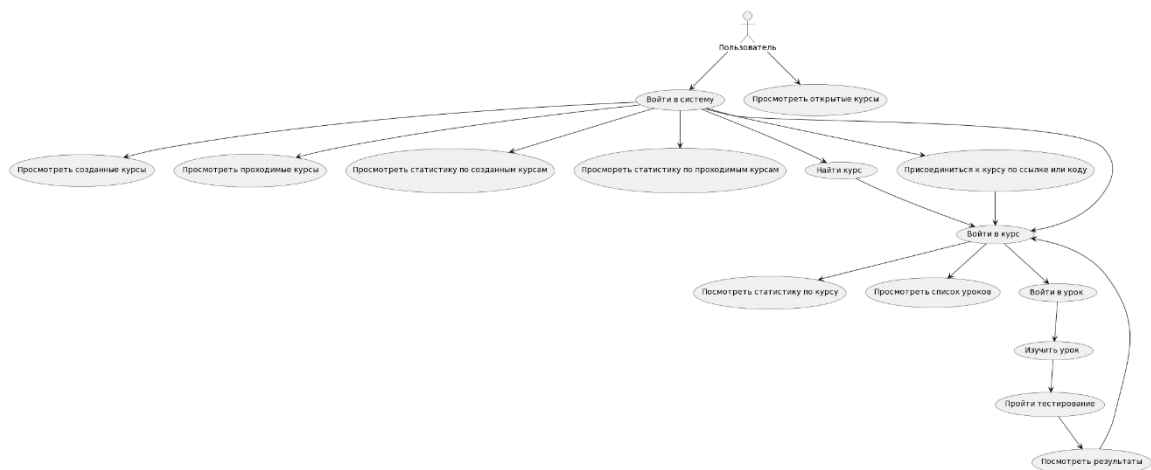


Рисунок 1 – Диаграмма вариантов использования

Разделения пользователей на роли преподавателя и ученика не предусмотрено, что означает, что каждому пользователю доступен функционал и создания, и прохождения курсов.

3. Архитектура проекта

Для реализации серверной части приложения был выбран фреймворк Django [2] языка программирования Python [3], поскольку в нем реализована вся необходимая логика взаимодействия компонентов web-приложения.

Приложение разбивается на подсистемы, каждая из которых предназначена для определенного логического блока (разбиение не является обязательным процессом). Каждая подсистема имеет следующие основные элементы структуры:

- `models.py` хранит модели базы данных, необходимые для подсистемы, модели реализуются с помощью классов с необходимыми атрибутами. После определения моделей необходимо провести миграции для сохранения обновлений в базу данных;
- директория `migrations` хранит все миграции базы данных;
- в `forms.py` происходит формирование списка данных (полей таблиц базы данных), которые необходимы для ввода или редактирования на стороне клиента;
- `urls.py` выполняет логику переадресации и перехода по ссылкам;
- `views.py` хранит представления, выполняет практически всю логику обработки данных, взаимодействует с описанными элементами структуры и является связующим звеном между серверной и клиентской частями.

Клиентская часть web-приложения реализована с помощью языков HTML [4], CSS [5] и JavaScript [6] и связана с серверной частью представлениями Django.

Для более детального понимания схемы взаимодействия элементов структуры рассмотрим последовательность передачи управления на примере редактирования профиля:

- 1) на странице профиля нажимается кнопка «Изменить», у которой в коде html прописан url с именем `edit_profile` и передается `id` пользователя;

2) соответствующий url ищется в списке всех адресов (файл `urls.py`), после нахождения вызывается метод `get` с параметром `id` у представления, указанном в найденном url (файл `views.py`);

3) в методе `get` получается объект пользователя, который хочет изменить свой профиль, а также формы (из `forms.py`) для редактирования основной и дополнительной информации о пользователе;

4) с серверной стороны возвращается информация о том, какую страницу необходимо отрисовать, какие формы подставить, какой пользователь запросил изменения и его нынешние данные;

5) на стороне клиента заполняются нужные поля и нажимается кнопка «Сохранить», которая вызывает метод `post` того же представления;

6) в методе `post` получается объект пользователя, который хочет изменить свой профиль, а также соответствующие формы из `forms.py`;

7) в случае корректности форм происходит сохранение данных и переадресация на страницу личного профиля;

8) в случае некорректности форм переадресации не происходит.

Описанный процесс может иметь различия в зависимости от типа запроса, необходимой информации и цели, например, url может вызываться без параметров, могут не запрашиваться формы, но общий сценарий имеет описанную выше последовательность действий.

4. Описание базы данных

База данных была спроектирована с помощью MySQL Workbench [7] и реализована с помощью миграций моделей фреймворка Django языка программирования Python.

Таблица User является встроенной, хранит в себе основную информацию о пользователях системы и содержит поля:

- id – идентификатор пользователя;
- email – адрес электронной почты пользователя;
- first_name – имя пользователя;
- last_name – фамилия пользователя;
- password – пароль в зашифрованном виде;
- username – сетевое имя пользователя.

Сетевое имя пользователя, адрес электронной почты и пароль вводятся пользователем при регистрации, идентификатор заполняется автоматически, а имя и фамилию пользователь может ввести при изменении данных пользователя в профиле. Пароль хранится в зашифрованном виде, просмотреть его нельзя даже через панель администратора Django [8].

Таблица Userdata соединена с таблицей User связью один-к-одному, хранит в себе дополнительную информацию о пользователях системы и содержит поля:

- id – идентификатор;
- user – поле связи с таблицей User;
- birthdate – дата рождения;
- fathersname – отчество;
- musicedu – музыкальное образование;
- phone – номер телефона;
- profile_avatar – фото профиля.

При регистрации пользователя автоматически создается запись в таблице и заполняются идентификатор и поле связи с таблицей User, а также

фото профиля заполняется изображением по умолчанию. Ввести отчество, дату рождения, музыкальное образование и номер телефона и выбрать другое фото профиля можно позже при изменении данных пользователя в профиле.

Таблица `Courses` хранит информацию о курсах и содержит поля:

- `id` – идентификатор курса;
- `course_creator` – создатель курса, внешний ключ к таблице пользователей;
- `course_avatar` – фото курса;
- `course_creation_date` – дата создания курса;
- `course_description` – описание курса;
- `course_name` – название курса;
- `course_theme` – тематика курса;
- `is_course_active` – логическое значение, хранящее информацию о том, открыт ли курс для просмотра.

При создании курса пользователем вводятся название, тематика и описание, загружается фото курса, а также выбирается, будет ли курс активным. Если при создании курса фото выбрано не было, то добавляется изображение по умолчанию. Идентификатор, создатель и дата создания курса заполняются автоматически.

Таблица `CourseProgress` хранит информацию о прогрессе пользователя при прохождении курса и содержит поля:

- `id` – идентификатор;
- `course` – проходимый курс, внешний ключ на таблицу курсов;
- `user` – пользователь, проходящий курс, внешний ключ на таблицу пользователей;
- `lesson` – номер урока, на котором остановился пользователь.

При записи пользователя на курс добавляется запись в таблицу прогресса, где идентификатор, курс и пользователь заполняются

автоматически, урок изначально выставляется нулевым значением, а затем обновляется по мере прохождения курса.

Таблица CourseFeedback хранит информацию об отзывах и оценках пользователей на курсы и содержит поля:

- id – идентификатор;
- course – курс, на который оставляется отзыв, внешний ключ на таблицу курсов;
- user – пользователь, оставивший отзыв, внешний ключ на таблицу пользователей;
- date – дата отзыва;
- feedback – текст отзыва;
- rating – оценка курса по пяти-бальной шкале.

При добавлении отзыва идентификатор, курс, пользователь и дата заполняются автоматически, текст отзыва вводится пользователем, а оценка формируется согласно количеству выбранных звезд в форме.

Таблица Lesson хранит информацию об уроках и содержит поля:

- id – идентификатор;
- lesson_course – внешний ключ на курс, в котором находится урок;
- lesson_creator – создатель урока;
- test – внешний ключ на прикрепленный тест;
- is_lesson_active – логическое значение, хранящее информацию о том, открыт ли урок для просмотра;
- lesson_content – материал урока;
- lesson_creation_date – дата создания урока;
- lesson_description – описание урока;
- lesson_name – название урока;
- lesson_priority – приоритет урока, по которому происходит сортировка списка уроков в курсе;
- lesson_theme – тематика курса.

При добавлении урока в курс пользователем вводятся название, тематика, описание, приоритет урока, отмечается, будет ли урок активным, добавляется материал, а также при необходимости прикрепляется тестирование. Идентификатор, ссылка на курс, создатель и дата создания урока заполняются автоматически.

Таблица Test хранит информацию о тестах и содержит поля:

- id – идентификатор;
- test_creator – создатель теста;
- name – название теста.

Название теста вводится пользователем при сохранении, а идентификатор и создатель заполняются автоматически.

Таблица TestQuestion хранит информацию о вопросах тестов и содержит поля:

- id – идентификатор;
- duration – ограничение по времени на вопрос;
- media – медиафайл (при необходимости);
- num_answers – количество вариантов ответа;
- question – полный текст вопроса;
- question_short – краткий текст вопроса;
- score – максимальное количество баллов за ответ на вопрос;
- type – тип вопроса.

При добавлении вопроса в тест пользователем заполняются поля полного и краткого текста вопроса, ограничения по времени на вопрос, максимального количества баллов за ответ, выбирается тип вопроса, количество вариантов ответа, а также при необходимости добавляется медиафайл. Идентификатор заполняется автоматически.

Таблица TestAnswerVariant хранит варианты ответа на вопросы тестов и содержит поля:

- id – идентификатор;

— question – вопрос, к которому прикреплен вариант ответа, внешний ключ на таблицу вопросов теста;

— answer – текст варианта ответа.

При добавлении варианта ответа к вопросу пользователем заполняется текст варианта ответа, а идентификатор и вопрос, к которому прикрепляется вариант, заполняются автоматически после сохранения вопроса.

Таблица TestAnswer хранит ответы пользователей на вопросы тестов и содержит поля:

— id – идентификатор;

— question – вопрос, на который дан текущий ответ, внешний ключ на таблицу вопросов теста;

— user – пользователь, давший текущий ответ, внешний ключ на таблицу пользователей;

— answer – данный ответ;

— checked – логическое значение, хранящее информацию о том, проверен ли ответ;

— comment – комментарий преподавателя (заполняется при необходимости во время проверки);

— media – медиафайл ответа;

— score – набранный балл.

Пользователем вводятся текст ответа либо прикрепляется медиафайл в зависимости от типа вопроса, остальные поля заполняются автоматически. Значение checked при ответе записывается True для автоматически проверяемых вопросов, и False для вопросов, которые проверяет преподаватель, score аналогично либо выставляется сразу, либо записывается нулевым значением до проверки преподавателя, а comment в начале является пустым полем. Для вопросов, которые требуют ручной проверки, комментарий преподавателя и набранный балл вводятся проверяющим, после отправки значение checked записывается как True.

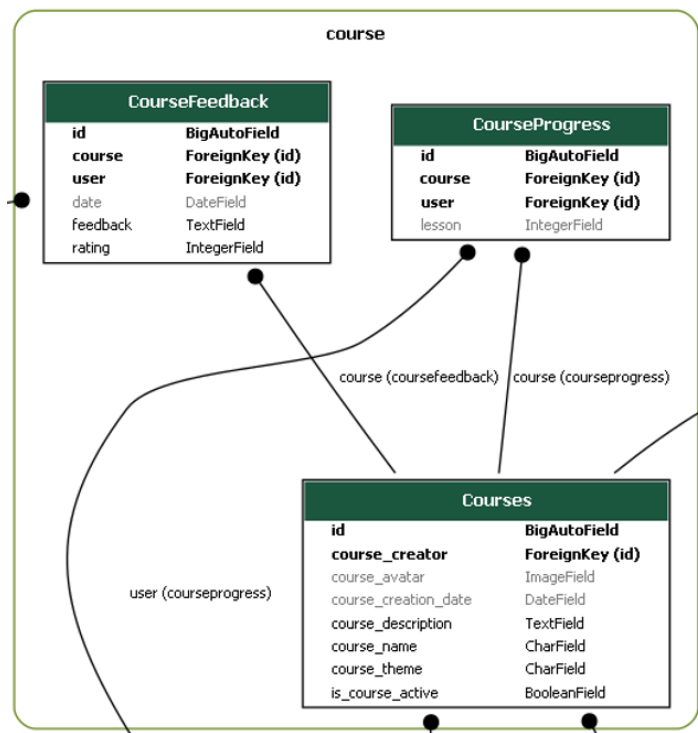


Рисунок 3 – Фрагмент базы данных с курсами

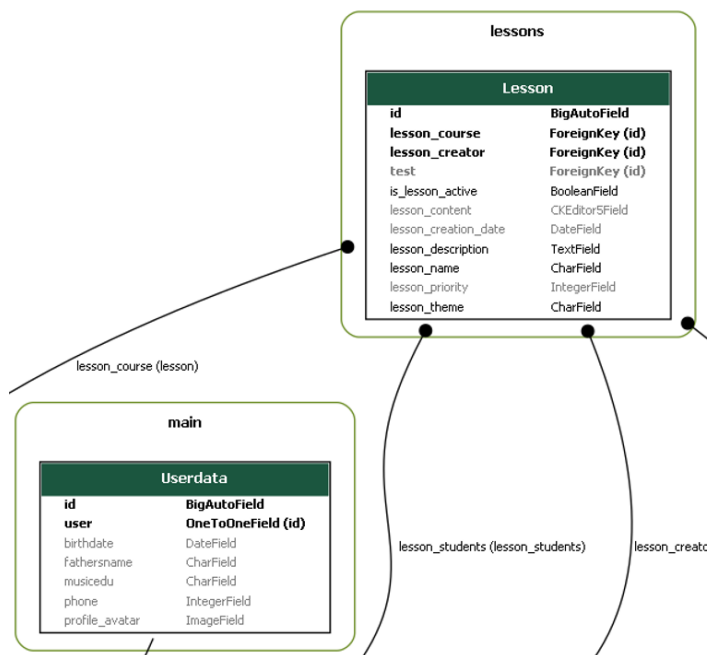


Рисунок 4 – Фрагмент базы данных с уроками и дополнительными данными пользователя

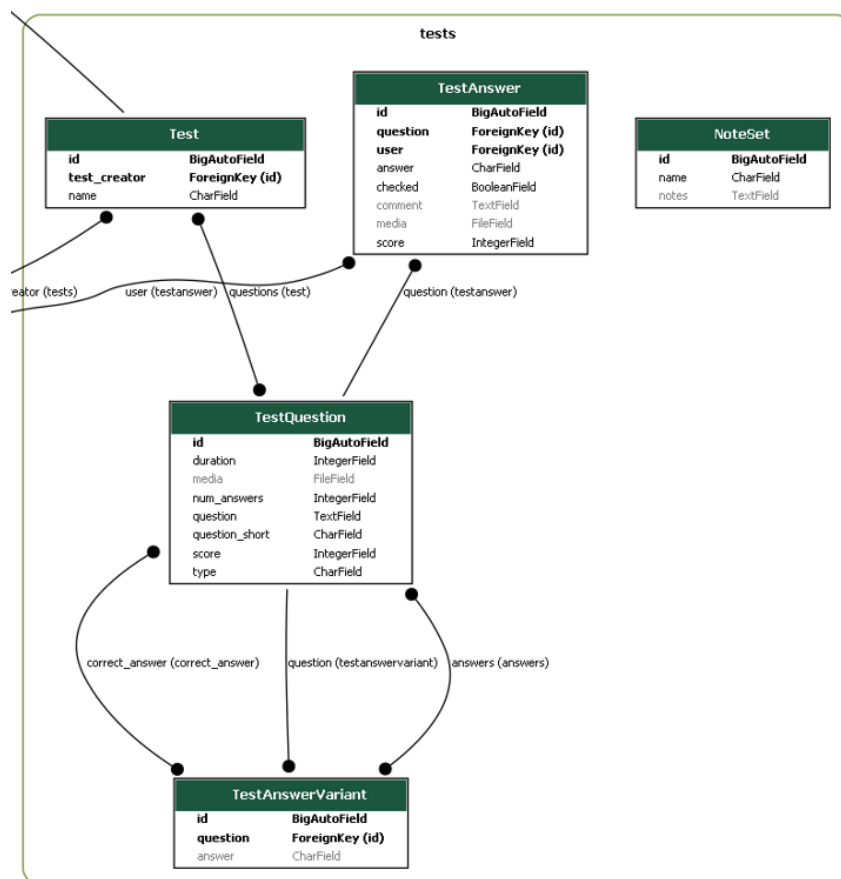


Рисунок 5 – Фрагмент базы данных с тестированием и нотным материалом

5. Реализация серверной части приложения

Серверная часть web-приложения (далее – Django-проект) написана с помощью языка программирования Python и фреймворка Django.

Django-проект состоит из следующих подсистем:

- musicserver – корневая, главная директория проекта, в которой хранятся общие настройки проекта;
- main – основная подсистема, в которой хранятся все страницы, а также всё, что связано с пользователем;
- myauth – подсистема, в которой реализована регистрация и авторизация;
- course – подсистема, в которой реализованы курсы;
- lessons – подсистема, в которой реализованы уроки;
- tests – подсистема, в которой реализовано всё, что связано с тестами и нотным редактором.

Также в директории проекта хранятся следующие данные:

- media – директория с media-файлами;
- db.sqlite3 – база данных web-приложения;
- erd.dot, erd.png – файлы выгрузки базы данных;
- manage.py – основной файл взаимодействия с проектом.

Структура проекта представлена на рисунке 6.

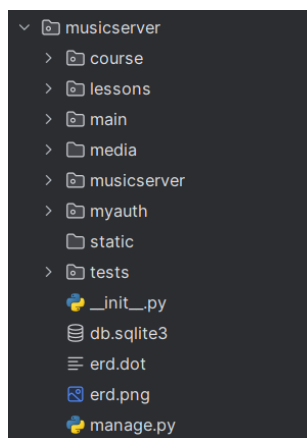


Рисунок 6 – Структура проекта

Подсистемы `main` и `myauth` отвечают за логику, связанную с пользователем и общими страницами. Основными представлениями являются:

- регистрация, авторизация и выход из системы;
- изменение данных профиля пользователя;
- отображение общих страниц.

Подсистема `course` отвечает за логику взаимодействия с курсами. Основными представлениями являются:

- добавление, редактирование и удаление курса;
- фильтрация и отображение проходимых и созданных курсов;
- отображение информации о курсе, а также уроков в нем;
- запись на курс и удаление с него;
- добавление и отображение отзывов и оценок курсов, а также подсчет средней оценки.

Подсистема `lessons` отвечает за логику взаимодействия с уроками. Основными ее представлениями являются:

- добавление, редактирование и удаление урока;
- отображение информации об уроке, материала и прикрепленной проверки усваиваемости материала.

Подсистема `tests` отвечает за логику, связанную с проверкой усваиваемости материала и нотной информацией. Основными представлениями являются:

- добавление, редактирование, удаление тестов;
- добавление, редактирование, удаление вопросов теста;
- обработка ответов на вопросы теста;
- организация проверки ответов преподавателем;
- нотный редактор;
- отображение нотной информации.

Редактор материала урока реализован с помощью `django-ckeditor-5` [9].

Проверка усваиваемости материала (также – тестирование) имеет несколько типов вопросов по принимаемым ответам:

- 1) один правильный вариант ответа: ответ обучающегося автоматически сравнивается с ответом, указанным как правильный, и назначаются баллы;
- 2) несколько правильных вариантов ответа: множество вариантов ответа ученика автоматически сравнивается с множеством ответов, отмеченных как правильные, и назначаются набранные баллы;
- 3) открытый ответ: этот тип вопроса проверки не содержит правильное решение, ответ ученика отправляется на проверку преподавателю, после проверки результат и комментарий виден в общей сводке результатов тестирования;
- 4) ответ, требующий прикрепление файла: этот тип вопроса проверки так же не содержит правильное решение, ответ ученика проверяется аналогично открытому ответу;
- 5) ответ, в котором требуется отметить последовательность нот: этот тип вопроса проверки не содержит правильное решение, ответ ученика проверяется аналогично открытому ответу.

При редактировании вопроса теста первого и второго типа варианты ответа добавляются динамически. Пользователь вводит текст варианта, нажимает «Добавить», и вариант добавляется во временный список. Также есть возможность очистки введенных вариантов ответа. Сохранение вариантов и прикрепление их к вопросу осуществляется после сохранения всего вопроса.

Код курса представляет собой идентификатор курса, по которому происходит поиск курса при необходимости.

6. Реализация клиентской части приложения

В начале для визуализации общей картины был создан макет будущего приложения, который разрабатывался с помощью online-сервиса Figma [10]. Примеры фрагментов разработанного макета представлены на рисунках 7 – 10.

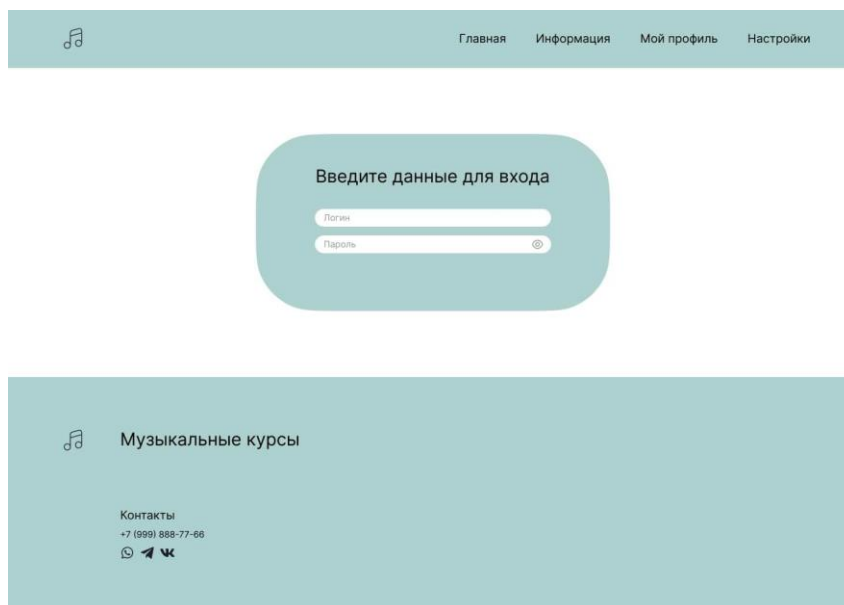


Рисунок 7 – Фрагмент макета страницы авторизации

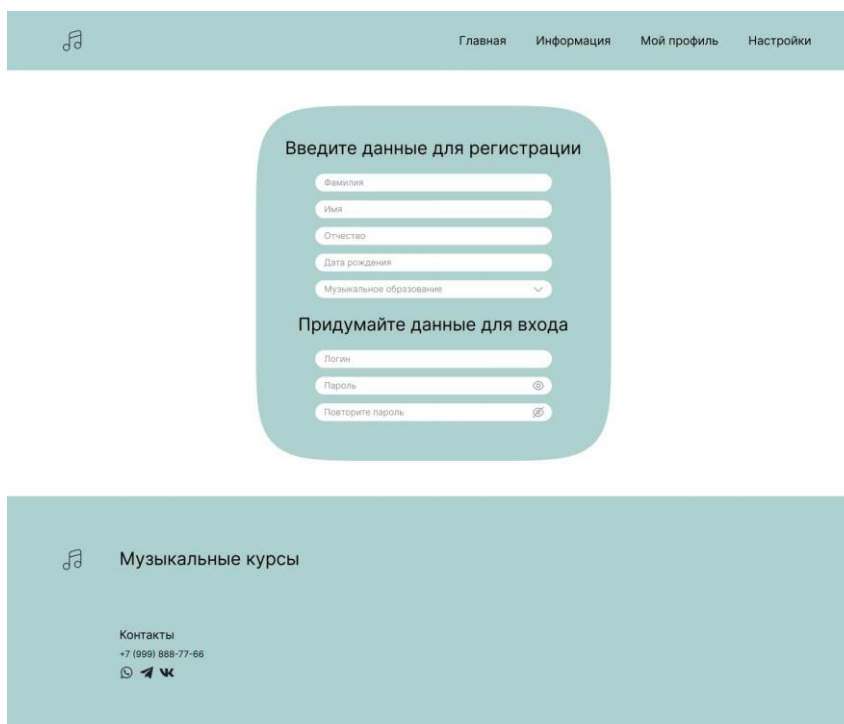


Рисунок 8 – Фрагмент макета страницы регистрации

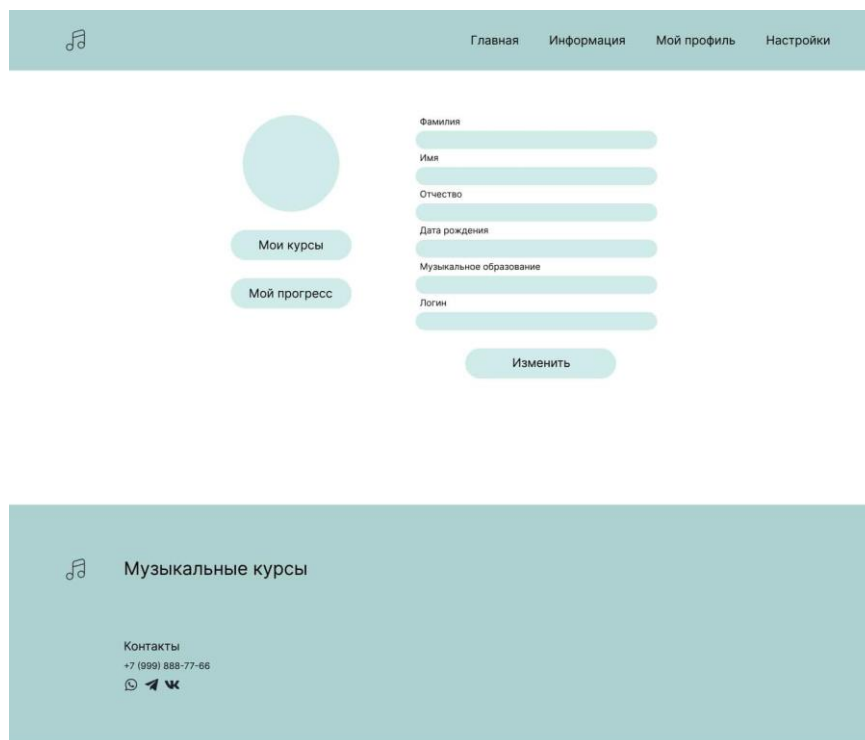


Рисунок 9 – Фрагмент макета страницы личного профиля

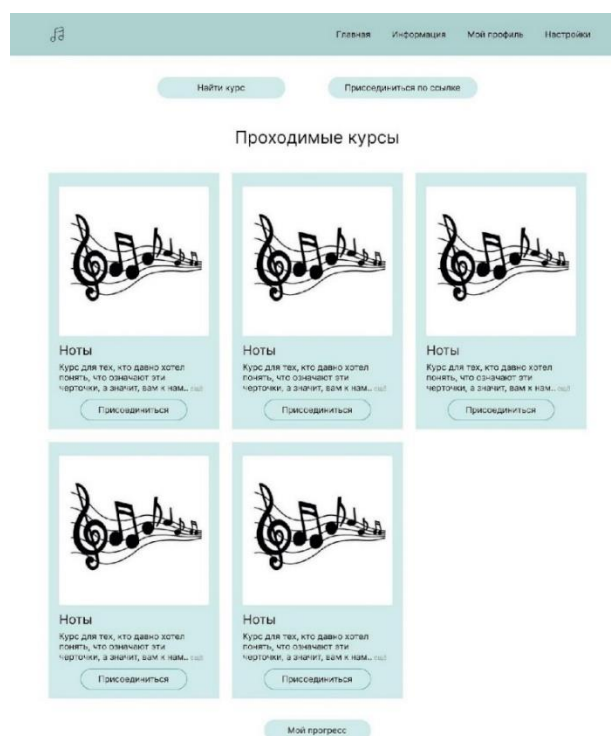


Рисунок 10 – Фрагмент макета страницы курсов

По разработанным макетам впоследствии проводилась верстка. В начале был создан шаблон с общими элементами страниц, после чего он дополнялся необходимой информацией, требуемой для каждой из страниц. Основными страницами являются общая страница курсов, страница курса с информацией

о нем и списком уроков, страница урока с информацией о нем, материалом и разделом проверки знаний, а также страницы с редактированием курсов и уроков.

Также на стороне клиента происходит обработка таких событий, как:

- добавление вариантов ответа на вопрос теста,
- выбор варианта ответа как правильный,
- удаление вариантов ответа к вопросу теста,
- взаимодействие с нотным редактором.

7. Реализация нотного редактора

Нотный редактор – элемент, который необходим для самостоятельного набора последовательностей нот для включения в материал урока или организации проверки знаний.

Реализован нотный редактор с помощью элемента HTML Canvas [11] и языка JavaScript. Есть три основных события:

- движение мыши;
- нажатие левой кнопки мыши;
- нажатие клавиш стрелок «вверх», «вниз» на клавиатуре.

Движение мыши отслеживается для того, чтобы при попадании в область следующей по порядку ноты происходила отрисовка добавочных линий, тем самым подсказывая допустимую область нот (рисунок 11).

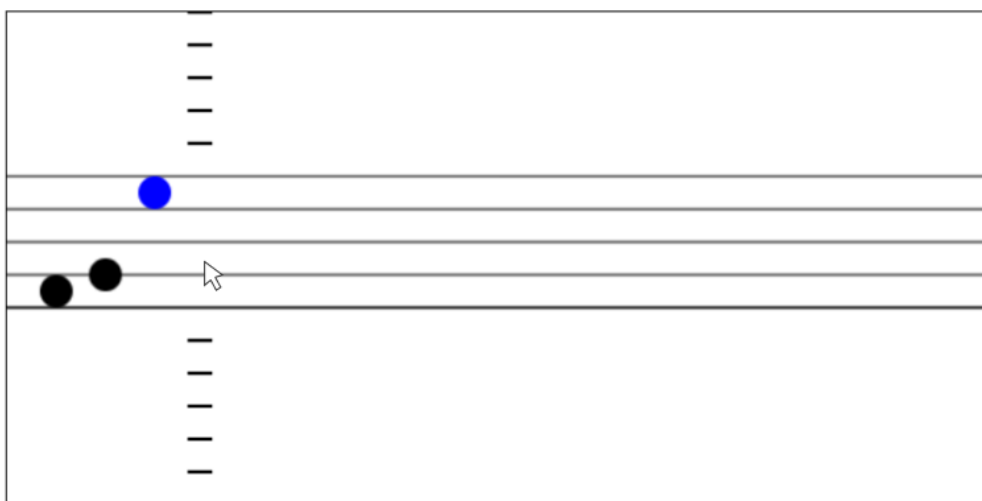


Рисунок 11 – Движение мыши на нотном редакторе

Нажатие левой кнопки мыши способствует следующим изменениям:

- при нажатии в области следующей по порядку ноты добавляется нота, которая до следующего события выделена синим, что говорит о том, что она доступна для изменений клавишами стрелок (рисунок 11);
- при нажатии на уже отрисованную ноту эта нота выбирается для изменений и выделяется синим (рисунок 12);
- при нажатии в другой области отменяется выделение ноты.

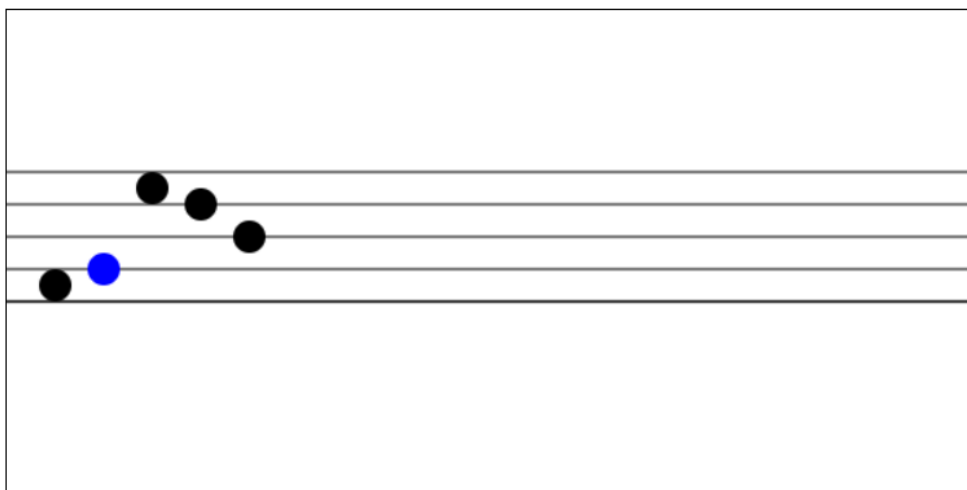


Рисунок 12 – Выбранная нота на нотном редакторе

Нажатие клавиши стрелки «вверх» на клавиатуре изменяет ноту согласно следующим правилам:

- если перед нажатием у ноты не было дополнительных свойств, то перед нотой добавляется знак «диез» (рисунок 13);
- если перед нажатием у ноты было свойство «бемоля», то оно отменяется;
- если перед нажатием у ноты было свойство «диеза», то оно отменяется и нота изменяется на следующую по высоте.

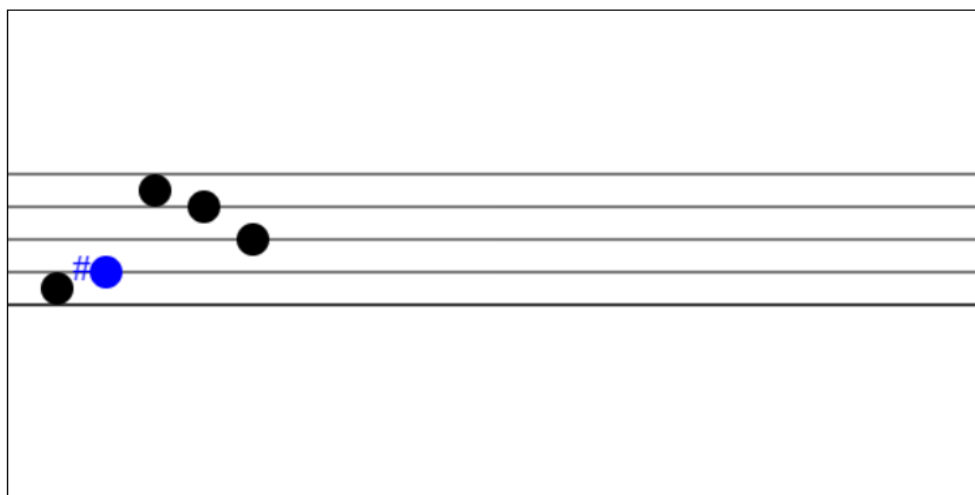


Рисунок 13 – Добавление диеза перед нотой

Нажатие клавиши стрелки «вниз» на клавиатуре изменяет ноту согласно следующим правилам:

- если перед нажатием у ноты не было дополнительных свойств, то перед нотой добавляется знак «бемоль» (рисунок 14);
- если перед нажатием у ноты было свойство «диеза», то оно отменяется;
- если перед нажатием у ноты было свойство «бемоля», то оно отменяется и нота изменяется на предыдущую по высоте.

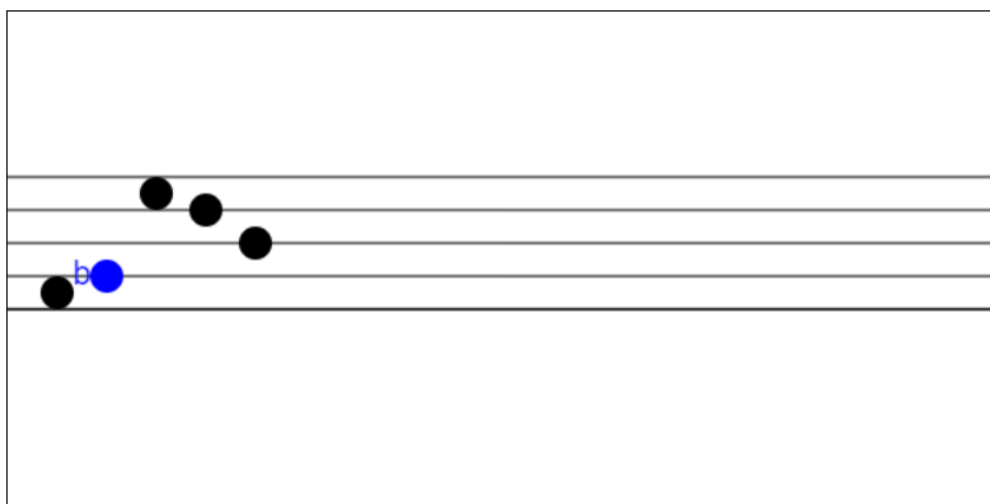


Рисунок 14 – Добавление бемоля перед нотой

Нотная последовательность после сохранения преобразуется в строку определенного вида, которая хранит координаты ноты, порядковый номер, а также свойства «бемоля» и «диеза».

Нотные фрагменты реализованы не только для редактирования, но и для чтения, что необходимо для отображения в материале урока без возможности редактирования. В таком режиме нотная последовательность отрисовывается по координатам и свойствам.

8. Демонстрация работы приложения и тестирование

При переходе в приложение открывается главная страница, на которой представлена основная информация о проекте (рисунок 15). На этой же странице есть возможность найти курс по коду, а также представлен список всех курсов, которые находятся в общем доступе, с их оценками (рисунок 16).

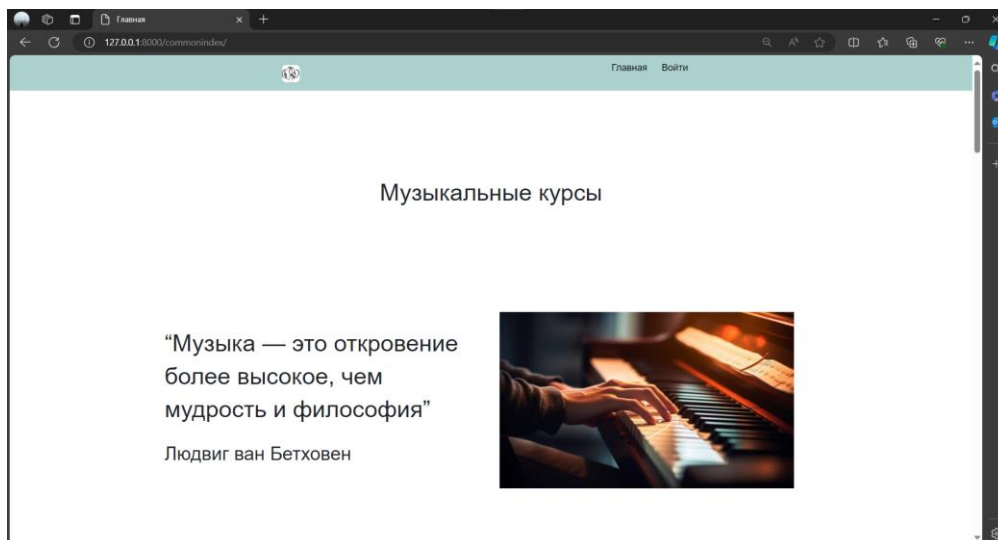


Рисунок 15 – Главный экран

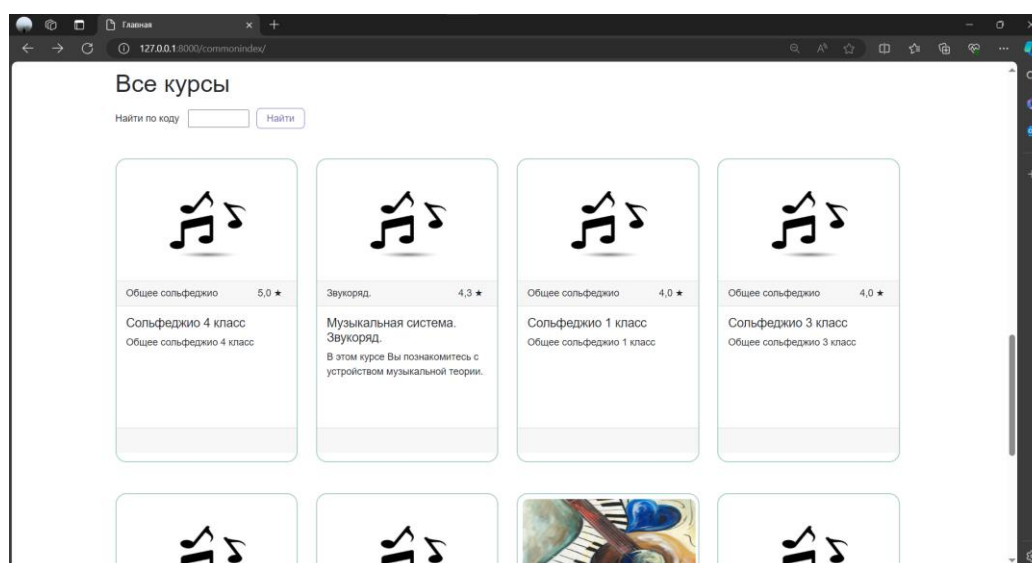


Рисунок 16 – Список курсов с оценками на главной странице

С главной страницы можно перейти на страницу регистрации и авторизации. При неправильном вводе информации отображается соответствующая ошибка (рисунки 17 – 19).

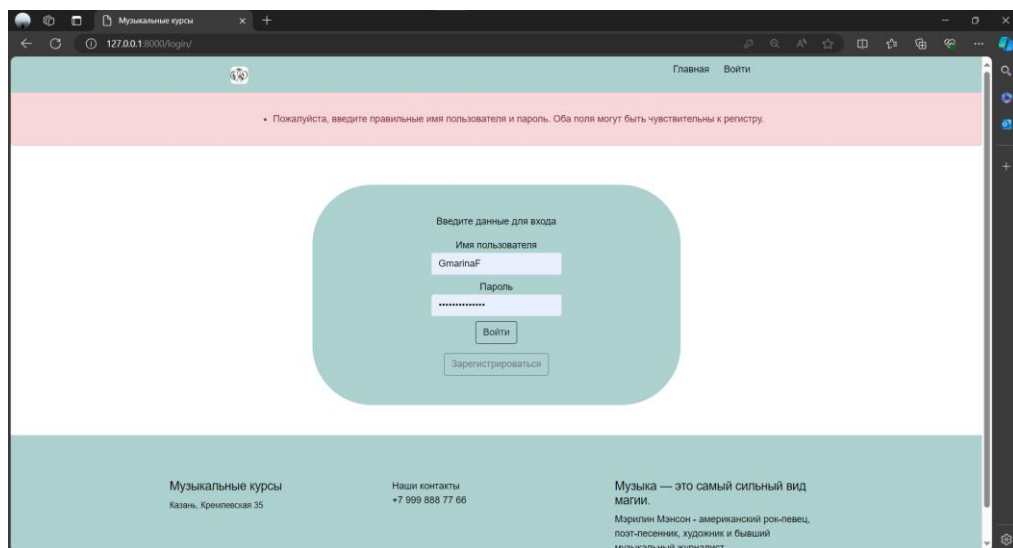


Рисунок 17 – Ошибка неверно введенных данных при авторизации

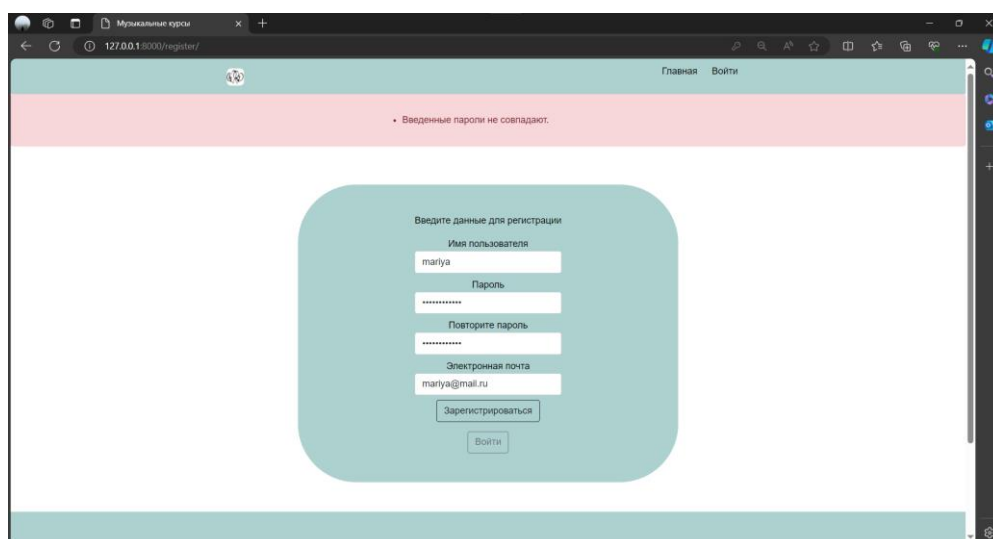


Рисунок 18 – Ошибка разных паролей при регистрации

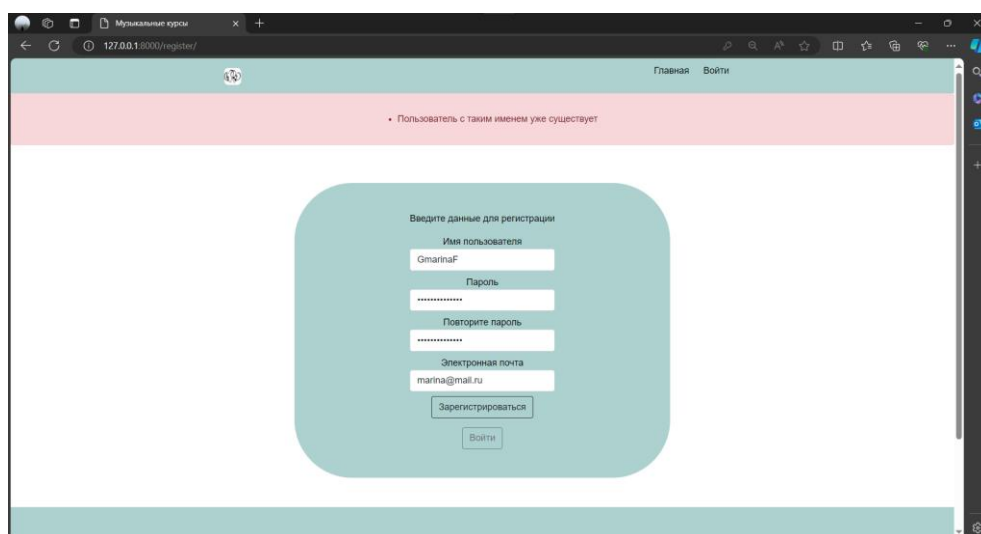


Рисунок 19 – Ошибка повторного использования имени пользователя при регистрации

Идентификация пользователя происходит через имя пользователя, а пароль хранится в хэшированном виде, то есть узнать его невозможно, даже через панель администратора Django (рисунок 20).

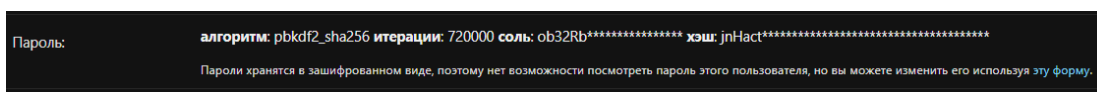


Рисунок 20 – Хранение пароля

После регистрации и авторизации происходит переадресация на главный экран, который идентичен описанному выше, за исключением появившегося навигационного меню.

В навигационном меню есть возможность перейти в личный профиль (рисунок 21), в котором есть возможность указать и изменить информацию о себе. На странице изменения присутствует кнопка отмены, нажатие на которую возвращает на страницу профиля и не сохраняет изменений.

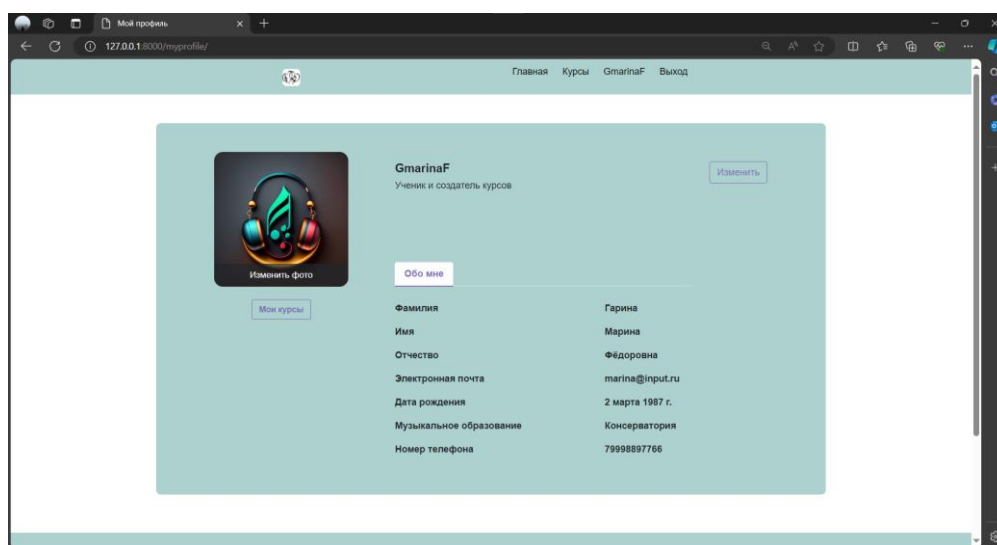


Рисунок 21 – Страница пользователя

Также в навигационном меню есть возможность перейти на страницу курсов, где указываются курсы, которые проходит пользователь, и курсы, которые он создал (рисунки 22, 23). В разделе проходимых курсов есть возможность просмотреть результаты тестов, которые отображают информацию о тестах, набранных баллах и позволяют перейти в соответствующему уроку для просмотра более подробной информации.

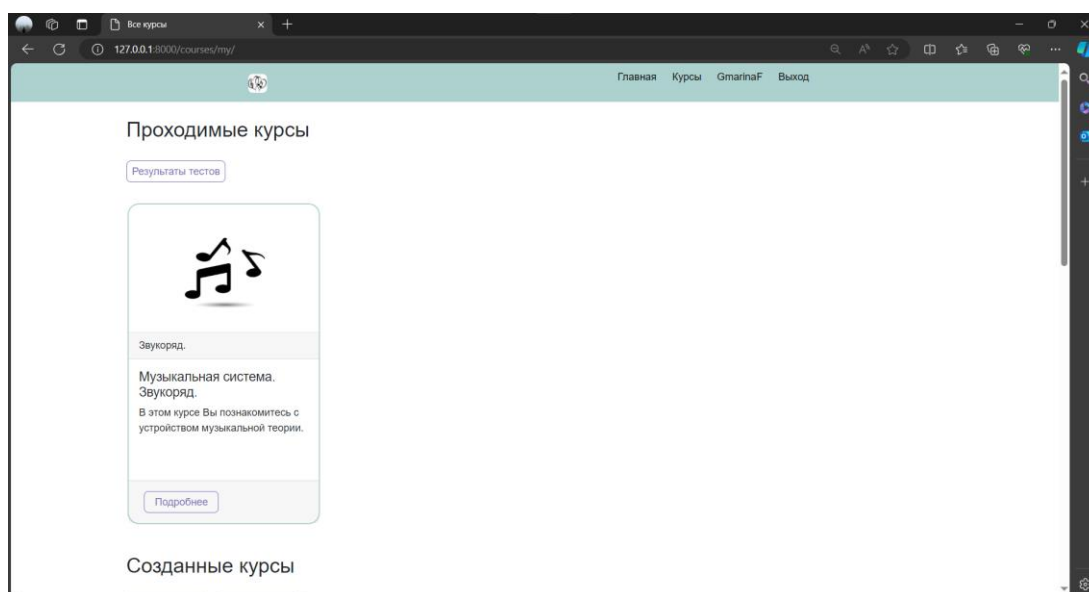


Рисунок 22 – Проходимые курсы пользователя

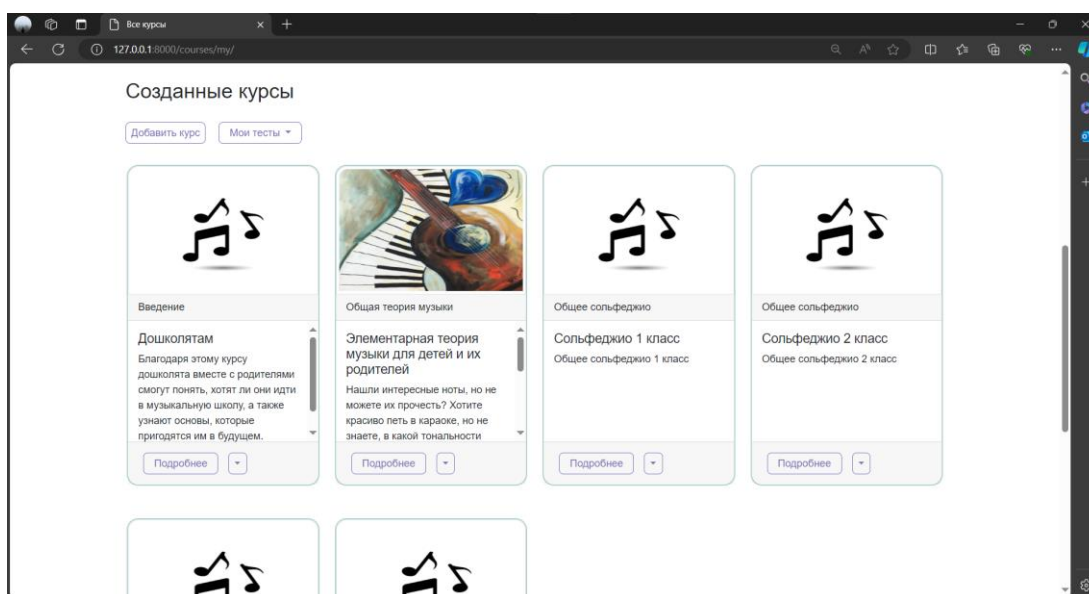


Рисунок 23 – Созданные курсы пользователя

Вкладка «Мои тесты» позволяет открыть раздел проверки тестов, в который отправляются все ответы учеников на вопросы открытого типа, открыть список всех созданных тестов, а также добавить тест (рисунок 24). После добавления теста открывается страница теста, где отображаются все вопросы и имеется возможность добавить, а также изменить вопрос теста (рисунки 25, 26).

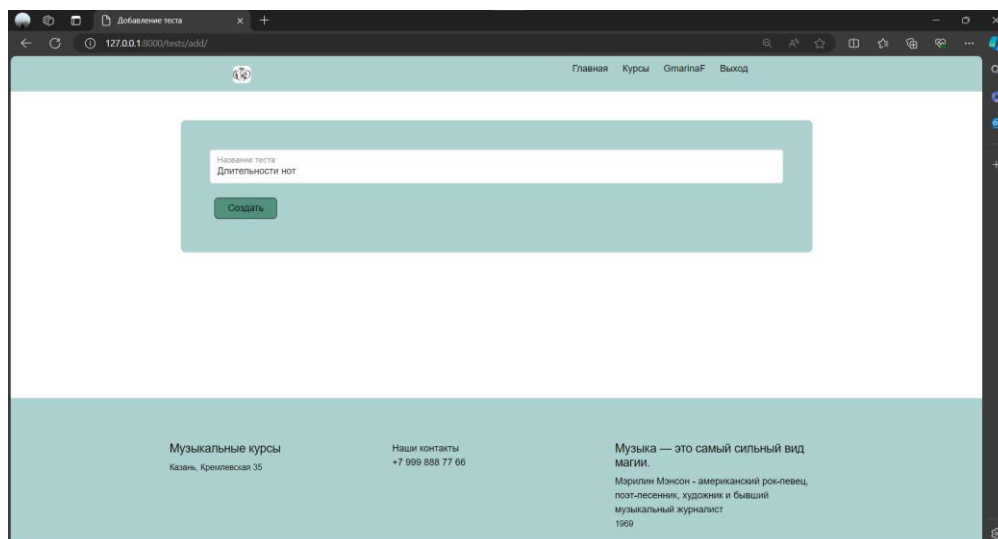


Рисунок 24 – Форма добавления теста

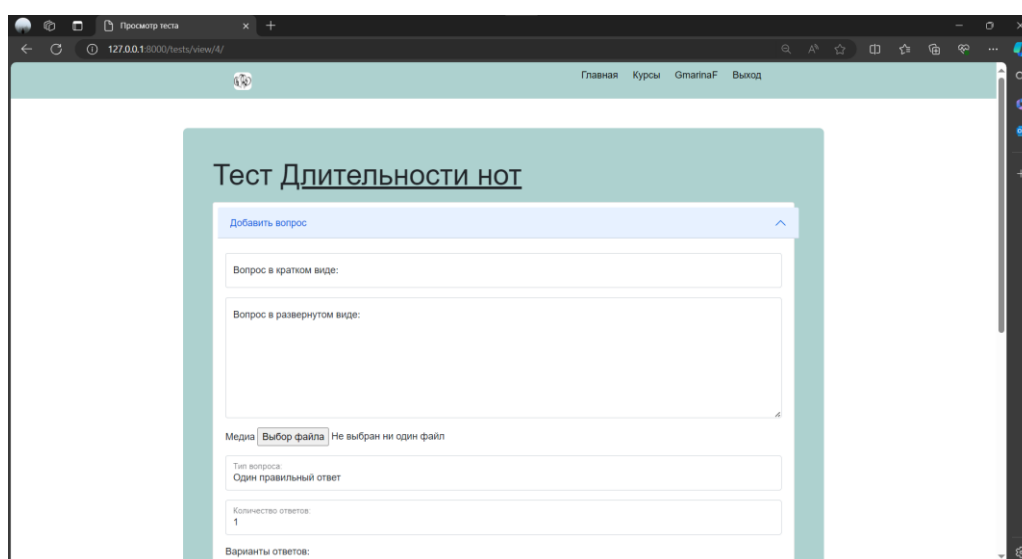


Рисунок 25 – Начало формы добавления вопроса теста

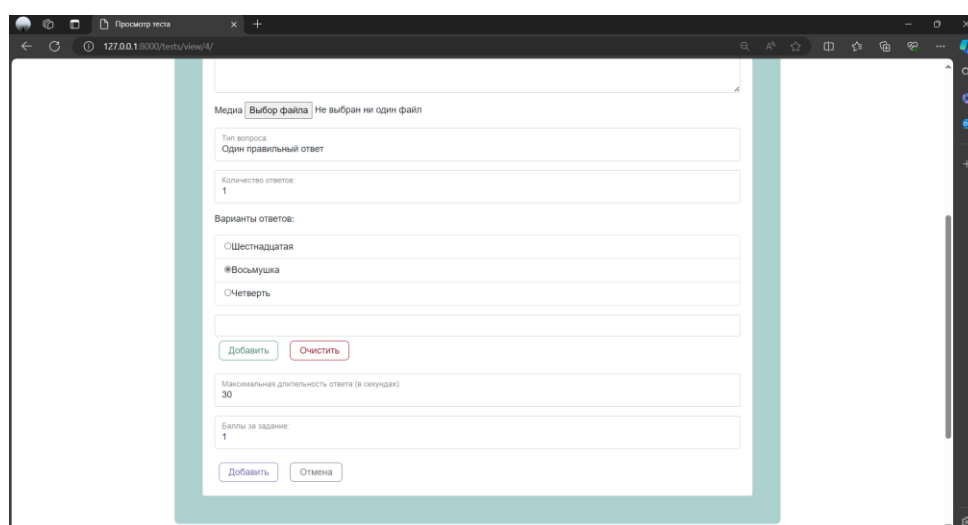


Рисунок 26 – Продолжение формы добавления вопроса теста

При переходе на курс, который пользователь не создавал и не проходил, у него открывается перечень уроков с возможностью просмотреть лишь первый (рисунок 27) и появляется возможность записаться на курс, после чего появляется возможность просмотреть прогресс прохождения курса и открываются для просмотра и прохождения все уроки (рисунок 28).

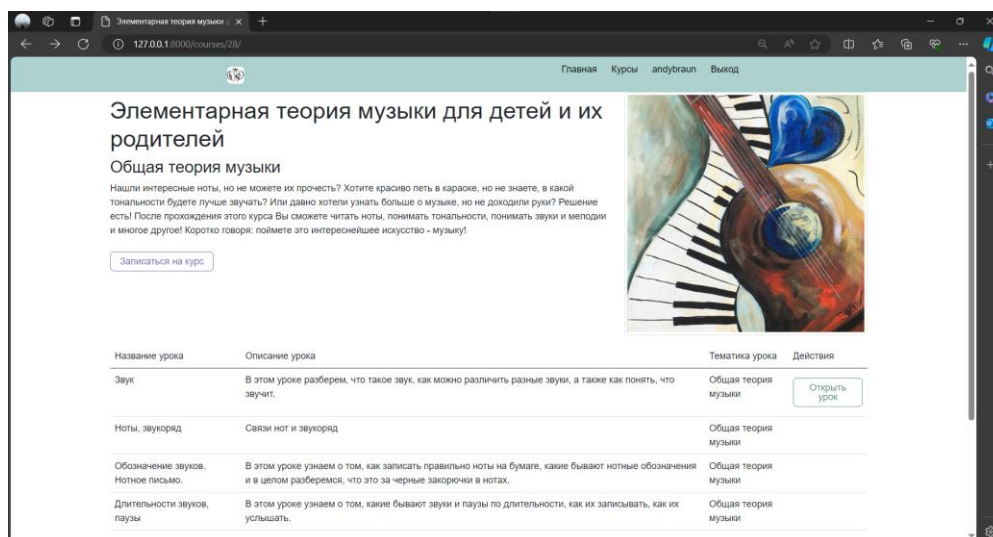


Рисунок 27 – Страница курса для пользователя, который не является учеником этого курса

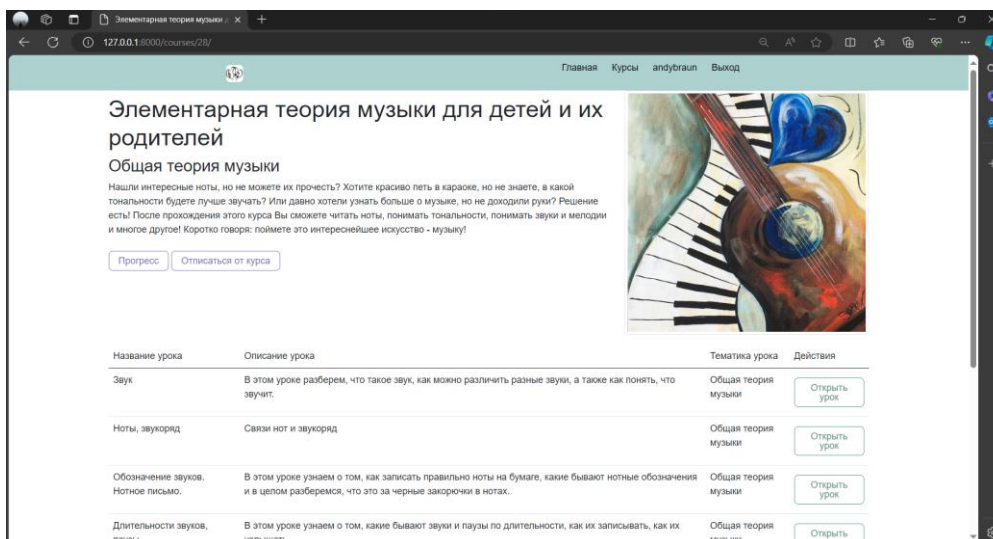


Рисунок 28 – Страница курса для пользователя, который является учеником этого курса

Если пользователь является создателем курса, то у него есть возможность просмотреть все отзывы на этот курс, а если пользователь не является создателем, то просмотреть (рисунок 29) и оценить (рисунок 30) курс (оставить отзыв на него).

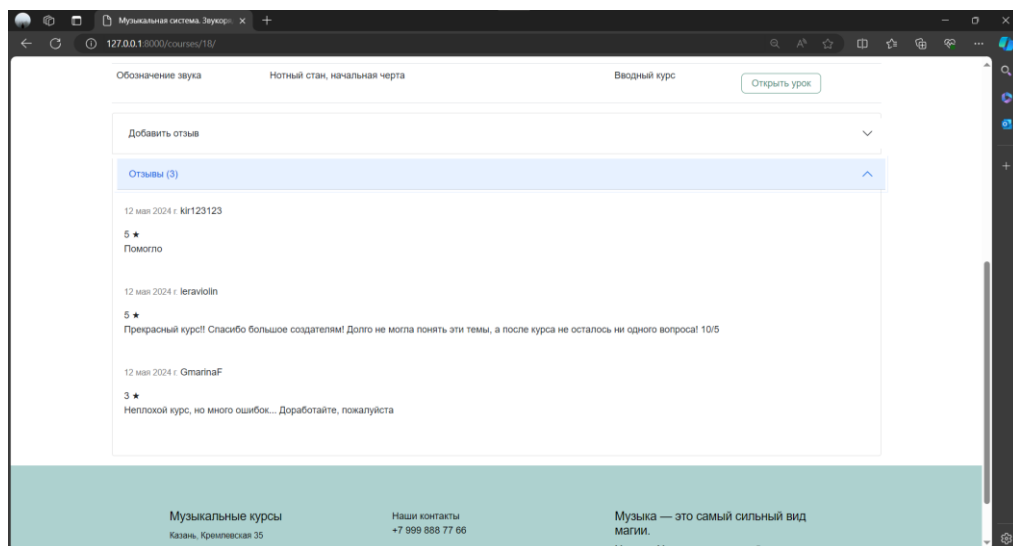


Рисунок 29 – Существующие отзывы на курс

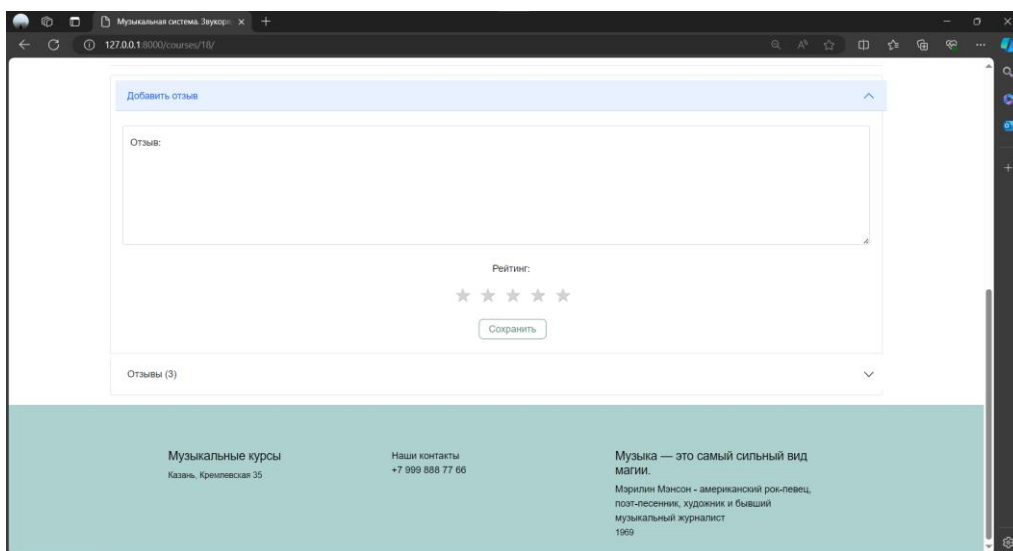
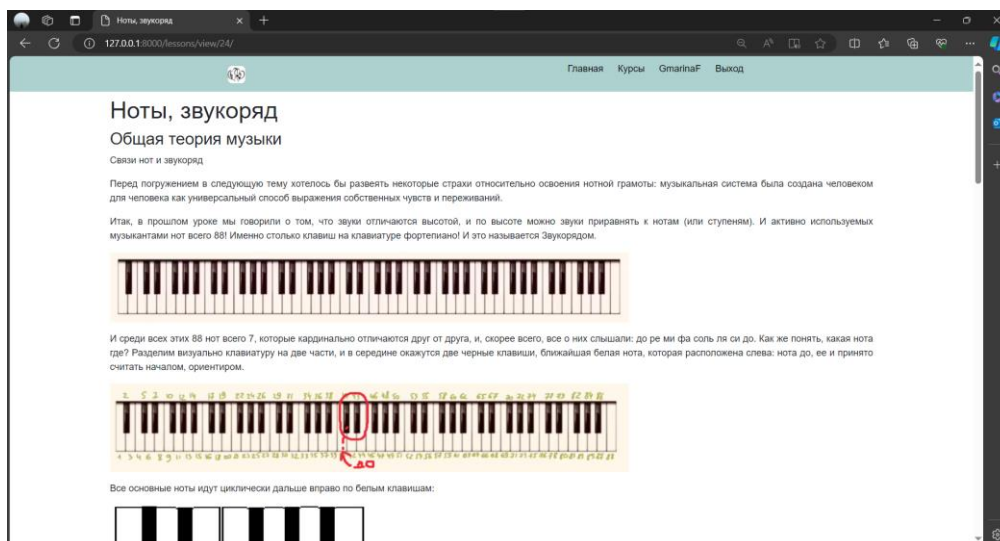
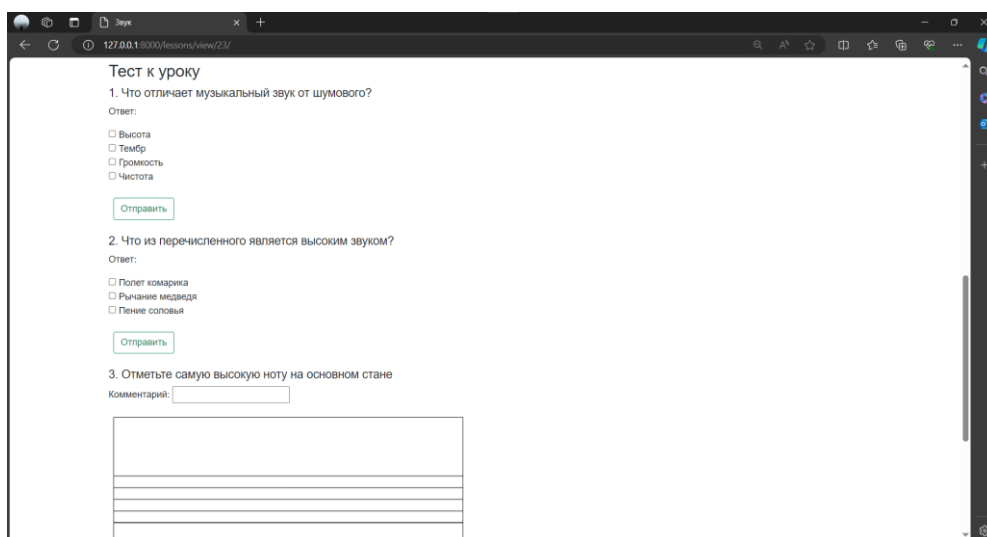


Рисунок 30 – Форма оценки курса

После перехода к уроку пользователю открывается материал выбранного урока (рисунок 31).



В конце страницы урока представлен фрагмент с проверкой знаний по пройденному уроку (рисунок 32). После прохождения тестирования в конце урока отображаются результаты пройденного теста.



Страницы редактирования курсов (рисунок 33) и уроков (рисунки 34, 35) имеют необходимые поля для ввода, а также кнопки отмены, которая без изменения данных возвращает на страницу всех курсов и текущего курса соответственно.

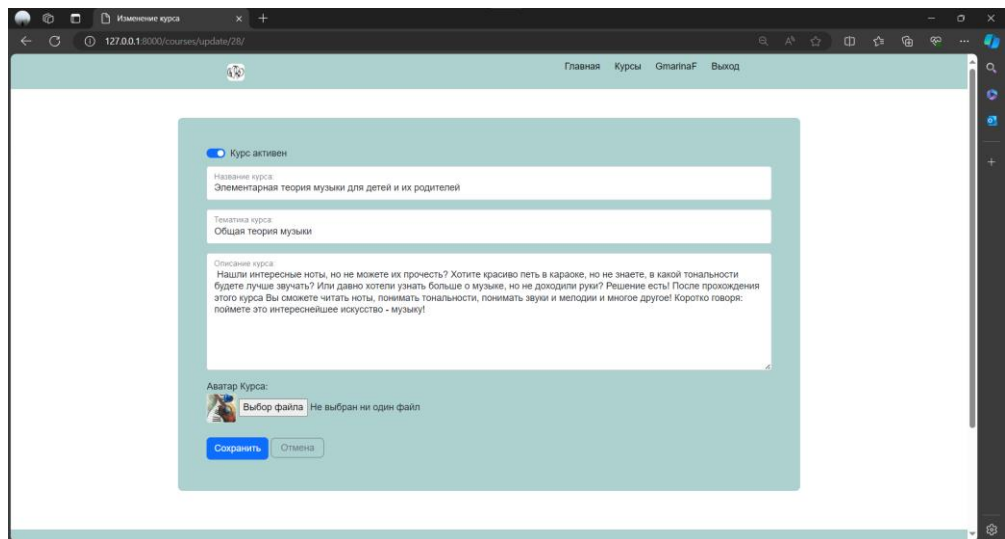


Рисунок 33 – Страница редактирования курса

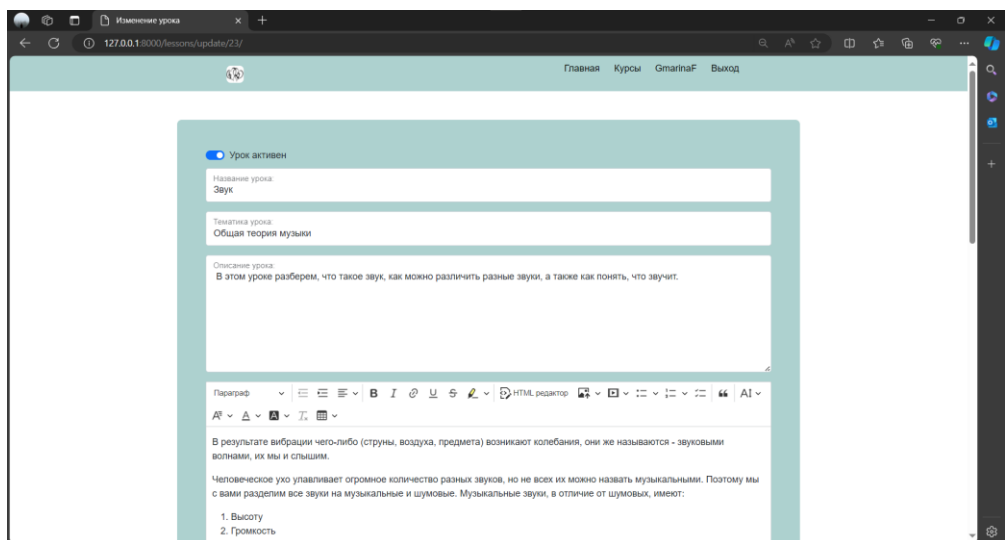


Рисунок 34 – Начало страницы редактирования урока

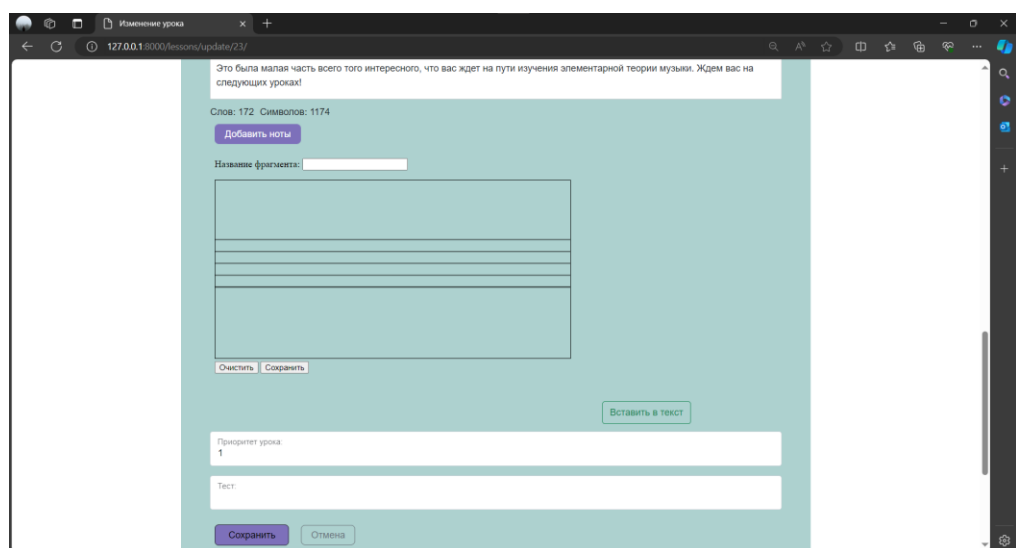


Рисунок 35 – Продолжение страницы редактирования урока

Работа приложения была протестирована на нескольких браузерах (Яндекс браузер, Google Chrome и Microsoft Edge), различий в работе выявлено не было.

9. Анализ полученных результатов

Реализованное приложение обладает всем необходимым функционалом, ниже представлены достоинства и недостатки приложения, а также идеи на перспективу развития проекта.

Достоинства приложения:

- доступность: все возможности и разделы видны с основных страниц, а поля на страницах редактирования имеют подсказки, что предотвращает большинство вопросов в начале пользования приложением;
- универсальность: каждый пользователь может одновременно быть и учителем, и учеником;
- музыкальность: у пользователя есть возможность добавить элементы, необходимые для музыкальной предметной области.

Недостатки приложения:

- базовый пользовательский интерфейс, что может отталкивать некоторых пользователей;
- отсутствие звуков в нотном редакторе, что ограничивает восприятие материала;
- при прохождении теста необходимо отправлять результаты после каждого вопроса, что через время может стать неудобно.

Описанные недостатки могут быть исправлены при дальнейшем развитии проекта. Также может быть реализован следующий дополнительный функционал:

- поиск курсов и уроков по строке запроса, что позволит пользователям находить курсы и уроки по конкретным интересующим темам;
- возможность открытия курса для ограниченного круга лиц, что позволит учителям не открывать доступ к курсу всем пользователям, а открыть его только для своих учеников;

— возможность добавления администратора, что позволит создателю курса передать некоторые обязанности по ведению курса другому пользователю.

ЗАКЛЮЧЕНИЕ

Перед началом работы было разработано техническое задание к проекту, далее были выбраны подходящие технологии для реализации, было спроектировано приложение, состоящие из базы данных, серверной и клиентской частей.

В процессе разработки была исследована музыкальная предметная область, на необходимом уровне были изучены фреймворк Django языка Python, языки разметки HTML и CSS, язык JavaScript, а также были выполнены все поставленные задачи.

Были реализованы редактор материала, нотный редактор, добавление и настройка курсов и уроков, тестов, а также динамическое добавление вариантов ответа на вопросы тестов, добавление отзывов и оценок пользователей на курсы, что может помочь другим пользователям выбрать наиболее подходящий и качественный курс.

С помощью реализованного web-приложения желающие смогут дистанционно и в интерактивном формате изучить музыкальную теорию, а учителя музыкальных школ смогут добавить интерактивный дистанционный формат обучения в дополнение к основному формату.

У пользователей имеется возможность создавать, настраивать и распространять курсы, добавлять к курсам уроки и настраивать их, добавлять материал к уроку, вводить нотный материал, создавать тестирование и прикреплять его к уроку, выбирать курсы по отзывам и оценкам, найти курс по коду, проходить курсы других пользователей, отслеживать свой прогресс при прохождении курсов.

В завершение был проведен анализ полученного результата, были выявлены достоинства и недостатки, на основании которых были сформулированы предложения по развитию проекта, например, добавление функциональных возможностей поиска курсов и уроков по строке запроса, предоставления доступа к курсу ограниченному кругу лиц, а также добавления администратора к курсу.

За время обучения были приобретены компетенции, описанные в таблице 1.

Таблица 1 – Приобретенные компетенции

Компетенция	Расшифровка компетенции	Описание приобретенных знаний, умений и навыков
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	В процессе работы было необходимо изучить большое количество информации с дальнейшим его анализом для получения качественного результата
УК-2	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	Из-за большого количества задач было необходимо грамотно распределить их выполнение и оптимально находить решение
УК-3	Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде	Были приобретены навыки работы и коммуникации в команде
УК-4	Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах)	Были приобретены навыки деловой коммуникации с людьми как в устной форме, так и в письменной на разных языках
УК-5	Способен воспринимать межкультурное разнообразие общества в социально-историческом, этическом и философском контекстах	Были приобретены компетенции восприятия межкультурного и межрегионального разнообразия общества в социально-историческом, этическом и философском контекстах
УК-6	Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни	Из-за ограниченного количества времени, было необходимо ранжировать задачи и распределить их по приоритету
УК-7	Способен поддерживать должный уровень физической подготовленности для обеспечения полноценной социальной и профессиональной деятельности	Были приобретены навыки физической подготовки и её поддержки для обеспечения полноценной социальной и профессиональной деятельности
УК-8	Способен создавать и поддерживать безопасные условия жизнедеятельности, в том числе при возникновении чрезвычайных ситуаций	Были приобретены навыки создания и поддержания безопасных условий жизнедеятельности, в том числе при возникновении чрезвычайных ситуаций
ОПК-1	Способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности	Были приобретены навыки применения фундаментальных знаний, полученных в области математических и естественных наук, а также их использования в профессиональной деятельности
ОПК-2	Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности	В процессе работы для реализации было необходимо выбрать технологии, а также программное обеспечение, поддерживающее эти технологии, где впоследствии реализовывалось приложение

Продолжение таблицы 1

Компетенция	Расшифровка компетенции	Описание приобретенных знаний, умений и навыков
ОПК-3	Способен к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям	В процессе работы были спроектированы и реализованы база данных, серверная и клиентская части
ОПК-4	Способен участвовать в разработке технической документации программных продуктов и комплексов с использованием стандартов, норм и правил, а также в управлении проектами создания информационных систем на стадиях жизненного цикла	В процессе проектирования было необходимо разработать техническую документацию, такую как техническое задание, описание функции и другое
ОПК-5	Способен устанавливать и сопровождать программное обеспечение информационных систем и баз данных, в том числе отечественного происхождения, с учетом информационной безопасности	Были приобретены навыки в установке программного обеспечения и его поддержке, в том числе баз данных как иностранного, так и отечественного производства, с учетом информационной безопасности
ПК-1	Преподавание по дополнительным общеобразовательным программам	Были приобретены компетенции преподавания по дополнительным общеобразовательным программам
ПК-2	Проверка работоспособности и рефакторинг кода программного обеспечения	Были приобретены компетенции проверки работоспособности и рефакторинга кода программного обеспечения
ПК-3	Интеграция программных модулей и компонент и верификация выпусков программного продукта	Были приобретены навыки интеграции программных модулей и компонентов и верификация выпусков программного продукта
ПК-4	Разработка требований и проектирование программного обеспечения	Перед началом реализации был проведен анализ будущего программного обеспечения с целью изначально эффективного проектирования
ПК-5	Оценка и выбор варианта архитектуры программного средства	Перед началом реализации было проведено проектирование будущего программного обеспечения
ПК-6	Разработка тестовых случаев, проведение тестирования и исследование результатов	В рамках работы был проведен эксперимент с последующим анализом результатов
ПК-7	Обеспечение функционирования баз данных	Были приобретены навыки в обеспечении функционирования баз данных
ПК-8	Оптимизация функционирования баз данных	Были приобретены навыки в оптимизации функционирования баз данных

Продолжение таблицы 1

Компетенция	Расшифровка компетенции	Описание приобретенных знаний, умений и навыков
ПК-9	Обеспечение информационной безопасности на уровне базы данных	Были приобретены компетенции обеспечения информационной безопасности на уровне базы данных
ПК-10	Выполнение работ по созданию (модификации) и сопровождению информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы	Были проведены работы по созданию (модификации) и сопровождению информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы
ПК-11	Создание и сопровождение требований и технических заданий на разработку и модернизацию систем и подсистем малого и среднего масштаба и сложности	Были приобретены компетенции создания и сопровождения требований и технических заданий на разработку и модернизацию систем и подсистем малого и среднего масштаба и сложности
ПК-12	Способен находить организационно-управленческие решения в нестандартных ситуациях и готов нести за них ответственность	Были приобретены навыки находить организационно-управленческие решения в нестандартных ситуациях и готов нести за них ответственность
ПК-13	Способен к коммуникации, восприятию информации, умению логически верно, аргументировано и ясно строить устную и письменную речь на русском языке для решения профессиональных задач	В процессе работы были проведены беседы с научным руководителем с целью научиться аргументированно и верно строить как письменное, так и словесное описание проекта
ПК-14	Способен использовать действующее законодательство и другие правовые документы в своей деятельности, демонстрировать готовность и стремление к совершенствованию и развитию общества на принципах гуманизма, свободы и демократии	Были приобретены компетенции использования действующего законодательства и других правовых документов в своей деятельности, демонстрации готовности и стремления к совершенствованию и развитию общества на принципах гуманизма, свободы и демократии

СПИСОК ЛИТЕРАТУРЫ

- 1) Что такое веб-тестирование? Типы тестирования веб приложений: сайт. – 2023. – URL: <https://habr.com/ru/articles/715376/> (дата обращения: 02.05.2024).
- 2) Django-documentation: сайт. – 2018. – URL: <https://docs.djangoproject.com/en/5.0/> (дата обращения: 12.02.2024).
- 3) Python: сайт. – 2017. – URL: <https://www.python.org/> (дата обращения: 12.02.2024).
- 4) Основы HTML: сайт. – 2015. – URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата обращения: 01.03.2024).
- 5) CSS documentation: сайт. – 2020. – URL: <https://devdocs.io/css/> (дата обращения: 02.03.2024).
- 6) JavaScript reference: сайт. – 2021. – URL: <https://devdocs.io/javascript/> (дата обращения: 16.03.2024).
- 7) MySQL Workbench: сайт. – 2020. – URL: <https://www.mysql.com/products/workbench/> (дата обращения: 15.02.2024).
- 8) The Django admin site: сайт. – 2023. – URL: <https://docs.djangoproject.com/en/5.0/ref/contrib/admin/> (дата обращения: 12.02.2024).
- 9) Django-ckeditor-5: сайт. – 2023. – URL: <https://pypi.org/project/django-ckeditor-5/> (дата обращения: 01.04.2024).
- 10) Гид по Фигме для начинающих веб-дизайнеров: сайт. – 2020. – URL: <https://tilda.education/articles-figma> (дата обращения: 16.02.2024).
- 11) HTML Canvas Reference: сайт. – 2020. – URL: https://www.w3schools.com/tags/ref_canvas.asp (дата обращения: 16.03.2024).

ПРИЛОЖЕНИЕ

Реализация нотного редактора. Файл notes_editor.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Canvas Нотный стан</title>
  <style>
    canvas {
      border: 1px solid black;
      margin-bottom: 20px;
      cursor: pointer;
    }
  </style>
  <script
src="/static/django_ckeditor_5/dist/bundle.js"></script>
  <script
src="/static/django_ckeditor_5/dist/translations/ru.js"></script
>
</head>
<body>
  <p>Название фрагмента: <input type="text"
id="notes_name"></p>
  <canvas id="staffCanvas"></canvas>
  <button onclick="clearCanvas()">Очистить</button>
  <button onclick="saveCanvas()">Сохранить</button>
  <script>
    const canvas = document.getElementById('staffCanvas');
    const ctx = canvas.getContext('2d');
    const gridSize = 10;
    let lastX = 0;
    let orderCounter = 1;
    const notes = [];
    canvas.width = gridSize * 60
    canvas.height = gridSize * 30

    // функция отрисовки нотного стана
    function drawLines () {
      const sbl = gridSize * 2; // space between lines
      const begin_x = 0;
      const begin_y = canvas.height / 2 - sbl * 2.5;
      const end_x = canvas.width;
      const end_y = canvas.height;

      ctx.beginPath();
      ctx.moveTo(begin_x, begin_y);
      ctx.lineTo(end_x, begin_y);
      ctx.moveTo(begin_x, begin_y + sbl);
      ctx.lineTo(end_x, begin_y + sbl);
```

```

        ctx.moveTo(begin_x, begin_y + 2 * sbl);
        ctx.lineTo(end_x, begin_y + 2 * sbl);
        ctx.moveTo(begin_x, begin_y + 3 * sbl);
        ctx.lineTo(end_x, begin_y + 3 * sbl);
        ctx.moveTo(begin_x, begin_y + 4 * sbl);
        ctx.lineTo(end_x, begin_y + 4 * sbl);
        ctx.closePath();
        ctx.stroke();
    }
    drawLines();

    // добавление ноты
    canvas.addEventListener('click', function(event) {
        const rect = canvas.getBoundingClientRect();
        const x = Math.round((event.clientX - rect.left) /
gridSize) * gridSize;
        const y = Math.round((event.clientY - rect.top) /
gridSize) * gridSize;

        if (notes.length === 0 && x === gridSize * 3) {
            const order = orderCounter++;
            const accidental = '';
            notes.push({ x, y, order, accidental });
            lastX = x;
            drawLines();
            renderNotes();
        }

        // Проверяем, что новая точка добавляется только в
        // правильном направлении по горизонтали
        if (x === lastX + gridSize * 3) {
            const order = orderCounter++;
            const accidental = '';
            notes.push({ x, y, order, accidental });
            console.log(notes)
            lastX = x;
            drawLines();
            renderNotes();
        }
    });

    // Функция для очистки холста
    function clearCanvas() {
        notes.length = 0;
        orderCounter = 1;
        lastX = 0;
        renderNotes();
    }

    // отслеживание мыши и прорисовка добавочных линий

```

```

        canvas.addEventListener('mousemove', function(event) {
            const rect = canvas.getBoundingClientRect();
            const mouseX = event.clientX - rect.left;
            const mouseY = event.clientY - rect.top;

            const lastNote = notes[notes.length - 1];
            const lastNoteX = lastNote.x + gridSize;

            // Если курсор мыши находится на позиции, следующей
за крайней нотой
            if ((mouseX > lastNoteX && mouseX < lastNoteX +
gridSize * 3)) {
                const additionalLinesCount = 10;
                const additionalLineWidth = gridSize * 1.5;
                const additionalLineSpacing = gridSize * 2;

                const sbl = gridSize * 2; // space between lines
                const begin_y = canvas.height / 2 - sbl * 2.5

                ctx.beginPath();
                for (let i = 0; i < additionalLinesCount / 2;
i++) {
                    const additionalLineXBegin = lastNoteX +
gridSize;
                    ctx.moveTo(additionalLineXBegin, begin_y +
(5 + i) * sbl);
                    ctx.lineTo(additionalLineXBegin +
additionalLineWidth, begin_y + (5 + i) * sbl);
                }
                for (let i = 0; i < additionalLinesCount / 2;
i++) {
                    const additionalLineXBegin = lastNoteX +
gridSize;
                    ctx.moveTo(additionalLineXBegin, begin_y -
(1 + i) * sbl);
                    ctx.lineTo(additionalLineXBegin +
additionalLineWidth, begin_y - (1 + i) * sbl);
                }
                ctx.strokeStyle = 'black';
                ctx.lineWidth = 1;
                ctx.stroke();
                ctx.closePath();
            }
            else {
                renderNotes()
            }
        });

        let selectedCircleIndex = -1;

        // выделение ноты
        canvas.addEventListener('click', function(event) {

```

```

const rect = canvas.getBoundingClientRect();
const mouseX = event.clientX - rect.left;
const mouseY = event.clientY - rect.top;

// Проверяем, находится ли курсор над каким-либо
кругом
for (let i = 0; i < notes.length; i++) {
  const circle = notes[i];
  const dx = mouseX - circle.x;
  const dy = mouseY - circle.y;
  const distance = Math.sqrt(dx * dx + dy * dy);

  if (distance < gridSize) {
    selectedCircleIndex = i;
    renderNotes();
    return;
  }
}

// Если клик был сделан вне круга, снимаем выделение
selectedCircleIndex = -1;
renderNotes();
});

// изменение нот (добавление и удаление бемолей и
диезов)
document.addEventListener('keydown', function(event) {
  const keyCode = event.keyCode;

  if (selectedCircleIndex !== -1) {
    const selectedNote = notes[selectedCircleIndex];

    if (keyCode === 38 || keyCode === 40) {
      if (selectedNote.accidental === '#' &&
keyCode === 38) {
        selectedNote.accidental = '';
        selectedNote.y -= gridSize;
      } else if (selectedNote.accidental === '#'
&& keyCode === 40) {
        selectedNote.accidental = '';
      } else if (selectedNote.accidental === 'b'
&& keyCode === 38) {
        selectedNote.accidental = '';
      } else if (selectedNote.accidental === 'b'
&& keyCode === 40) {
        selectedNote.accidental = '';
        selectedNote.y += gridSize;
      } else if (!selectedNote.accidental &&
keyCode === 38) {
        selectedNote.accidental = '#';
      }
    }
  }
});

```



```

        } else if (!selectedNote.accidental &&
keyCode === 40) {
            selectedNote.accidental = 'b';
        }

        renderNotes();
    }
});

// отрисовка последовательности нот
function renderNotes() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.lineWidth = 1;
    drawLines();

    notes.forEach((note, index) => {
        ctx.beginPath();
        ctx.arc(note.x, note.y, gridSize, 0, Math.PI *
2);

        ctx.fillStyle = 'black';

        if (index === selectedCircleIndex) {
            ctx.fillStyle = 'blue';
        }

        ctx.fill();
        ctx.font = '10px Arial';
        ctx.textAlign = 'center';
        ctx.beginPath();
        ctx.lineWidth = 3;
        ctx.moveTo(note.x + gridSize * 0.8, note.y);
        ctx.lineTo(note.x + gridSize * 0.8, note.y -
50);

        ctx.closePath();
        ctx.stroke();
        if (note.accidental) {
            ctx.font = '20px Arial';
            ctx.fillText(note.accidental, note.x - 15,
note.y + 5);
        }

        ctx.font = '10px Arial';
        ctx.fillText(note.order, note.x, note.y + 3);
        ctx.closePath();
    });
}
var saved=-1;

// сохранение холста и передача данных
function saveCanvas() {

```

```

        const testt = 'test data';
        const formData = new FormData();
        const jsonData = JSON.stringify(notes);
        formData.append('testt', testt);
        formData.append('notes', jsonData);
        formData.append('name',
document.getElementById('notes_name').value)
        fetch('/tests/noteseeditor/', {
            method: 'POST',
            headers: {'X-CSRFToken': "{{ csrf_token }}"},
            body: formData,
        })
            .then(async response => {
                let j = await response.json()
                console.log('Данные успешно отправлены', j);
                saved = j
            })
            .catch(error => {
                console.error('Произошла ошибка:', error);
            });
    }

    // добавление фрагмента в текст
    parent.document.getElementById('notes_insert').onclick =
function(e) {
    const domEditableElement =
parent.document.querySelector( '.ck-editor__editable' );
    const editorInstance =
domEditableElement.ckeditorInstance;
    const viewFragment =
editorInstance.data.processor.toView( `<iframe
src="/tests/notes_lesson/${saved['id']}" width="650px"
height="400px">` );
    const modelFragment = editorInstance.data.toModel(
viewFragment );
    editorInstance.model.insertContent(modelFragment);
}

</script>
</body>
</html>

```

Реализация нотного редактора на режим чтения. Файл
notes_readonly.html.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

```

```

<title>Canvas Нотный стан</title>
<style>
    canvas {
        border: 1px solid black;
        margin-bottom: 20px;
        cursor: pointer;
    }
</style>
</head>
<body>
    <canvas id="staffCanvas"></canvas>
    <script>
        {% autoescape off %}
        let srcdata = {{ answer }}
        {% endautoescape %}
        console.log(srcdata);
        const canvas = document.getElementById('staffCanvas');
        const ctx = canvas.getContext('2d');
        const gridSize = 10;
        let lastX = 0;
        let orderCounter = 1;
        {% if asstring %}
            const notes = JSON.parse(srcdata['notes']);
        {% else %}
            const notes = srcdata;
        {% endif %}
        canvas.width = gridSize * 60
        canvas.height = gridSize * 30

        // отрисовка нотного стана
        function drawLines () {
            const sbl = gridSize * 2; // space between lines
            const begin_x = 0;
            const begin_y = canvas.height / 2 - sbl * 2.5;
            const end_x = canvas.width;
            const end_y = canvas.height;

            ctx.beginPath();
            ctx.moveTo(begin_x, begin_y);
            ctx.lineTo(end_x, begin_y);
            ctx.moveTo(begin_x, begin_y + sbl);
            ctx.lineTo(end_x, begin_y + sbl);
            ctx.moveTo(begin_x, begin_y + 2 * sbl);
            ctx.lineTo(end_x, begin_y + 2 * sbl);
            ctx.moveTo(begin_x, begin_y + 3 * sbl);
            ctx.lineTo(end_x, begin_y + 3 * sbl);
            ctx.moveTo(begin_x, begin_y + 4 * sbl);
            ctx.lineTo(end_x, begin_y + 4 * sbl);
            ctx.closePath();
            ctx.stroke();
        }
        drawLines();
    </script>

```

```

// отрисовка последовательности нот
function renderNotes() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawLines();
    console.log(notes)

    notes.forEach((note, index) => {
        ctx.beginPath();
        ctx.arc(note.x, note.y, gridSize, 0, Math.PI *
2);

        ctx.fillStyle = 'black';

        ctx.fill();
        ctx.font = '10px Arial';
        ctx.textAlign = 'center';

        ctx.beginPath();
        ctx.lineWidth = 3;
        ctx.moveTo(note.x + gridSize * 0.8, note.y);
        ctx.lineTo(note.x + gridSize * 0.8, note.y -
50);

        ctx.closePath();
        ctx.stroke();

        if (note.accidental) {
            ctx.font = '20px Arial';
            ctx.fillText(note.accidental, note.x - 15,
note.y + 5);
        }

        ctx.font = '10px Arial';
        ctx.fillText(note.order, note.x, note.y + 3);
        ctx.closePath();
    });
}
renderNotes();

</script>
</body>
</html>

```

Страница с общими элементами. Файл htmltemplate.html.

```

{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>{% block title %}{% endblock %}</title>
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.0.2/dist/css/boot

```

```

strap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuaCOMLASjC
" crossorigin="anonymous">
    <link rel="stylesheet" href="{% static
'main\css\htmltemplatestyles.css' %}">
    <link rel="stylesheet" href="{% static 'main\css\custom.css'
%}">
</head>
<body class="d-flex flex-column min-vh-100">
    <div class="backcontainer" style="width: 100%; height: 60px;
background-color: #ADD1CF">
        <!-- навигационное меню -->
        <header class="container justify-content-between align-
items-center">
            <div class="logo"></div>
            <nav>
                <ul>
                    <li><a href="{% url 'commonindex'
%}">Главная</a></li>
                    {% if user.is_authenticated %}
                    <li><a href="{% url 'my_courses'
%}">Курсы</a></li>
                    <li><a href="{% url 'myprofile' %}">{{
user.username }}</a></li>
                    <li><a href="{% url 'logout'
%}">Выход</a></li>
                    {% else %}
                    <li><a href="{% url 'login'
%}">Войти</a></li>
                    {% endif %}
                </ul>
            </nav>
        </header>
    </div>

    <script
src="https://cdn.jsdelivrivr.net/npm/@popperjs/core@2.11.8/dist/umd
/popper.min.js" integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r
" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.3/dist/js/bootst
rap.min.js" integrity="sha384-
0pUGZvbk66XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy
" crossorigin="anonymous"></script>
    {% block maincontent %}
    <!-- блок для наполнения -->
    {% endblock %}
    <div class="mt-50px">
        <p></p>

```

```

    </div>
    <!-- нижняя часть сайта -->
    <footer class="container-fluid justify-content-center mt-
auto">
        <div class="row justify-content-around mb-0 pt-5 pb-0 ">
            <div class=" col-11">
                <div class="row justify-content-center">
                    <div class="col-md-3 col-12 font-italic
align-items-center mt-md-3 mt-4"><h5>Музыкальные курсы</h5>
                        <small>Казань, Кремлевская 35</small>
                    </div>
                    <div class="col-md-3 col-12 my-sm-0 mt-5">
                        <ul class="list-unstyled">
                            <li class="mt-md-3 mt-4">Наши
контакты</li>
                                <li>+7 999 888 77 66</li>
                            </ul>
                        </div>
                        <div class="col-md-3 col-12 my-sm-0 mt-5">
                            <ul class="list-unstyled">
                                <li class="mt-md-3 mt-4"><h5>Музыка
– это самый сильный вид магии.</h5></li>
                                    <li>Мэрилин Мэнсон - американский
рок-певец, поэт-песенник, художник и бывший музыкальный
журналист</li>
                                        <li><small>1969</small></li>
                                    </ul>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </body>
</html>

```

Представления подсистемы курсов. Файл `course\views.py`.

```

from django.shortcuts import render, redirect
from django.views import View
from . import forms
from .models import Courses, CourseProgress, CourseFeedback
from lessons.models import Lesson

# представление отображения созданных и проходимых курсов
class MyCoursesView(View):
    def get(self, request):
        user = request.user
        my_created_courses =
Courses.objects.filter(course_creator=user)
        my_passing_courses =
Courses.objects.filter(course_students__in=[user])

```

```

        progresses =
[CourseProgress.objects.get_or_create(user=request.user,
course=c)[0] for c in my_passing_courses]
        max_priority = [max([l.lesson_priority for l in
Lesson.objects.filter(lesson_course=c)]) for c in
                        my_passing_courses]
        context = {'my_created_courses': my_created_courses,
'my_passing_courses': my_passing_courses,
                    'progresses': progresses,
                    'zip': zip(my_passing_courses, progresses,
max_priority)}
        template_name = 'main/my_courses.html'
        return render(request, template_name, context)

# представление отображения курса
class CourseView(View):
    def get(self, request, pk):
        course = Courses.objects.get(id=pk)
        lessons =
Lesson.objects.filter(lesson_course=course).order_by('lesson_pri
ority')
        progress =
CourseProgress.objects.get_or_create(user=request.user,
course=course)[0]
        feedbacks = CourseFeedback.objects.filter(course=course)
        max_priority = max([lesson.lesson_priority for lesson in
lessons])
        context = {
            'course': course,
            'lessons': lessons,
            'progress': progress,
            'max_priority': max_priority,
            'btn': 'my',
            'feedbacks': feedbacks,
        }
        if
course.course_students.filter(id=request.user.id).exists():
            context['btn'] = 'passing'
        elif course.course_creator == request.user:
            context['btn'] = 'created'
        else:
            context['btn'] = 'signup'
        template_name = 'main/current_course.html'
        form = forms.AddFeedbackForm()
        context['form'] = form
        return render(request, template_name, context)

    def post(self, request, pk):
        course = Courses.objects.get(id=pk)
        form = forms.AddFeedbackForm(request.POST)
        if form.is_valid():

```

```

        feedback = form.save(commit=False)
        feedback.course = course
        feedback.user = request.user
        feedback.save()
        return redirect('current_course', pk=pk)

# представления создания курса
class AddCourseView(View):
    def get(self, request):
        form = forms.AddCourseForm()
        return render(request, 'main/add_course.html',
                        {'form': form, 'action': 'Добавление',
                        'button_text': "Добавить"})

    def post(self, request):
        form = forms.AddCourseForm(request.POST, request.FILES)
        if form.is_valid():
            course = form.save(commit=False)
            course.course_creator = request.user
            course.save()
            Lesson.objects.create(lesson_course=course,
lesson_priority=0, lesson_name="Знакомство")
            return redirect('my_courses')
        return render(request, 'main/my_courses.html',
                        {'form': form, 'action': 'Добавление',
                        'button_text': "Добавить"})

# представление удаления курса
class DeleteCourseView(View):
    def get(self, request, pk):
        if request.user !=
Courses.objects.get(id=pk).course_creator:
            return redirect('my_courses')
        course = Courses.objects.get(id=pk)
        course.delete()
        return redirect('my_courses')

# представление изменения курса
class UpdateCourseView(View):
    def get(self, request, pk):
        if request.user !=
Courses.objects.get(id=pk).course_creator:
            return redirect('my_courses')
        course = Courses.objects.get(id=pk)
        form = forms.AddCourseForm(instance=course)
        return render(request, 'main/add_course.html',
                        {'form': form, 'action': 'Изменение',
                        'course': course, 'button_text': "Сохранить"})

```



```

    def post(self, request, pk):
        if request.user !=
Courses.objects.get(id=pk).course_creator:
            return redirect('my_courses')
        course = Courses.objects.get(id=pk)
        form = forms.AddCourseForm(request.POST, request.FILES,
instance=course)
        if form.is_valid():
            form.save()
            return redirect('my_courses')
        return render(request, 'main/add_course.html',
                        {'form': form, 'action': 'Изменение',
'course': course, 'button_text': "Сохранить"})

# представление записи на курс
class CourseSignUpView(View):
    def get(self, request, pk):
        course = Courses.objects.get(id=pk)
        course.course_students.add(request.user)
        return redirect('current_course', pk=course.id)

# представление ухода с курса
class CourseSignOutView(View):
    def get(self, request, pk):
        course = Courses.objects.get(id=pk)
        course.course_students.remove(request.user)
        return redirect('current_course', pk=course.id)

# представление оценки курса
class RateCourseView(View):
    def get(self, request, pk):
        course = Courses.objects.get(id=pk)
        return render(request, 'main/rate_course.html')

```

Представления подсистемы уроков. Файл lessons\views.py.

```

from django.shortcuts import render, redirect
from django.views import View
from . import forms
from .models import Lesson
from course.models import Courses, CourseProgress
from tests.models import Test, TestAnswer, TestQuestion
from tests.forms import OneAnswerTest, ManyAnswerTest,
FileAnswerTest, NoteAnswerForm, OpenAnswerTest
from musicserver.settings import MEDIA_URL

# представление создания урока
class AddLessonView(View):
    def get(self, request, pk):
        course = Courses.objects.get(id=pk)
        form = forms.AddLessonForm()

```

```

        tests = Test.objects.filter(test_creator=request.user)
        return render(request, 'main/add_lesson.html', {'form':
form, 'action': 'Добавление',

'button_text': "Добавить", 'tests': tests, 'course': course})

    def post(self, request, pk):
        form = forms.AddLessonForm(request.POST, request.FILES)
        course = Courses.objects.get(id=pk)
        if form.is_valid():
            lesson = form.save(commit=False)
            lesson.lesson_course = course
            lesson.lesson_creator = request.user
            lesson.save()
            return redirect('current_course', pk=course.id)
        return render(request, 'main/add_lesson.html', {'form':
form, 'action': 'Добавление',

'button_text': "Добавить", 'course': course})

# представление просмотра урока
class LessonView(View):
    def get(self, request, pk):
        lesson = Lesson.objects.get(id=pk)
        try:
            test = lesson.test
            test_forms = []
            answers_render = []
            answers =
TestAnswer.objects.filter(user=request.user,
question__test=test)
            answered_questions =
TestQuestion.objects.filter(test=test,
id__in=answers.values('question'))
            remained_questions =
test.questions.all().difference(answered_questions)
            print(answered_questions, remained_questions)
            for question in remained_questions:
                if question.type == 'one':
                    f = OneAnswerTest()
                    f.fields['question'].initial = question.id
                    f.fields['answer'].widget.choices =
[(answer.answer, answer.answer) for answer in
question.answers.all()]
                    test_forms.append(f)
                elif question.type == 'many':
                    f = ManyAnswerTest()
                    f.fields['question'].initial = question.id
                    f.fields['answer'].widget.choices =
[(answer.answer, answer.answer) for answer in
question.answers.all()]

```

```

        test_forms.append(f)
    elif question.type == 'open':
        f = OpenAnswerTest()
        f.fields['question'].initial = question.id
        test_forms.append(f)
    elif question.type == 'file':
        f = FileAnswerTest()
        f.fields['question'].initial = question.id
        test_forms.append(f)
    elif question.type == 'note':
        f = NoteAnswerForm()
        f.fields['question'].initial = question.id
        test_forms.append(f)
    if len(remained_questions) == 0:
        for answer in answers:
            answers_render.append([
                answer.answer.strip('[]').replace('"',
'') if answer.question.type != 'file' else f'<a
href="{MEDIA_URL}{answer.media}">ссылка</a>',
                answer.question.question, ", ".join([a
for a in
answer.question.correct_answer.all().values_list('answer',
flat=True)]),
                answer.score,
                answer.question.score,
                answer.comment])
        return render(request, 'main/lesson.html',
                        {'lesson': lesson, 'test_forms':
test_forms, 'test': zip(test_forms, remained_questions),
'answers_render': answers_render})
    except AttributeError:
        return render(request, 'main/lesson.html',
                        {'lesson': lesson, 'test_forms': [],
'test': zip([], [])})

    def post(self, request, pk):
        lesson = Lesson.objects.get(id=pk)
        progress =
CourseProgress.objects.get(course=lesson.lesson_course,
user=request.user)
        progress.lesson = lesson.lesson_priority
        progress.save()
        return redirect('current_course',
pk=lesson.lesson_course.id)

# представление изменения урока
class UpdateLessonView(View):
    def get(self, request, pk):
        lesson = Lesson.objects.get(id=pk)
        form = forms.AddLessonForm(instance=lesson)
        tests = Test.objects.filter(test_creator=request.user)

```

```

        return render(request, 'main/add_lesson.html', {'form':
form, 'action': 'Изменение',

'button_text': "Сохранить", 'tests': tests, 'course':
lesson.lesson_course, 'lesson': lesson})

    def post(self, request, pk):
        lesson = Lesson.objects.get(id=pk)
        form = forms.AddLessonForm(request.POST, request.FILES,
instance=lesson)
        if form.is_valid():
            form.save()
            return redirect('current_course',
pk=lesson.lesson_course.id)
        return render(request, 'main/add_lesson.html', {'form':
form, 'action': 'Изменение',

'button_text': "Сохранить", 'course': lesson.lesson_course,
'lesson': lesson})

# представление удаления урока
class DeleteLessonView(View):
    def get(self, request, pk):
        lesson = Lesson.objects.get(id=pk)
        lesson.delete()
        return redirect('current_course',
pk=lesson.lesson_course.id)

```

Представления подсистемы проверки знаний. Файл tests\views.py.

```

from django.shortcuts import render
from django.views import View
from .models import Test, NoteSet, TestQuestion, TestAnswer,
TestAnswerVariant
from . import forms
from django.http import JsonResponse
from django.shortcuts import redirect
import ast
from lessons.models import Lesson

# представление нотного редактора
class NotesEditor(View):
    def get(self, request):
        return render(request, 'main/notes_editor.html')

    def post(self, request):
        note = NoteSet()
        note.notes = request.POST.get('notes')
        note.name = request.POST.get('name')
        print(note.notes)
        note.save()

```

```

        return JsonResponse({'id': note.id})

# представление создания теста
class AddTestView(View):
    def get(self, request):
        form = forms.AddTestForm()
        return render(request, 'main/add_test.html', {'form':
form})

    def post(self, request):
        form = forms.AddTestForm(request.POST)
        if form.is_valid():
            test = form.save(commit=False)
            test.test_creator = request.user
            test.save()
            return redirect('view_test', pk=form.instance.id)
        return render(request, 'main/add_test.html', {'form':
form})

# представление изменения теста
class EditTestView(View):
    def get(self, request, pk):
        test = Test.objects.get(pk=pk)
        form = forms.AddTestForm(instance=test)
        return render(request, 'test_form.html', {'form': form})

    def post(self, request, pk):
        test = Test.objects.get(pk=pk)
        form = forms.AddTestForm(request.POST, instance=test)
        if form.is_valid():
            form.save()
            return redirect('view_lesson',
pk=test.lesson_set.first().id)
        return render(request, 'test_form.html', {'form': form})

# представление обработчика тестирования
class TestProcessor(View):
    def post(self, request, *args, **kwargs):
        # обработчик тестирования для вопросов с одним
        # правильным вариантом ответа
        if kwargs['type'] == 'one':
            f = forms.OneAnswerTest(request.POST)
            if f.is_valid():
                print(f.cleaned_data)
                question =
TestQuestion.objects.get(id=f.cleaned_data['question'])
                answer = TestAnswer()
                answer.question = question
                answer.user = request.user

```

```

        answer.answer = f.cleaned_data['answer']
        answer.checked = True
        if f.cleaned_data['answer'] ==
question.correct_answer.first().answer:
            answer.score = question.score
            answer.save()
            return redirect('view_lesson',
pk=kwargs['lesson'])
        else:
            print(f.errors)
            return redirect('view_lesson',
pk=kwargs['lesson'])
        # обработчик тестирования для вопросов с несколькими
        правильными вариантами ответа
        elif kwargs['type'] == 'many':
            f = forms.ManyAnswerTest(request.POST)
            if f.is_valid():
                question =
TestQuestion.objects.get(id=f.cleaned_data['question'])
                answer = TestAnswer()
                answer.question = question
                answer.user = request.user
                answer.answer = f.cleaned_data['answer']
                answer.checked = True
                form_answers =
ast.literal_eval(f.cleaned_data['answer'])
                question_answers =
question.correct_answer.all().values_list('answer', flat=True)
                if
len(set(form_answers).intersection(question_answers)) ==
len(question_answers):
                    answer.score = question.score
                    answer.save()
                    return redirect('view_lesson',
pk=kwargs['lesson'])
                else:
                    print(f.errors)
                    return redirect('view_lesson',
pk=kwargs['lesson'])
        # обработчик тестирования для вопросов с открытым
        ответом
        elif kwargs['type'] == 'open':
            f = forms.OpenAnswerTest(request.POST)
            if f.is_valid():
                question =
TestQuestion.objects.get(id=f.cleaned_data['question'])
                answer = TestAnswer()
                answer.question = question
                answer.user = request.user
                answer.answer = f.cleaned_data['answer']
                answer.save()

```

```

        return redirect('view_lesson',
pk=kwargs['lesson'])
    else:
        print(f.errors)
        return redirect('view_lesson',
pk=kwargs['lesson'])
    # обработчик тестирования для вопросов, в ответ на
    # которые требуется прикрепить файл
    elif kwargs['type'] == 'file':
        f = forms.FileAnswerTest(request.POST,
request.FILES)
        if f.is_valid():
            question =
TestQuestion.objects.get(id=f.cleaned_data['question'])
            answer = TestAnswer()
            answer.question = question
            answer.user = request.user
            answer.media = f.cleaned_data['answer']
            answer.answer = f.cleaned_data['comment']
            answer.save()
            return redirect('view_lesson',
pk=kwargs['lesson'])
        else:
            print(f.errors)
            return redirect('view_lesson',
pk=kwargs['lesson'])
    # обработчик тестирования для вопросов, ответ на которые
    # заключаются в нотах
    elif kwargs['type'] == 'note':
        f = forms.NoteAnswerForm(request.POST)
        if f.is_valid():
            question =
TestQuestion.objects.get(id=f.cleaned_data['question'])
            answer = TestAnswer()
            answer.question = question
            answer.user = request.user
            answer.answer = f.cleaned_data['answer']
            answer.comment = f.cleaned_data['comment']
            answer.save()
            return redirect('view_lesson',
pk=kwargs['lesson'])
        else:
            print(f.errors)
            return redirect('view_lesson',
pk=kwargs['lesson'])

# представление отображения списка тестирований пользователя
# (прогресса)
class UserTests(View):
    def get(self, request, *args, **kwargs):
        tests = TestAnswer.objects.filter(user=request.user)

```

```

user_tests = []
for test in tests:
    q = test.question
    t = Test.objects.get(questions__in=[q])
    print(t.lesson_set.first().lesson_name)
    user_tests.append(t)
user_tests = list(set(user_tests))
checked = []
res = []
maxs = []
for test in user_tests:
    value = True
    score = 0
    maxsc = 0
    for question in test.questions.all():
        answer =
TestAnswer.objects.get(user=request.user, question=question)
        maxsc += question.score
        if not answer.checked:
            value = False
        else:
            score += answer.score
        checked.append(value)
        res.append(score)
        maxs.append(maxsc)
    return render(request, 'main/all_tests.html',
                  {'tests': zip(user_tests, checked, res,
maxs), 'user_tests': user_tests})

# представление просмотра теста
class TestView(View):
    def get(self, request, pk):
        test = Test.objects.get(pk=pk)
        for q in test.questions.all():
            for a in
TestAnswerVariant.objects.filter(question=q):
                if a not in q.answers.all():
                    a.delete()
            form = forms.AddTestQuestionForm()
            return render(request, 'main/test.html', {'test': test,
'form': form})

    def post(self, request, pk):
        test = Test.objects.get(pk=pk)
        form = forms.AddTestQuestionForm(request.POST)
        if form.is_valid():
            question = form.save()
            for var in
TestAnswerVariant.objects.filter(question_id=100015):
                var.question = question
                var.save()

```



```

        question.answers.add(var)
        test.questions.add(question)
        test.save()
        return redirect('view_test', pk=pk)
    return render(request, 'main/test.html', {'test': test,
'form': form})

# представление удаления вопроса теста
class DeleteQuestion(View):
    def get(self, request, pk):
        question = TestQuestion.objects.get(pk=pk)
        test = question.test_set.first().id
        question.delete()
        return redirect('view_test', pk=test)

# представление изменения вопроса теста
class UpdateQuestion(View):
    def get(self, request, pk):
        question = TestQuestion.objects.get(pk=pk)
        test = question.test_set.first()
        form = forms.AddTestQuestionForm(instance=question)
        return render(request, 'main/test.html', {'form': form,
'question': question, 'test': test, 'expanded': True})

    def post(self, request, pk):
        question = TestQuestion.objects.get(pk=pk)
        test = question.test_set.first()
        form = forms.AddTestQuestionForm(request.POST,
instance=question)
        if form.is_valid():
            question = form.save()
            for var in
TestAnswerVariant.objects.filter(question_id=question.id):
                var.question = question
                var.save()
                question.answers.add(var)
            test.questions.add(question)
            test.save()
            return redirect('view_test',
pk=question.test_set.first().id)
        return render(request, 'main/test.html', {'form': form,
'question': question, 'test': test, 'expanded': True})

# представление отображения списка ответов учеников на проверку
преподавателю
class TestChecker(View):
    def get(self, request):
        lessons =
Lesson.objects.filter(lesson_creator=request.user)

```

```

        tests = [l.test for l in lessons if l.test is not None]
        answers_final = []
        for t in tests:
            for q in t.questions.all():
                answers = TestAnswer.objects.filter(question=q,
checked=False)
                print(tests, t, answers)
                if len(answers) > 0:
                    for a in answers:
                        answers_final.append(a)
            form = forms.CheckAnswerForm()
            return render(request, 'main/test_checker.html',
{'answers': list(set(answers_final)), 'form': form})

# представление обработки проверки ответа ученика преподавателем
class TestCheckSaver(View):
    def post(self, request, pk):
        form = forms.CheckAnswerForm(request.POST)
        if form.is_valid():
            answer = TestAnswer.objects.get(pk=pk)
            answer.checked = True
            answer.score = form.cleaned_data['score']
            answer.comment = form.cleaned_data['comment']
            answer.save()
            return redirect('test_checker')

# представление нотного редактора на режим чтения
class NotesReadonlyView(View):
    def get(self, request, pk):
        ans = TestAnswer.objects.get(pk=pk)
        return render(request, 'main/notes_readonly.html',
{'answer': ans.answer, 'asstring': True})

# представление нотного редактора на режим чтения в уроке
class NotesReadonlyLessonView(View):
    def get(self, request, pk):
        note = NoteSet.objects.get(pk=pk)
        return render(request, 'main/notes_readonly.html',
{'answer': note.notes, 'asstring': False})

# представление добавления варианта ответа на вопрос теста
class AddTestVariant(View):
    def post(self, request, pk):
        question = TestQuestion.objects.get(pk=pk)
        form = forms.AddTestAnswerVariantForm(request.POST)
        correct = question.correct_answer.all()
        correct = [c.id for c in correct]
        if form.is_valid():

```

```

        if form.cleaned_data['answer'] is not None:
            v = form.save(commit=False)
            v.question = question
            v.save()
            vars = [(v.id, v.answer) for v in
TestAnswerVariant.objects.filter(question=question)]
            return JsonResponse({'status': 'ok', 'variants':
vars, 'correct': correct})
            vars = [(v.id, v.answer) for v in
TestAnswerVariant.objects.filter(question=question)]
            return JsonResponse({'status': 'error', 'variants':
vars, 'correct': correct})

# представление теста с нотами в качестве ответа
class NotesTest(View):
    def get(self, request):
        return render(request, 'main/notes_answer.html')

# представление очистки вариантов ответа
class DeleteAnswerVariant(View):
    def post(self, request):
        variant =
TestAnswerVariant.objects.filter(question_id=100015)
        print(variant)
        for v in variant:
            v.delete()
        return JsonResponse({'status': 'ok'})

# представление отображения списка всех созданных тестов
class AllCreatedTestsView(View):
    def get(self, request):
        tests = Test.objects.filter(test_creator=request.user)
        return render(request, 'main/all_created_tests.html',
{'tests': tests})

```