СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 4

ВВЕДЕНИЕ 5

1 ПОСТАНОВКА ЗАДАЧИ 6

1.1 Описание предметной области 6

1.2 Цели и задачи на проектирование 6

2 ПРОЕКТИРОВАНИЕ 8

2.1 Проектирование схемы базы данных 8

2.2 Обоснование выбора технологии 11

2.3 Описание среды реализации 13

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 15

3.1 Физическая структура базы данных 15

3.2 Структура информационной системы 19

3.3 Тестирование системы 20

3.4 Описание применения 23

ЗАКЛЮЧЕНИЕ 29

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 30

ПРИЛОЖЕНИЕ А 32

ПРИЛОЖЕНИЕ Б 33

ПРИЛОЖЕНИЕ В 34

ПРИЛОЖЕНИЕ Г 35

ПРИЛОЖЕНИЕ Д 36

ПРИЛОЖЕНИЕ Е 37

ПРИЛОЖЕНИЕ Ж 38

ВВЕДЕНИЕ

Образовательный курс – серия учебных занятий, идущая по плану и расписанию, нацеленная на обучение одному учебному предмету – началам какой-либо конкретной науки или освоению определённого вида человеческой деятельности (например, языка, физкультуры, рисования). [1]

В данном курсовом проекте будет разработано веб-приложение образовательных курсов.

В современном мире образовательные курсы являются неотъемлемой частью обучения. Существуют различные виды проведения занятий при обучении.

Образовательные курсы аудиторного типа неудобны своим жестким графиком и привязкой к определенному месту. Эти минусы отсутствуют в онлайн-занятиях.

Большинство людей, которые ищут образовательные курсы по определенным дисциплинам, пользуются специализированными веб-приложениями, на что тратится большое количество времени.

Веб-приложение с множеством дисциплин поможет людям уменьшить время на поиск необходимого курса и не привязываться к графику и месту проведения.

# ПОСТАНОВКА ЗАДАЧИ

## 1.1 Описание предметной области

В сети интернет существуют множество разнообразных курсов по различным дисциплинам. Эти курсы можно разместить на одной веб-площадке, а саму информацию о курсах хранить в базе данных. Это обеспечит удобство для пользователя, надежность хранения и быстрый доступ к обучающей информации.

Рассматриваемая предметная область представляет собой веб-приложение, направленное на хранение и предоставление курсов для интернет-пользователей.

В большинстве подобных веб-приложениях слишком перегруженный интерфейс или отсутствует возможность выбора различных дисциплин. Поэтому разрабатываемое веб-приложение направлено на скорость работы и многогранность образовательных курсов.

Одним из существующих аналогов является веб-ресурс «Stepik». [2]

Плюсом данного веб-приложения является наличие видеосвязи с ментором.

Огромными недостатками являются сильная перегруженность веб-ресурса, а также малое количество дисциплин, по которым возможно обучение.

## 1.2 Цели и задачи на проектирование

Цель данного курсового проекта разработать «Веб-приложение образовательных курсов». Основными задачами, стоящими перед проектируемой программой, являются:

1 Описать и проанализировать предметную область.

2 Разработать базу данных приложения и освоить средства поддержки баз данных.

3 Создать и отладить программное средство для взаимодействия с базой данных.

4 Выбрать и обосновать технологию и среду реализации проекта.

5 Провести тестирование системы.

6 Сформировать умения использовать справочную литературу.

7 Приобрести навыки по оформлению текстовой и графической документации согласно требованиям государственных стандартов и стандарта предприятия. [3]

Для создания веб-приложения образовательных курсов необходимо проанализировать, какие возможности необходимы пользователям данной системы.

Для администратора:

− авторизация;

− редактирование данных пользователей;

− удаление пользователей;

− добавление, редактирование и удаление курса;

− просмотр информации о курсе;

− оценка курса;

− выход из системы.

Для клиента:

− регистрация в системе;

− авторизация;

− просмотр информации о курсе;

− выход из системы.

Разрабатываемая программа должна иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено.

При конструировании приложения необходимо предусмотреть возможность защиты данных от изменения, установить ограничения на некорректный ввод данных.

# ПРОЕКТИРОВАНИЕ

## 2.1 Проектирование схемы базы данных

В процессе проектирования схемы данных разработаем диаграмму вариантов использования пользователя и администратора.

Для администратора выделим следующие варианты использования:

– управлять информацией о курсе:

– добавить курс;

– редактировать данные курса;

– удалить курс;

– управлять информацией о лекциях:

– добавить лекцию;

– редактировать данные лекции;

– удалить лекцию;

– управлять информацией об уроках:

– добавить урок;

– редактировать данные уроков;

– удалить урок;

– управлять информацией о преподавателях:

– добавить преподавателя;

– редактировать данные преподавателя;

– удалить преподавателя;

– управлять информацией об учебных предметах:

– добавить предмет;

– редактировать данные предмета;

– удалить предмет.

– управлять информацией о пользователях:

– добавить администратора;

– редактировать логин пользователя;

– удалить пользователя.

Для пользователя выделим следующие варианты использования:

– авторизация в системе;

– регистрация в системе;

– просмотр краткой информации о курсах;

– просмотр полной информации о курсах.

Диаграмма вариантов использования представлена в приложении Д.

Для пользователя и администратора разработан алгоритм их действий при работе с системой. Попадая на домашнюю страницу системы, пользователь выбирает нужное ему действие: войти в систему или зарегистрироваться в системе, просмотреть краткую информацию о выбранном курсе. Для авторизации в системе у пользователя должен быть логин и пароль для входа в систему. Если логина и пароля нет, пользователь считается анонимным пользователем и не может просматривать полную информацию о курсе. После входа в систему пользователь может просматривать полную информацию о курсах.

Администратор имеет отдельное окно авторизации. Администратора может зарегистрировать только другой администратор. Авторизованный администратор может управлять доступом к кабинету администратора, изменяя уровень доступа к системе у пользователя и его роль на роль администратора и наоборот. Новый администратор не может управлять доступом существующих администраторов и пользователей. Также, при наличии прав, администратор может добавить, удалить или изменить любую информацию о курсах, лекциях, уроках, преподавателях, предметах.

Алгоритм работы пользователя представлен в приложении В. Алгоритм работы администратора представлен в приложении Г.

Также, в процессе проектирования системы разработаны диаграммы взаимодействия. Диаграммы взаимодействия для вариантов использования «Пользователь» и «Администратор» представлены в приложении Д.

В процессе проектирования были изучены виды архитектур информационных систем, представленных в виде веб-приложений, такие как:

− монолит (Monolithic Architecture);

− микросервисная архитектура;

− серверлесс архитектура (Serverless Architecture). [4]

Монолит – это архитектурное решение, в котором все компоненты и модули тесно связаны между собой и зависят друг от друга. Считается, что это устаревшее архитектурное решение, однако оно подходит под некоторые проекты. Рассмотрим достоинства и недостатки монолитной архитектуры.

Достоинства:

1 Простота развертывания. Монолиты очень быстро и относительно просто развертывать благодаря тому, что у монолита обычно единая точка входа.

2 Разработка. Разработка монолита обычно происходит быстро, потому что все компоненты и модули находятся в одной кодовой базе.

3 Отладка. Отладка монолита очень упрощена за счет того, что есть возможность отследить все звенья выполнения кода.

Недостатки:

1 Масштабирование. Монолиты масштабируются только целиком, то есть если нагрузка растет только на один модуль, невозможно масштабировать только этот модуль, необходимо масштабировать весь монолит.

2 Надежность. Если монолит выходит из строя, то выходит весь целиком.

3 Изменение и обновление технологий. В монолите это почти невозможно.

4 Недостаточная упругость. Монолиты негибкие, то есть смена одного модуля в монолите почти всегда будет влиять на другой модуль.

Микросервисы – это архитектурное решение, которое базируется на распределении модулей на отдельные системы, которые общаются между собой посредством сообщений.

Вся система – это набор маленьких систем, которые связаны между собой.

Микросервисная архитектура очень популярна, но у нее есть свои недостатки, и она не подходит для маленьких и средних проектов.

Преимущества:

1 Гибкость. Микросервисы очень гибки за счет того, что каждый сервис является самостоятельной системой, поэтому изменения в ней могут повлиять только на нее.

2 Масштабирование. В отличие от монолита, возможно масштабировать только определенную часть системы потому, что она самостоятельна.

3 Гибкость технологий. В микросервисах каждая из подсистем может быть реализована на любом языке программирования и с помощью любых технологий.

4 Надежность. Обычный выход из строя одной из подсистем не ломает всю систему в целом.

Недостатки:

1 Процесс разработки. Микросервисы очень непросто разрабатывать, потому что необходимо производить несколько подсистем и наладить взаимодействие между ними.

2 Коммуникация команд. Очень часто бывает так, что одна команда разрабатывает одну подсистему, поэтому нужно наладить коммуникации между командами, чтобы наладить взаимодействие между подсистемами.

3 Отладка. Микросервисы очень сложно отлаживать, потому что нужно найти, какой сервис сломался, и почему.

4 Развертывание. Начальное развертывание очень непростое, и добавление новых сервисов требует настройки ключевых частей проекта.

Серверлесс – это архитектурное решение, которое фокусируется на разработке вместо развертывания и взаимодействия между сервисами.

Серверлесс – это альтернатива микросервисной архитектуры, которая автоматизирует все развертывание благодаря облачным технологиям. Из названия можно подумать, что здесь отсутствует серверная часть, но это не так, здесь отсутствует работа с сервером как со средой.

Достоинства:

1 Гибкость. Серверлесс очень гибок за счет обособленности одного модуля от другого.

2 Абстракция от операционной системы. Облако само решит, какая ОС нужна и как ее нужно настроить.

3 Легкий порог входа. Обычно серверлесс – это очень простой, обособленный код, поэтому разобраться с проектом несложно.

4 Надежность. При правильном построении проекта надежность такая же, как и у микросервисной архитектуры.

Недостатки:

1 Гибкость. Это тоже минус, потому что у разработчика ограничено влияние на масштабирование, все решает облако.

2 Отладка. Так же, как и в микросервисной архитектуре, отладка усложнена взаимодействием между компонентами.

3 Vendor Lock. Каждое облако работает по собственным правилам, поэтому переезд с одного облака на другое почти невозможен.

4 Cascade Failur. При неправильном построении проекта компоненты могут сильно влиять на другие компоненты, что приводит к падению всего проекта.

Любой проект можно выполнить на любой архитектуре, но это может быть неэффективным либо в плане разработки, либо в плане эксплуатации проекта. Однако, так как курсовой проект является маленьким проектом, эффективнее всего реализовать его через монолитную архитектуру.

## 2.2 Обоснование выбора технологии

При разработке системы «УЧИ.БЕЛ» были использованы следующие инструменты и технологии:

– язык гипертекстовой разметки HTML 5.0;

– формальный язык описания внешнего вида документа CSS 3.0;

– язык программирования JavaScript;

– язык программирования Python;

– фреймворк Django для языка программирования Python.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества. Стандартная библиотека включает большой набор полезных функций, начиная с возможностей для работы с текстом и заканчивая средствами для написания сетевых приложений, включая возможность работы с базами данных. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений, разработка игр, могут реализовываться посредством обширного количества сторонних библиотек. [5]

Основные достоинства языка программирования Python при разработке веб-приложений:

1 Python – это язык с динамической типизацией. Это означает, что нет необходимости определять тип данных переменной, поскольку она автоматически присваивает типы данных переменным во время выполнения.

2 Python легко читать и изучать благодаря синтаксису, похожему на английский. Кроме того, исключается использование точки с запятой после конца оператора и разделителей для начала и конца блока и фигурных скобок.

3 Поскольку Python является интерпретируемым языком, он выполняет код построчно, останавливает выполнение в случае ошибки и сообщает об этом. Благодаря этому, Python эффективно использует память.

4 Python – бесплатный язык программирования с открытым исходным кодом, что дает возможность загружать и изменять его исходный код.

5 Стандартная библиотека Python представляет собой полный набор модулей. Python совместим и переносится в системы Windows, macOS и Unix/Linux.

6 Не имеет указателей, что не нарушает изолированность и безопасность кода.

8 Совместимость с огромным количество баз данных, поэтому Python считается одни из лучших языков для работы с базами данных. [6]

Django — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Созданный опытными разработчиками, Django берёт на себя большую часть хлопот веб-разработки, поэтому вы можете сосредоточиться на написании своего веб-приложения без необходимости изобретать велосипед. Он бесплатный и с открытым исходным кодом, имеет растущее и активное сообщество, отличную документацию и множество вариантов как бесплатной, так и платной поддержки. [7]

Преимущества фреймворка Django:

1 Встроенный интерфейс администратора с уже имеющимися переводами на многие языки;

2 Диспетчер URL на основе регулярных выражений;

3 Имеет интуитивно понятный синтаксис и простую структуру.

4 Расширяемая система шаблонов с тегами и наследованием;

5 Много подробной документации на русском языке. [8]

HTML – стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора. [9]

CSS – формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL. [10]

JavaScript – это язык программирования, который используют для написания frontend- и backend-частей сайтов, а также мобильных приложений. JavaScript поддерживают все популярные браузеры. Язык используют для создания интерактива (анимаций, всплывающих форм, автозаполнения), так как он связан с HTML и CSS и может ими манипулировать. JavaScript называют сценарным языком, так как с его помощью можно задать набор инструкций, которые выполняются при загрузке страницы. [11]

## 2.3 Описание среды реализации

При разработке системы «УЧИ.БЕЛ» были выбраны следующие среды реализации:

– СУБД MySQL SEVER;

– среда разработки PyCharm;

– среда разработки Visual Studio Code;

– Figma.

MySQL SEVER – свободная реляционная система управления базами данных, одна из самых распространенных в мире. Основными достоинствами данной СУБД являются надежность, быстродействие, безопасность и бесплатная форма распространения. [12]

PyCharm – это кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains на основе IntelliJ IDEA. Предоставляет пользователю комплекс средств для графических отладчиков и работы с кодом. Имеет удобный редактор кода и обширные возможности отладки приложения. Является лучшим вариантом для разработки веб-приложений. [13]

Visual Studio Code – текстовый редактор, разработанный Microsoft для операционных систем Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки WEB и облачных приложений. Для курсового проектирования используется в качестве интерфейса программирования веб-приложений с помощью разметки гипертекста HTML 5.0, таблицы стилей CSS 3.0 и языка программирования JavaScript. [14]

Figma – сервис для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени. Используется как для создания упрощённых прототипов интерфейсов, так и для детальной проработки дизайна интерфейсов мобильных приложений, веб-сайтов, корпоративных порталов. [15]

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## 3.1 Физическая структура базы данных

Физическая структура базы данных – модель данных, определяющая форму представления данных в СУБД.

Для организации хранения данных необходимо создать следующие таблицы: «Курс», «Изображение», «Лекция», «Урок», «Оценка», «Предмет», «Преподаватель». В схеме базы данных используются следующие наименования соответственно: «main\_courses», «main\_image», «main\_lecture», «main\_lessons», «main\_mark», «main\_subjects», «main\_teachers». Помимо приведенных выше таблиц база данных содержит таблицы, которые Django добавляет самостоятельно. К ним относятся: «auth\_group\_permissions», «auth\_permission», «auth\_user», «auth\_user\_groups», «auth\_user\_user\_permissions», «django\_admin\_log», «django\_content\_type», «django\_migrations», «django\_session». Схема базы данных представлена в приложении Б.

Далее приведена подробная информация о таблицах, которые были созданы самостоятельно. В таблицах используется обозначения привилегий. Описание привилегий представлено в таблице 3.1.

Таблица 3.1 – Описание привилегий

|  |  |
| --- | --- |
| Привилегия | Описание |
| PK – Primary Key | Обозначает первичный ключ данной сущности. |
| FK – Foreign Key | Обозначает внешний ключ данной сущности. |
| NN – Not Null | Означает, что при внесении данных данное поле не будет пустым. |
| AI – Auto Incremental | Означает, что поля атрибута будут автоматически заполняться натуральными значениями: 1, 2, 3 и т.д. |

Таблица «Курс» («main\_courses») хранит в себе всю необходимую информацию о курсах. Структура приведена в таблице 3.2.

Таблица 3.2 – Структура таблицы «Курс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | PK, AI, NN | Уникальный идентификатор курса. |
| title | VARCHAR (50) | 100 | NN | Название курса. |
| duration | INT | 4 | NN | Длительность курса. |
| idMark\_id | BIGINT | 8 | FK, NN | Уникальный идентификатор оценки. |
| idSubject\_id | BIGINT | 8 | FK, NN | Уникальный идентификатор предмета. |
| idTeacher\_id | BIGINT | 8 | FK, NN | Уникальный идентификатор преподавателя. |

Таблица «Изображение» («main\_image») хранит в себе информацию о месте хранения изображения. Структура приведена в таблице 3.3.

Таблица 3.3 – Структура таблицы «Изображение»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | PK, NN, AI | Уникальный идентификатор изображения. |
| image | VARCHAR (100) | 200 | NN | URL адрес хранения изображения. |

Таблица «Лекция» («main\_lecture») хранит в себе всю необходимую информацию о лекциях. Структура приведена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Лекция»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | NN, PK, AI | Уникальный идентификатор лекции. |
| title | VARCHAR (50) | 100 | NN | Название лекции. |
| info | LONGTEXT | 4e+9 | NN | Информация о лекции. |

Таблица «Урок» («main\_lessons») хранит в себе всю необходимую информацию об уроках. Структура приведена в таблице 3.5.

Таблица 3.5 – Структура таблицы «Урок»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | NN, PK, AI | Уникальный идентификатор урока. |
| title | VARCHAR (50) | 100 | NN | Название урока. |
| info | LONGTEXT | 4e+9 | NN | Информация об уроке. |

Таблица «Оценка» («main\_mark») хранит в себе всю необходимую информацию об оценках. Структура приведена в таблице 3.6.

Таблица 3.6 – Структура таблицы «Оценка»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | NN, PK, AI | Уникальный идентификатор оценки. |
| title | INT | 4 | NN | Значение оценки. |

Таблица «Предмет» («main\_subjects») хранит в себе всю необходимую информацию о предметах. Структура приведена в таблице 3.7.

Таблица 3.7 – Структура таблицы «Предмет»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | NN, PK, AI | Уникальный идентификатор предмета. |
| title | VARCHAR (50) | 10 | NN | Название предмета. |
| idImage\_id | BIGINT | 8 | NN, FK | Уникальный идентификатор изображения, связанного с предметом. |

Таблица «Преподаватель» («main\_teachers») хранит в себе всю необходимую информацию о преподавателях. Структура приведена в таблице 3.8.

Таблица 3.8 – Структура таблицы «Преподаватель»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Тип данных | Размер, байт | Привилегии | Описание |
| id | BIGINT | 8 | NN, PK, AI | Уникальный идентификатор преподавателя. |
| name | VARCHAR (50) | 100 | NN | Имя преподавателя. |

Следующие таблицы связаны между собой *необязательной* связью «один-ко-многим»:

– «main\_lessons» – «main\_lessons\_lectures»;

– «main\_lecture» – «main\_lessons\_lectures»;

– «main\_mark» – «main\_courses»;

– «main\_teachers» – «main\_courses»;

– «main\_lectures» – «main\_lectures\_courses»;

– «main\_lectures\_courses» – «main\_courses»;

– «main\_image» – «main\_subjects»;

– «main\_subjects» – «main\_courses».

Обоснование использования типов данных:

1 Тип данных BIGINT использован для создания первичных и внешних ключей, а также для заполнения числовых атрибутов, для которых нет необходимости использовать текстовый тип данных.

2 Тип данных VARCHAR использован для заполнения текстовых атрибутов. В таблице «users» данный тип данных используется в том числе для создания первичного ключа (атрибут «login»).

3 Тип данных INT использован для заполнения числовых атрибутов, для которых нет необходимости использовать текстовый тип данных.

## 3.2 Структура информационной системы

Структура информационной системы состоит из следующих компонентов:

– клиент;

– веб-приложение;

– база данных.

Графическое представление структуры изображено в приложении А.

Компонент системы «Клиент» представляет собой пользовательский интерфейс пользователя. На рисунке 3.1 представлен снимок экрана, показывающий один из модулей компонента «Клиент».

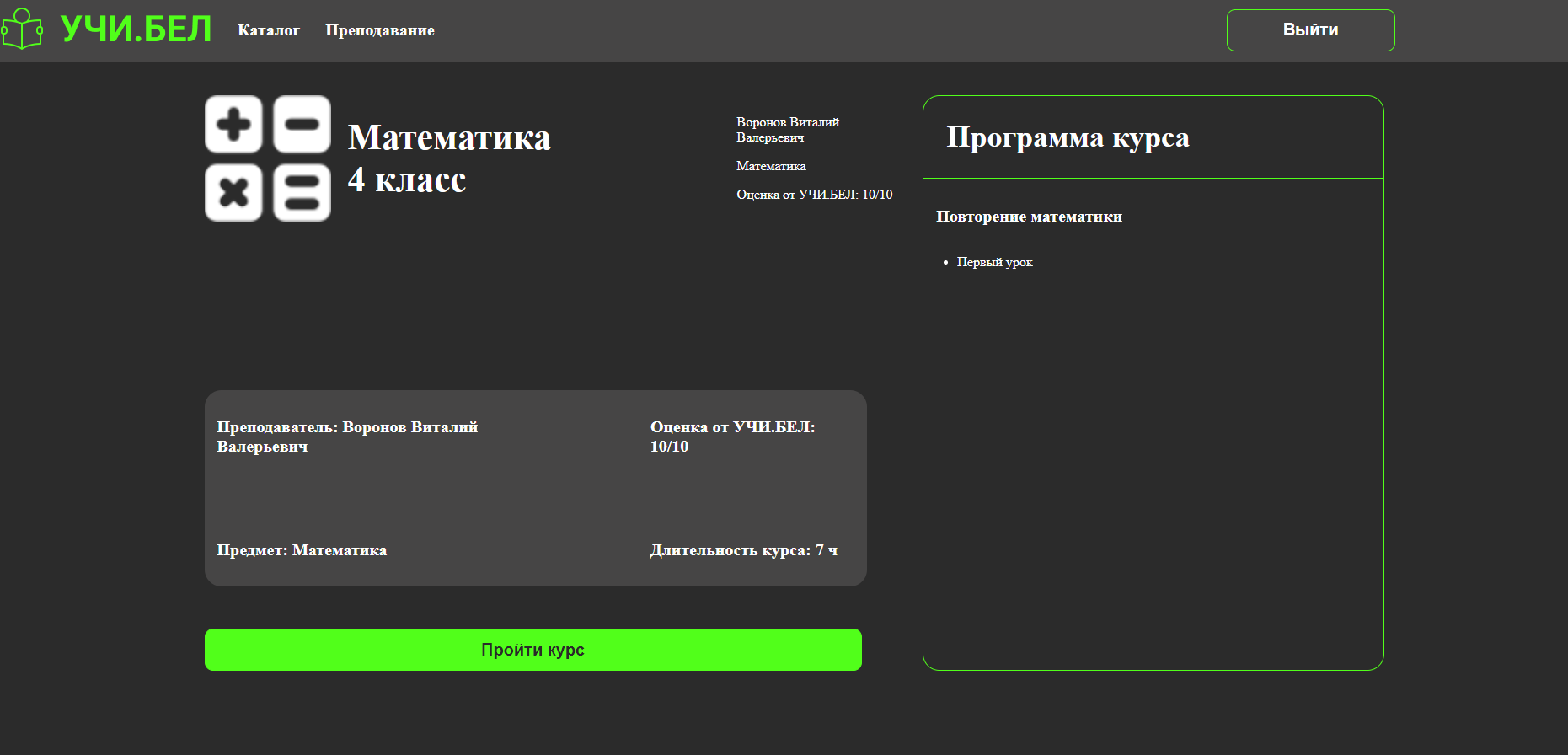


Рисунок 3.1 – Один из модулей компонента «Клиент»

Компонент системы «Веб-приложение» представляет собой набор файлов, реализующих бизнес-логику системы. Структура компонента состоит из следующих элементов:

– views.py – файл, в котором описаны все функции системы;

– models.py – файл, в котором описаны все классы, используемые в компоненте «Веб-приложение»;

– папка «templates» содержит все файлы с расширением html;

– папка «static» содержит файлы с расширениями css и js, а также папку «img», содержащую изображения.

Компонент системы «База данных» представляет собой базу данных, которая хранит в себе всю необходимую информацию для работы системы. Описание базы данных представлено в разделе «3.1 Физическая структура базы данных».

## 3.3 Тестирование системы

Тестирование системы проводилось с помощью написания позитивных и негативных тест-кейсов с добавлением в таблицы тест-кейсов колонки с описанием фактического результата.

Тест-кейс – это четкое описание действий, которые нужно выполнить для проверки отдельной функции приложения. [16]

В позитивных тест-кейсах используются корректные входные данные и сценарии ожидаемой работы системы. Цель – убедиться, что программный продукт выполняет то, что должен делать, и что система не выдаст ошибку, если это не предусмотрено. [16]

Негативные тест-кейсы используют некорректные входные данные и проверяют, не делает ли программа того, чего не должна делать. Негативное тестирование призвано гарантировать, что при получении некорректных входных данных система не будет работать по нормальному сценарию (например, отобразит ошибку). [16]

Перечень позитивных и негативных тест-кейсов представлен в таблице 3.9.

Таблица 3.9 – Перечень позитивных и негативных тест-кейсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Модуль | Описание теста | Ожидаемый результат | Фактический результат |
| NTC-1 | Авторизация | Авторизация.  1 Ввести корректное значение в поле «Логин» символы.  2 Ввести корректное значение в поле «Введите пароль».  3 Нажать на кнопку «Войти». | 1 Введен логин.  2 Введен пароль.  3 На главном окне появляется кнопка «Выйти» | 1 Выполнено.  2 Выполнено.  3 Выполнено. |
| NTC-2 | Авторизация | Авторизация.  1 Ввести в поле «Введите логин» незарегистрированный логин, используя корректные символы.  2 Ввести в поле «Введите пароль» любое значение.  3 Нажать на кнопку «Войти в аккаунт». | 1 Введен логин.  2 Введен пароль.  3 Страница авторизации обновляется. | 1 Выполнено.  2 Выполнено.  3 Выполнено. |
| NTC-3 | Авторизация | Авторизация.  1 Ввести корректный логин в поле «Введите логин».  2 Ввести в поле «Введите пароль» пароль, не соответствующий логину.  3 Нажать на кнопку «Войти в аккаунт». | 1 Введен логин.  2 Введен пароль.  3 Страница авторизации обновляется. | 1 Выполнено.  2 Выполнено.  3 Выполнено. |
| NTC-4 | Авторизация | Авторизация.  1 Оставить поля «Введите логин» и «Введите пароль» пустыми (либо одно из двух, либо оба).  2 Нажать на кнопку «Войти в аккаунт». | 1 Выбранные поля пустые.  2 Ошибка «Заполните это поле» | 1 Выполнено.  2 Выполнено. |
| NTC-5 | Регистрация | Регистрация.  1 Ввести в поле «Введите логин» уже зарегистрированный логин.  2 Ввести любое значение в поле «Введите пароль».  3 Ввести любое значение в поле «Повторите пароль».  4 Нажать на кнопку «Создать аккаунт». | 1 Введен логин.  2 Введен пароль.  3 Пароль повторен.  4 Окно регистрации обновляется. | 1 Выполнено.  2 Выполнено.  3 Выполнено.  4 Выполнено. |
| NTC-6 | Регистрация | Регистрация.  1 Оставить все поля пустыми или любое из них.  2 Нажать на кнопку «Создать аккаунт». | 1 Выбранные поля пустые.  2 Ошибка «Заполните это поле». | 1 Выполнено.  2 Выполнено. |
| NTC-7 | Просмотр информации о курсе | Переход на страницу просмотра полной информации о курсе  1 Быть авторизованным на платформе.  2 Нажать на кнопку «Пройти курс». | 1 Переход на страницу просмотра полной информации о курсе. | 1 Выполнено. |
| NTC-8 | Добавление нового курса | Добавление нового курса  1 Нажать на кнопку «Добавить курс».  2 Оставить поле «Название» пустым.  3 Выбрать оценку из выпадающего списка поля «IdMark».  4 Выбрать преподавателя из выпадающего списка поля «IdTeacher».  5 Ввести корректное значение в поле «Длительность».  6 Выбрать предмет из выпадающего спика поля «IdSubject».  7 Нажать на кнопку «Сохранить» | 1 Над полем названия появляется ошибка «Обязательно поле».  2 Ошибка «Please correct the error below.». | 1 Выполнено.  2 Выполнено. |
| NTC-9 | Редактирование курса | Редактирование курса  1 Выбрать необходимый курс  2 Оставить поле «Название» пустым.  3 Выбрать оценку из выпадающего списка поля «IdMark».  4 Выбрать преподавателя из выпадающего списка поля «IdTeacher».  5 Ввести корректное значение в поле «Длительность».  6 Выбрать предмет из выпадающего спика поля «IdSubject».  7 Нажать на кнопку «Сохранить» | 1 Над полем названия появляется ошибка «Обязательно поле».  2 Ошибка «Please correct the error below.». | 1 Выполнено.  2 Выполнено. |
| NTC-10 | Удаление курса | Удаление курса  1 Выбрать чекбокс на необходимом курс.  2 В выпадающем списке «Выполнить» выбрать «Удалить выбранные Курсы».  3 Нажать на кнопку «Выполнить».  4 В окне подтверждения нажать на кнопку «Да, я уверен» | 1 Сообщение «Успешно удалены 1 курс» | 1 Выполнено. |

Также было проведено функциональное тестирование по методу «белого ящика».

## 3.4 Описание применения

При запуске приложения пользователь попадает на главную страницу приложения, показанную на рисунке 3.2.

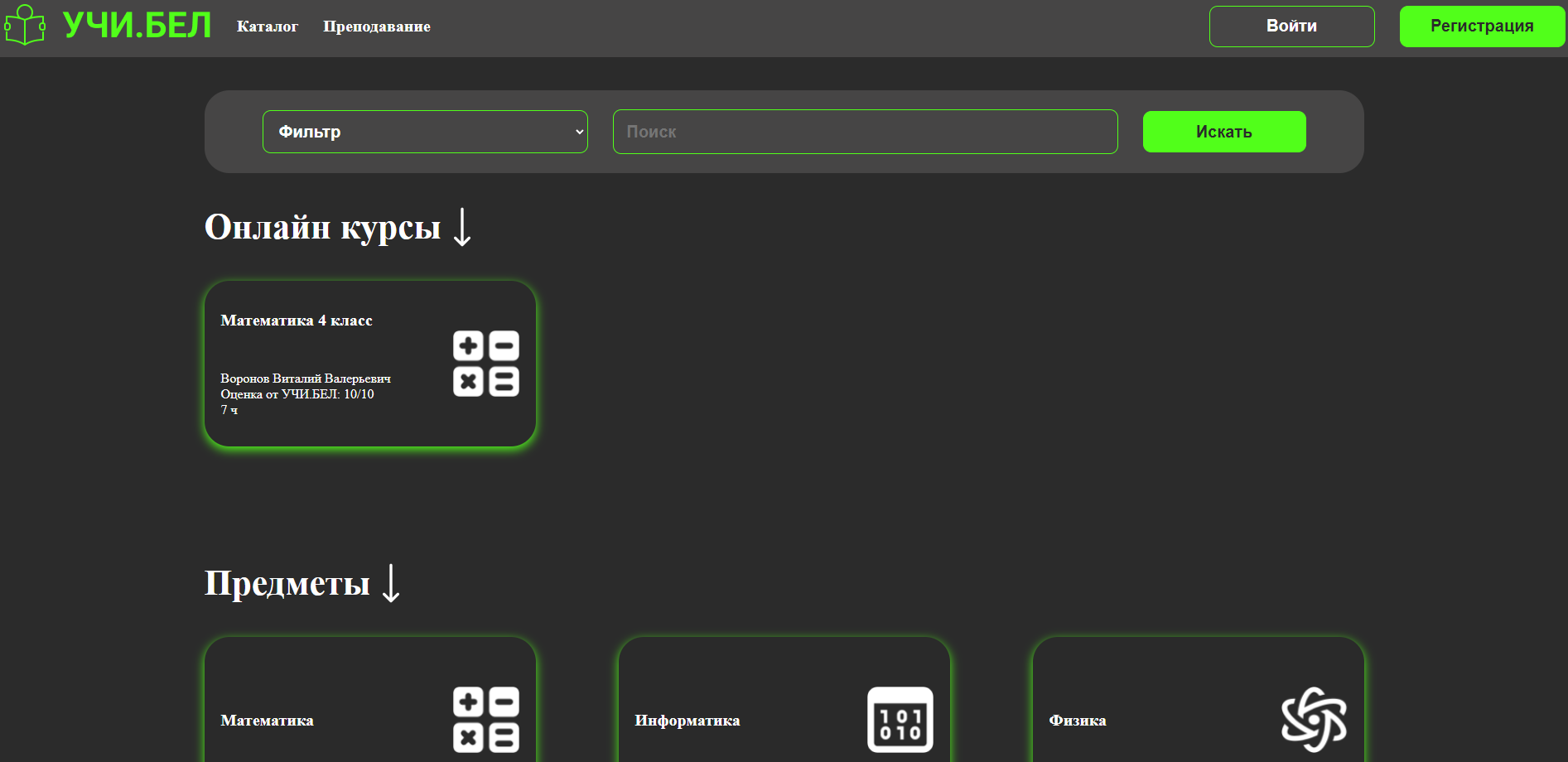


Рисунок 3.2 – Главная страница

При нажатии на кнопку «Войти», расположенную рядом с кнопкой «Регистрация», пользователь попадает на страницу авторизации, показанную на рисунке 3.3.

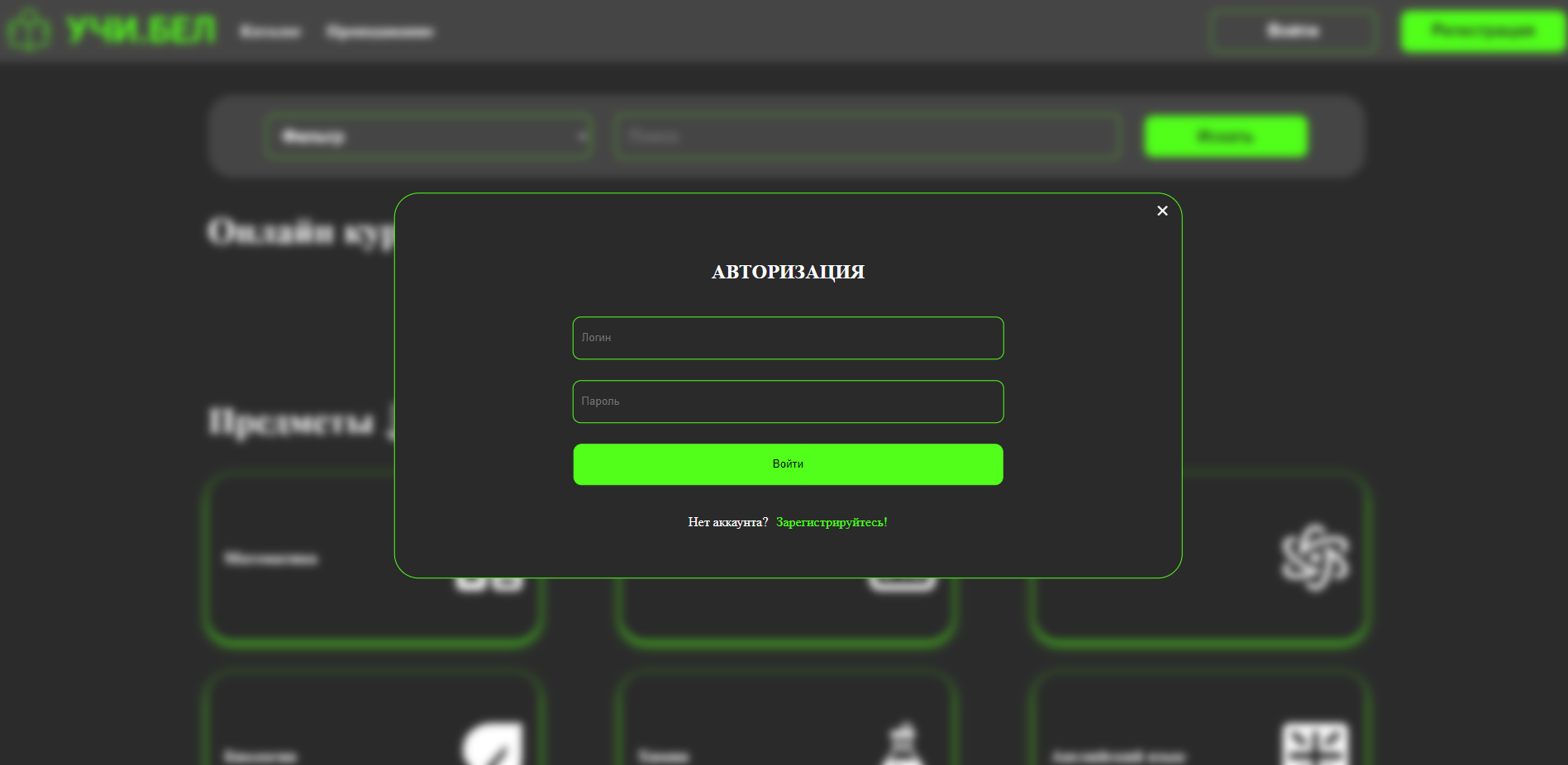
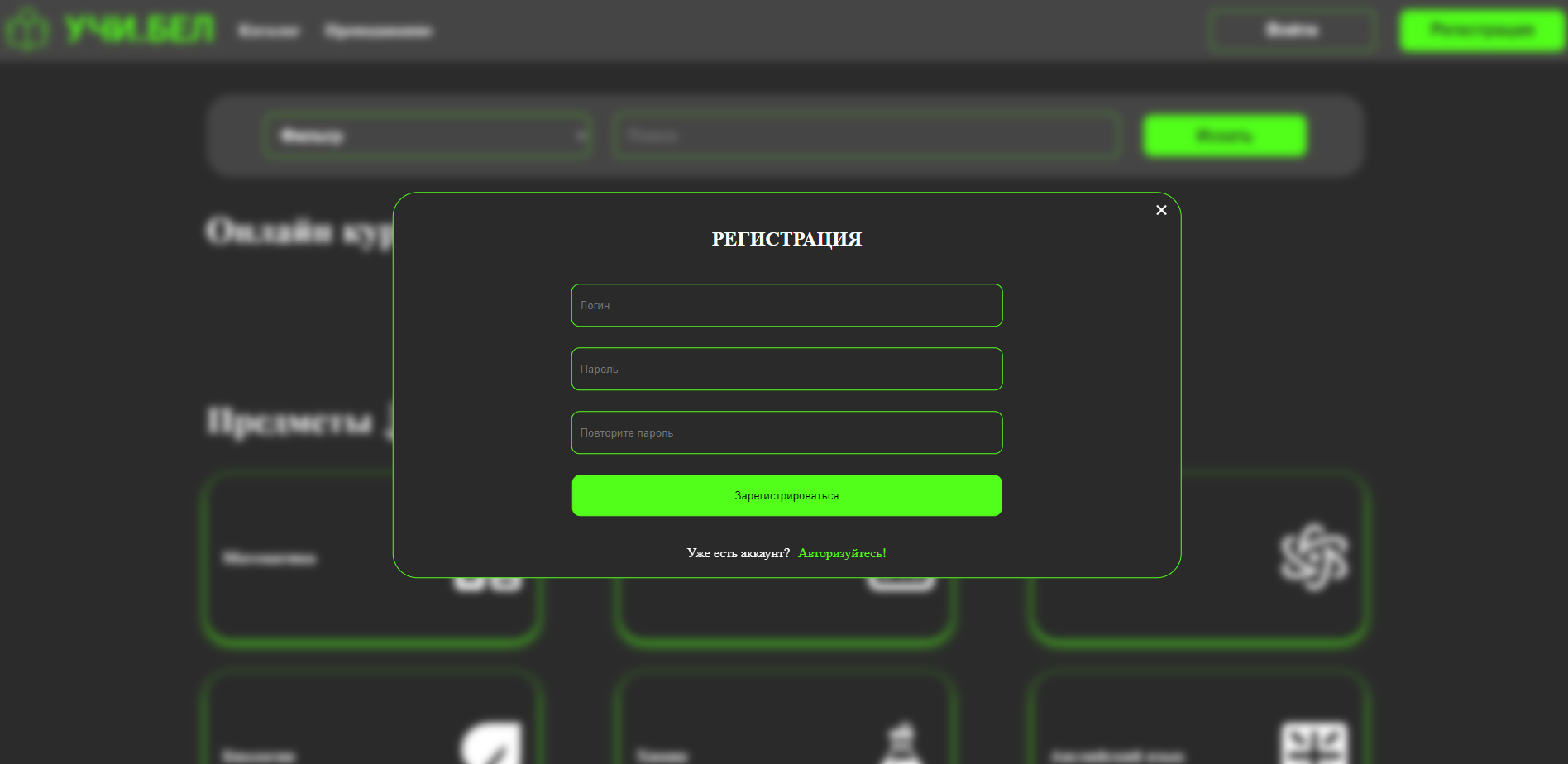
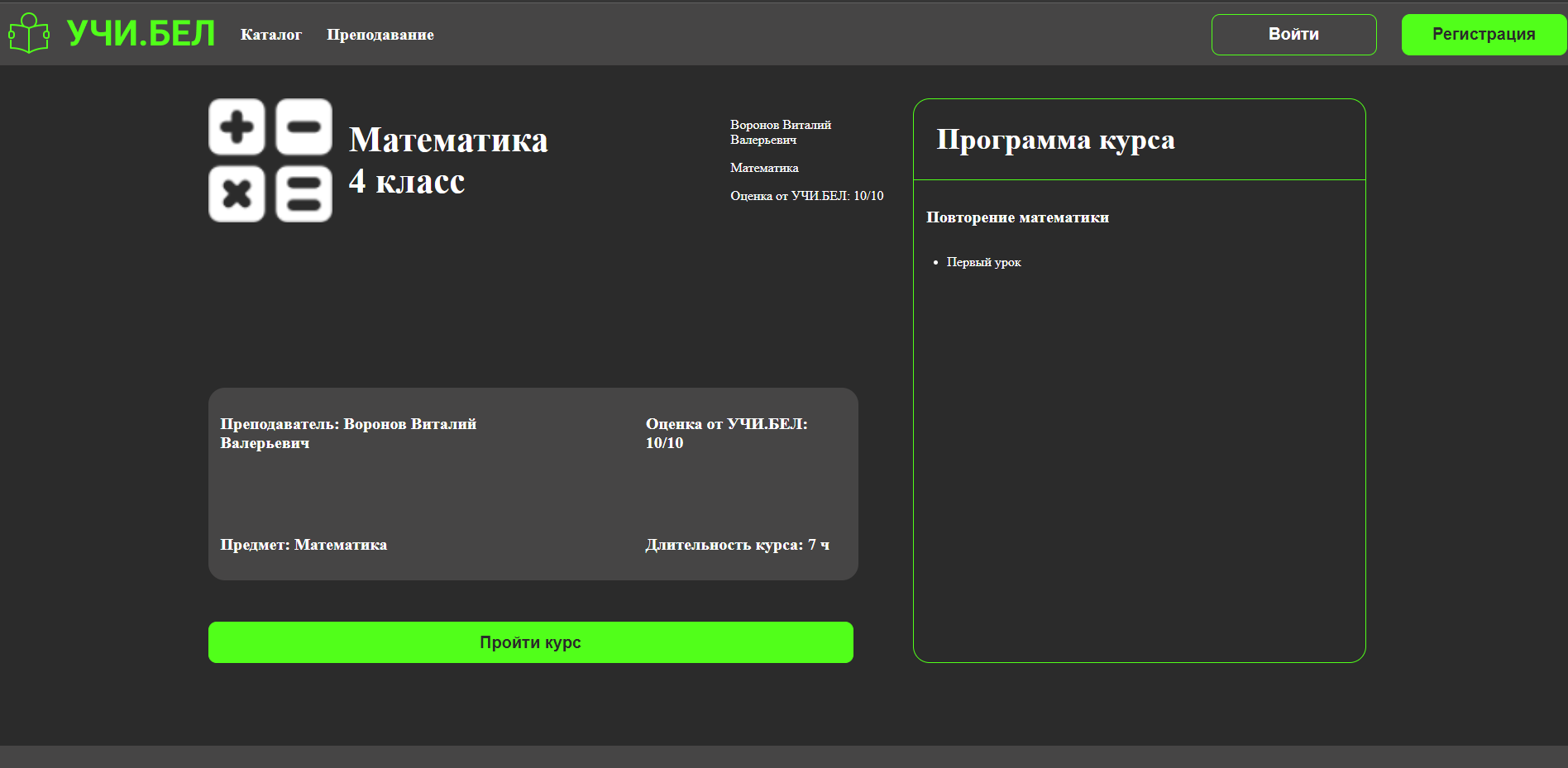


Рисунок 3.3 – Страница авторизации

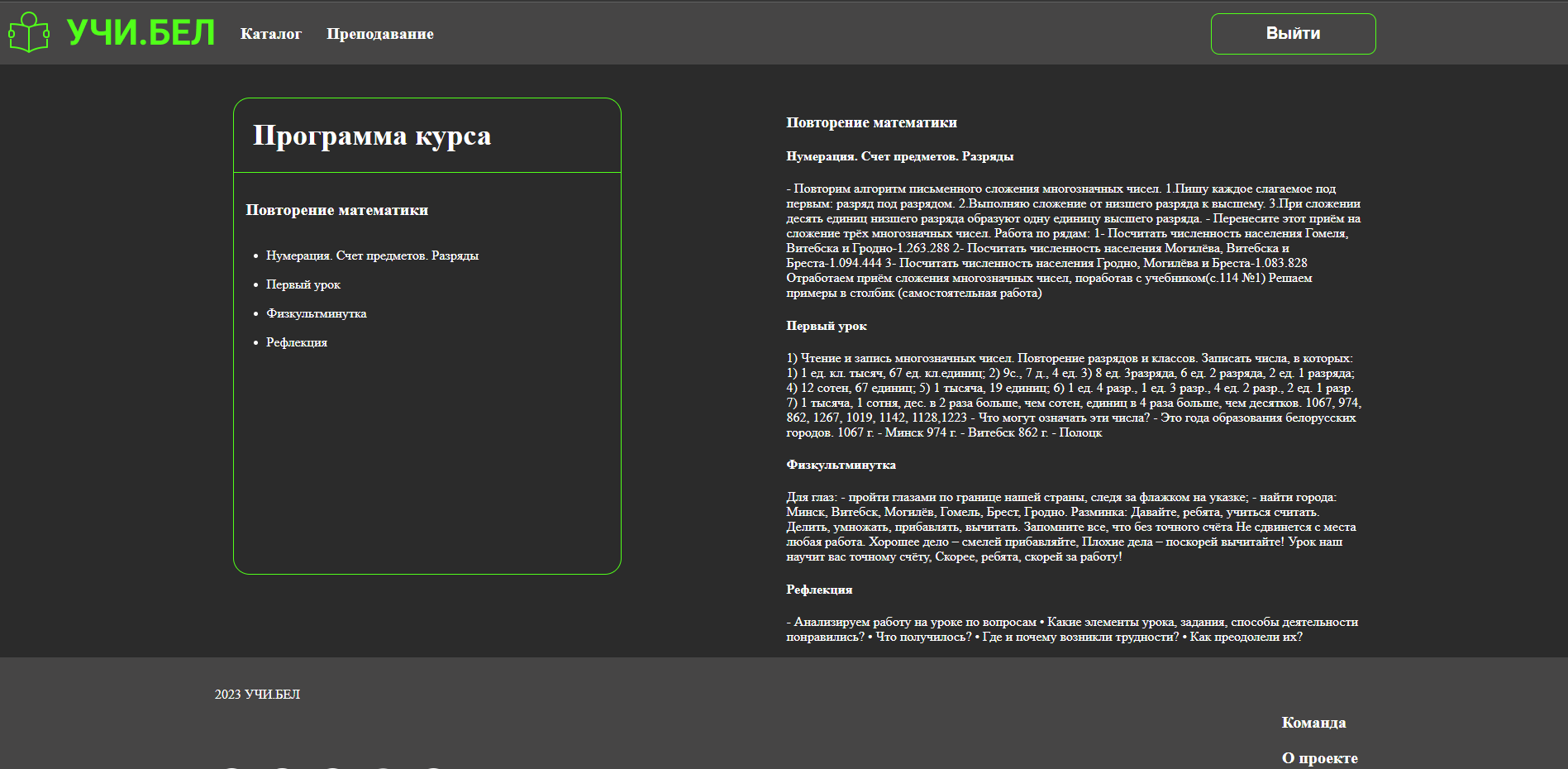
Если же нажать на кнопку «Регистрация», пользователь попадает на страницу регистрации, показанную на рисунке 3.4.

Рисунок 3.4 – Страница регистрации

При нажатии на определенный курс, который находится на главной странице, пользователь переходит на страницу с краткой информацией о данном курсе. Пример приведен на рисунке 3.5.

Рисунок 3.5 – Страница краткой информации о курсе

При успешной авторизации пользователь имеет возможность просмотреть полную информацию о курсе, нажав на кнопку «Пройти курс». Страница с полной информацией о курсе показана на рисунке 3.6.

Рисунок 3.6 – Страница полной информации о курсе

Чтобы перейти на страницу администратора необходимо на главной странице нажать на ссылку «Администраторы». В результате откроется страница авторизации администратора, приведенная на рисунке 3.7.

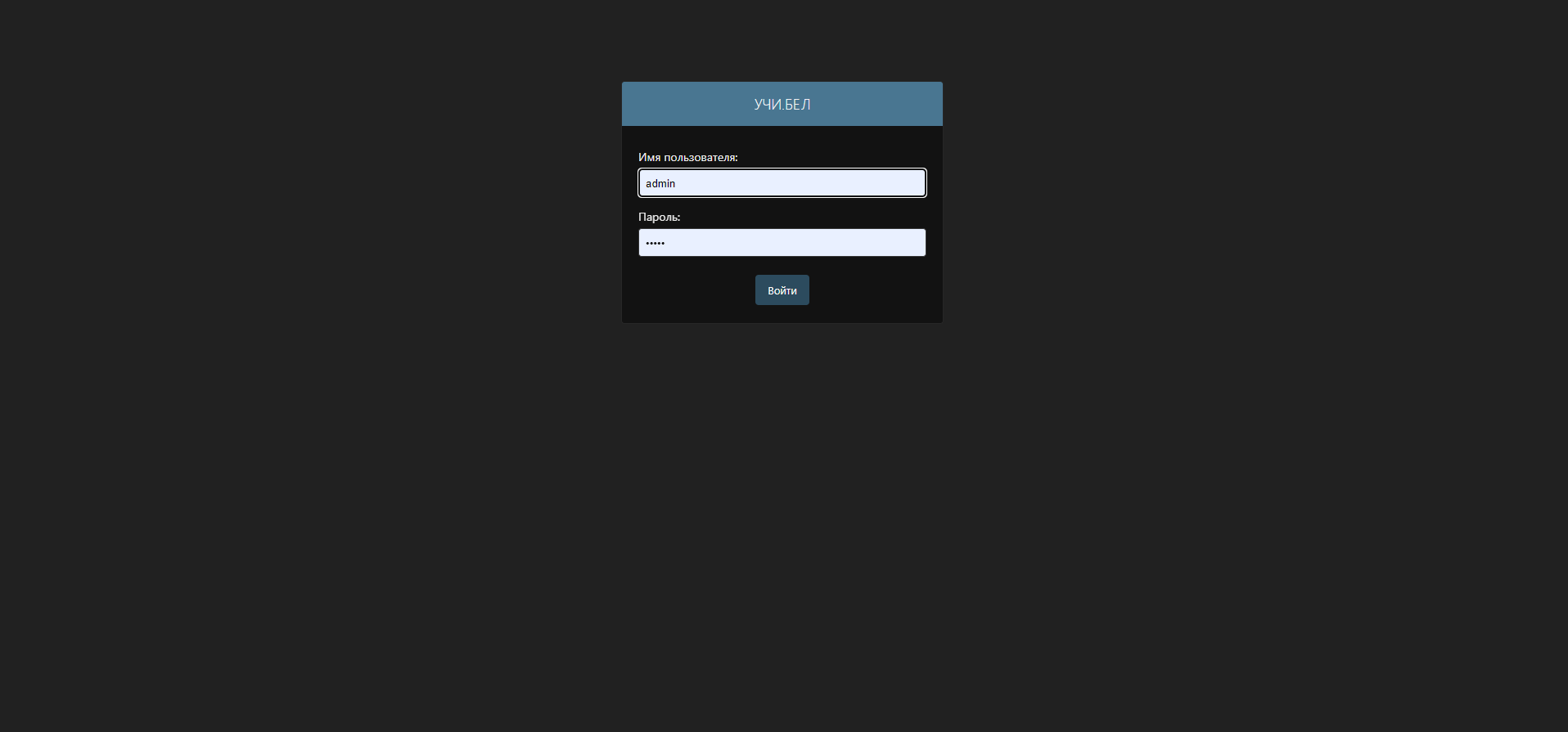


Рисунок 3.7 – Страница авторизации администратора

При успешном входе отобразится страница администратора, показанная на рисунке 3.8.

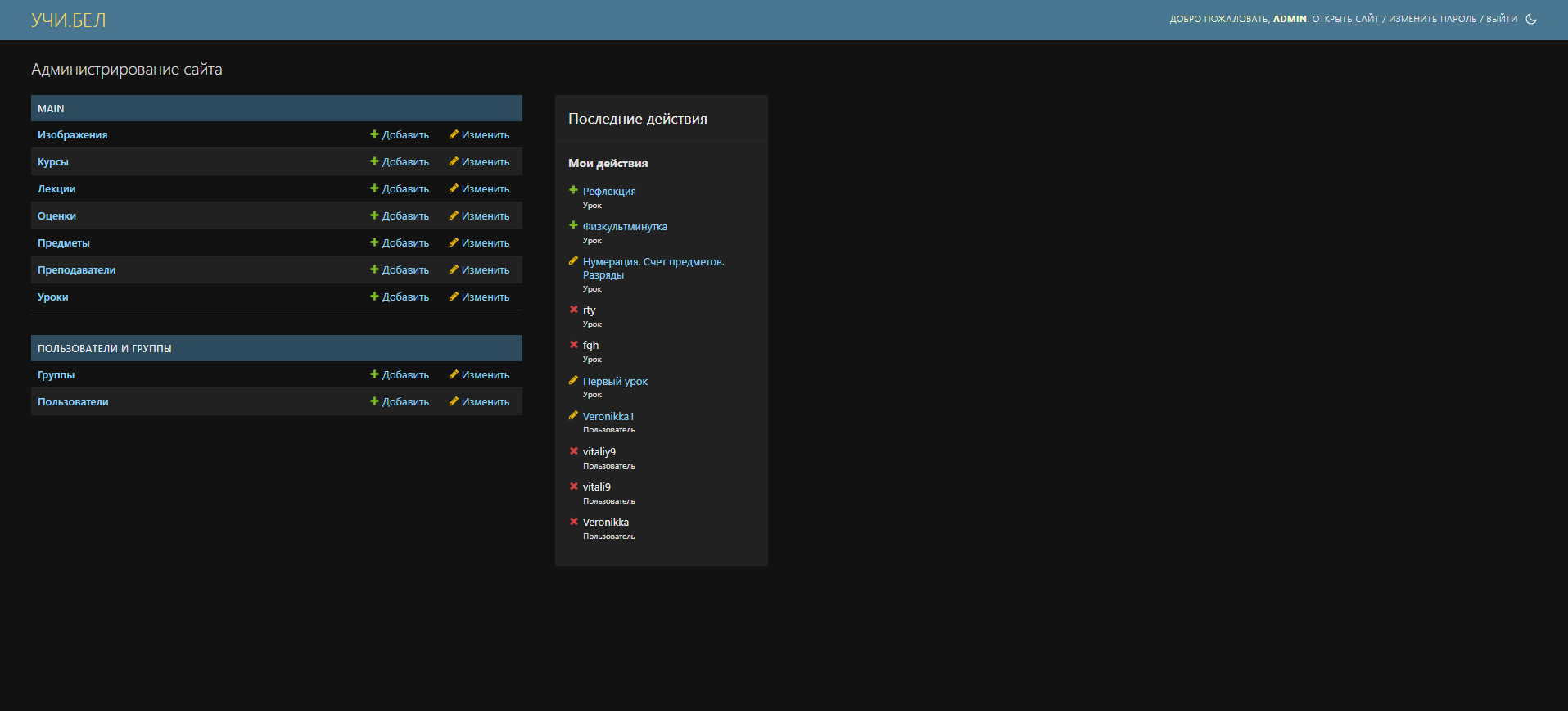


Рисунок 3.8 – Страница администратора

Принцип работы страницы администратора рассмотрим на примере редактирования оценок. Примеры редактирования оценок приведены на рисунках 3.9 и 3.10.

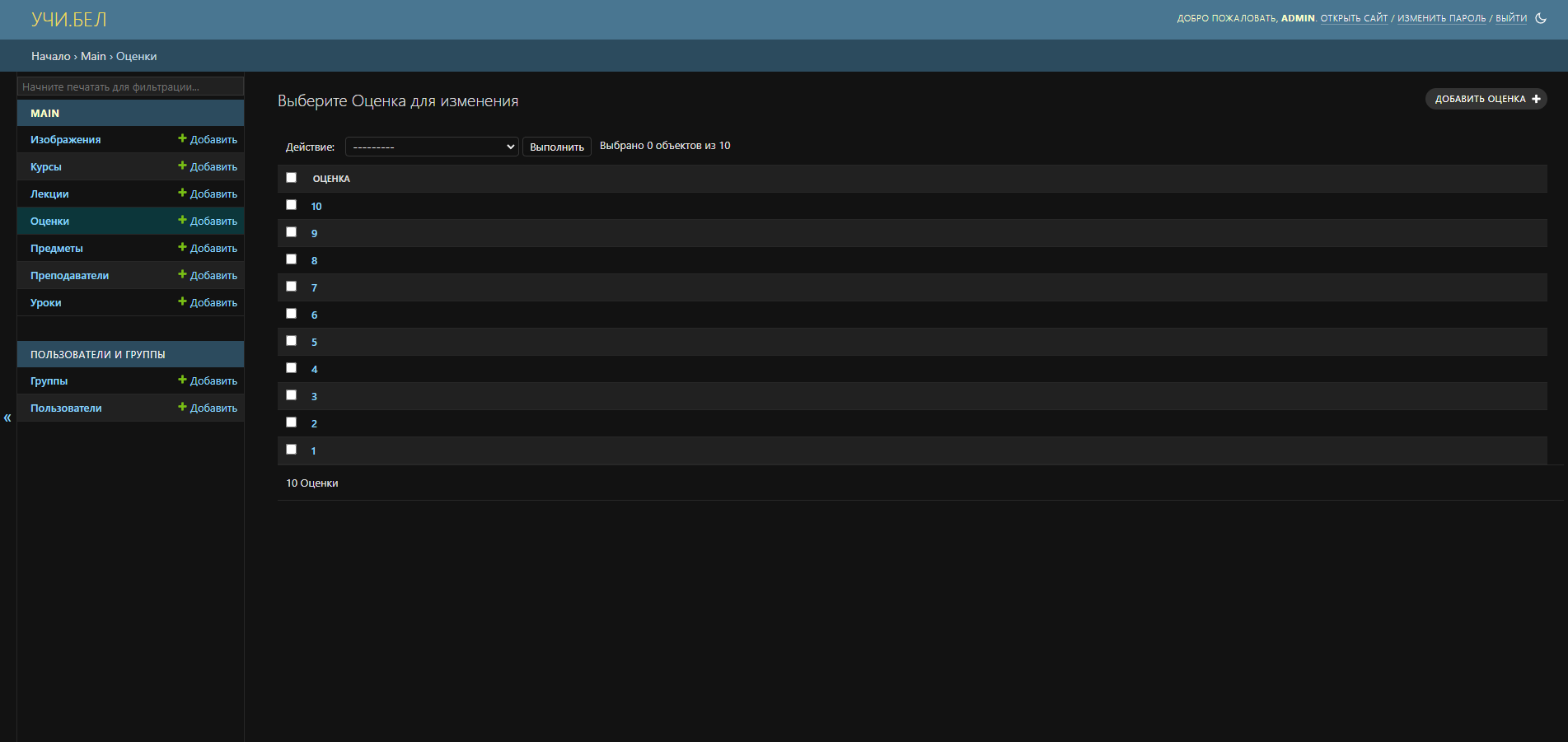


Рисунок 3.9 – Страница редактирования оценок

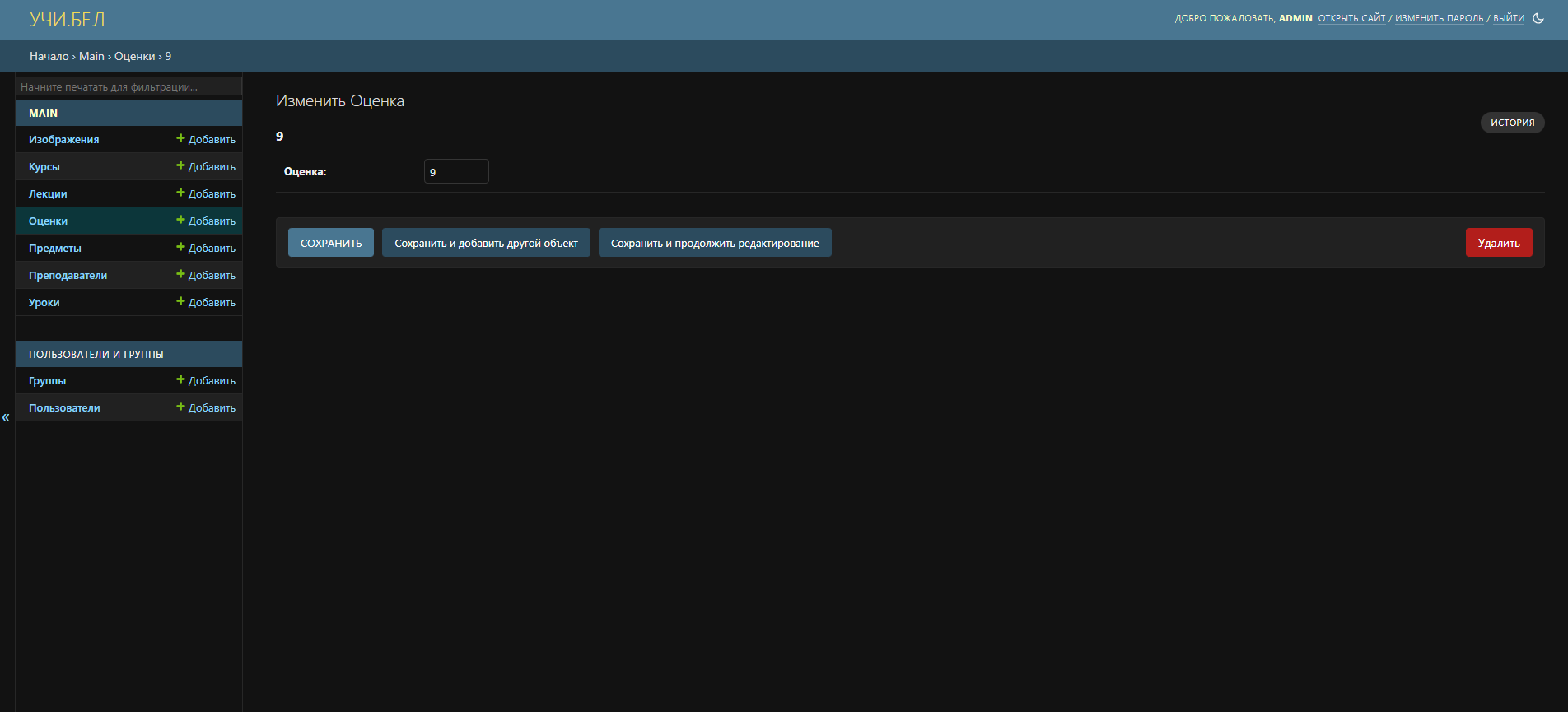


Рисунок 3.10 – Страница редактирования оценки(девятки)

ЗАКЛЮЧЕНИЕ

В процессе написания курсового проекта было разработано веб-приложение образовательных курсов, включающая в себя модули пользователя и администратора.

Все функциональные требования проекта были выполнены и протестированы.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

[1] Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D1%8B%D0%B9\_%D0%BA%D1%83%D1%80%D1%81.

[2] Stepik [Электронный ресурс]. – Режим доступа: https://welcome.stepik.org/ru.

[3] СТП 01-2017. Дипломные проекты (работы). Общие требования [Электронный ресурс]. – Режим доступа: <https://www.bsuir.by/m/12_100229_1_122976.pdf>.

[4] Hillelblog. Как выбрать архитектуру для веб-приложения [Электронный ресурс]. – Режим доступа: <https://blog.ithillel.ua/ru/articles/web-application-architecture>.

[5] ВикипедиЯ – Python [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Python.

[6] Devdelphi. «C# против Python – какой вариант лучше?» [Электронный ресурс]. – Режим доступа: <https://devdelphi.ru/blog/c-protiv-python-kakoi-variant-luchshe>.

[7] Mozilla. Django введение [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction

[8] Tproger. Django преимущества [Электронный ресурс]. – Режим доступа: <https://tproger.ru/articles/pochemu-vam-stoit-vybrat-frejmvork-django-dlja-svoego-sledujushhego-proekta/>

[9] ВикипедиЯ – HTML [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/HTML.

[10] ВикипедиЯ – CSS [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/CSS.

[11] Skillfactory Media. JavaScript [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/javascript/.

[12] MySQL Server для самых маленьких [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/mysql4u/.

[13] ВикипедиЯ – PyCharm [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/PyCharm.

[14] ВикипедиЯ – Visual Studio Code [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Visual\_Studio\_Code.

[15] ВикипедиЯ – Figma [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Figma.

[16] Testengeneer.ru – Как писать тест-кейсы: полное руководство [Электронный ресурс]. – Режим доступа: https://testengineer.ru/kak-pisat-test-kejsy-polnoe-rukovodstvo/#test-case-examples

ПРИЛОЖЕНИЕ А

**(обязательное)**

**Структура информационной системы**

На рисунке А.1 представлена структура информационной системы.



Рисунок А.1 – Структура информационной системы

ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Схема базы данных**

На рисунке Б.1 представлена схема базы данных.

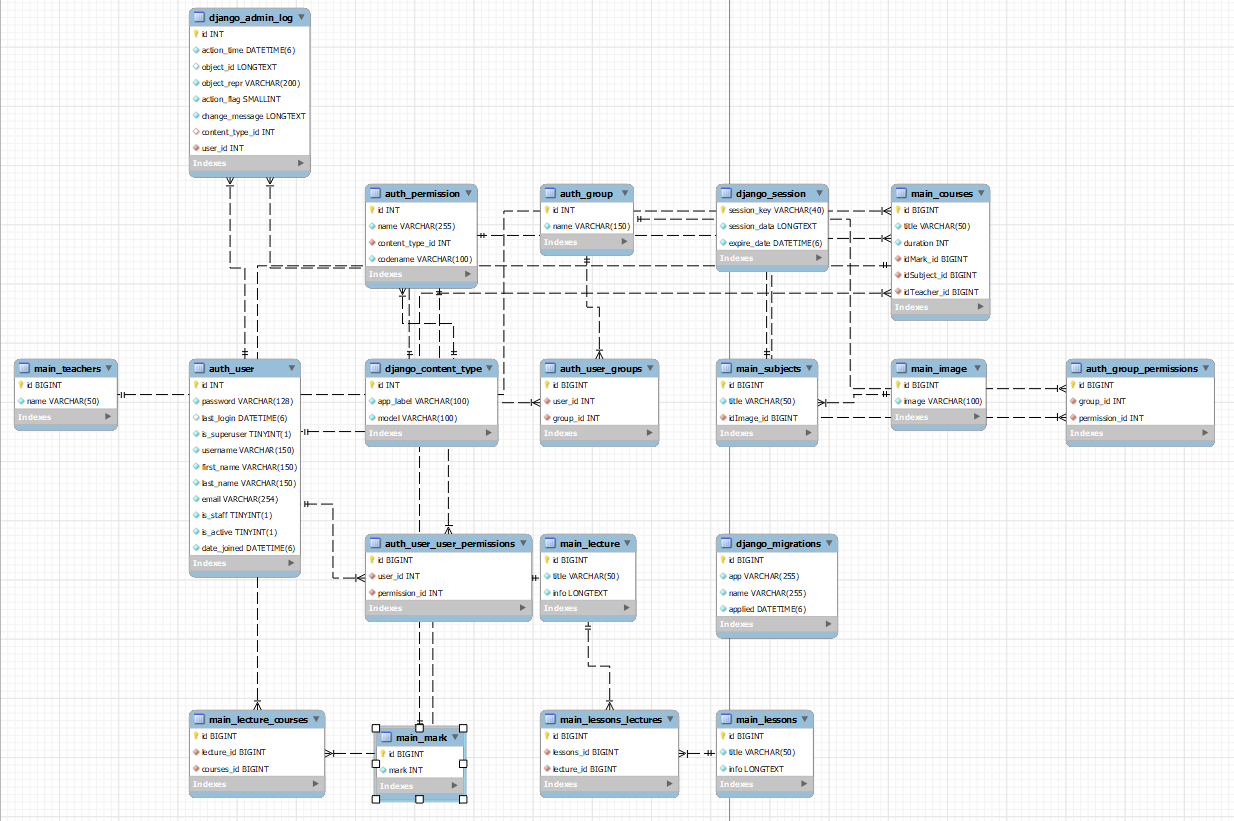


Рисунок Б.1 – Схема базы данных

ПРИЛОЖЕНИЕ В

**(обязательное)**

**Схема алгоритма работы пользователя**

На рисунке В.1 представлена схема алгоритма работы пользователя.

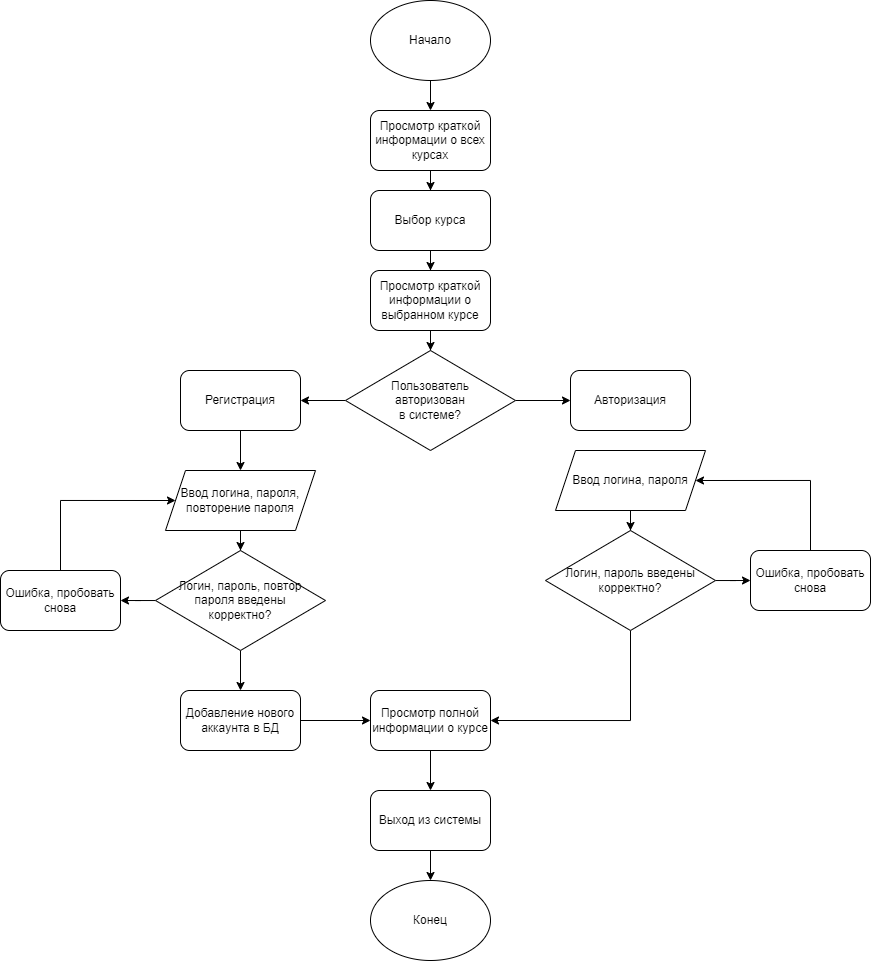


Рисунок В.1 – Схема алгоритма работы пользователя

ПРИЛОЖЕНИЕ Г

**(обязательное)**

**Схема алгоритма работы администратора**

На рисунке Г.1 представлена схема алгоритма работы администратора.



Рисунок Г.1 – Схема алгоритма работы администратора

ПРИЛОЖЕНИЕ Д

**(обязательное)**

**Диаграмма вариантов использования**

На рисунке Д.1 представлена диаграмма вариантов использования.

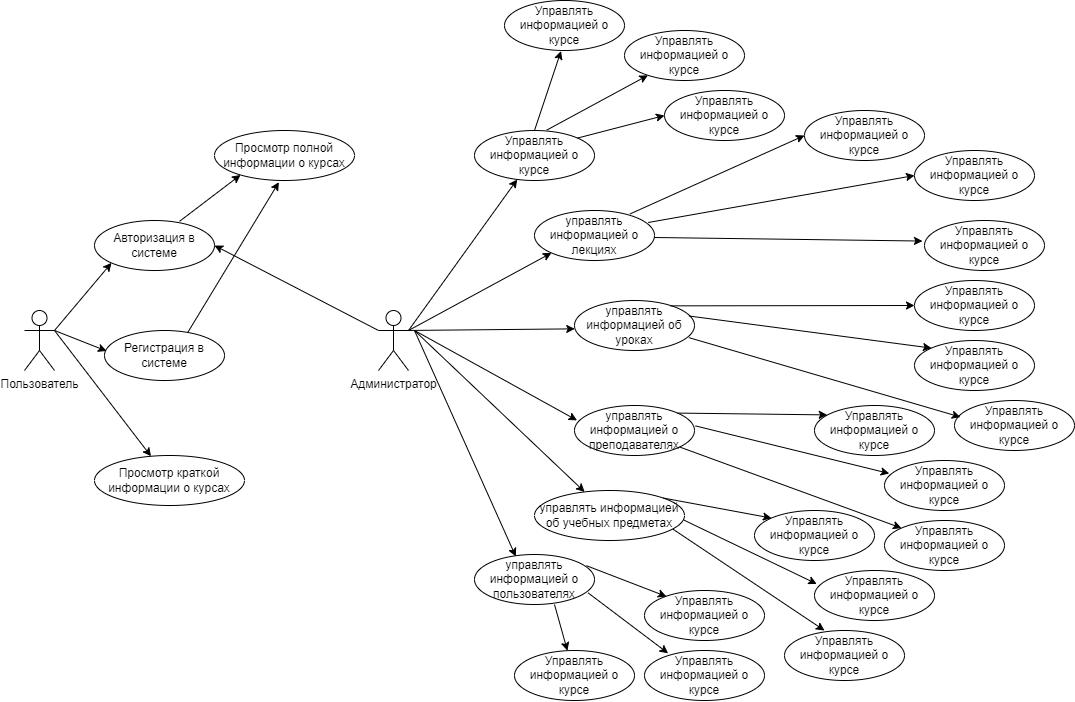


Рисунок Д.1 – Диаграмма вариантов использования

ПРИЛОЖЕНИЕ Е

**(обязательное)**

**Диаграмма взаимодействия**

На рисунке Е.1 представлена диаграмма взаимодействия. На диаграмме взаимодействия изображено отображение полной информации о курсе.

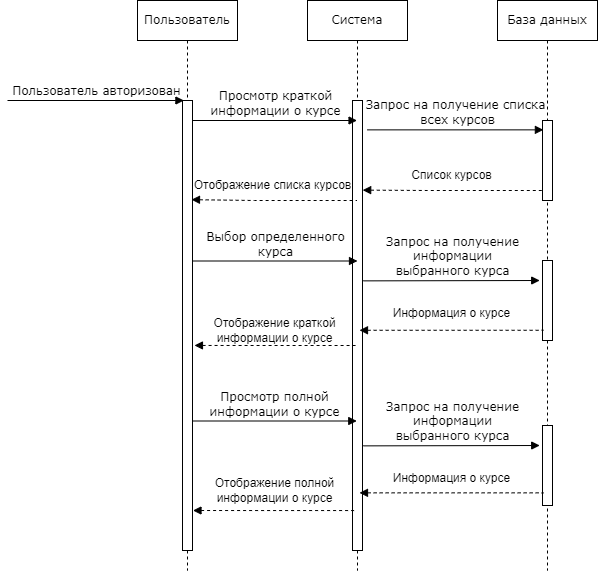


Рисунок Е.1 – Диаграмма взаимодействия

ПРИЛОЖЕНИЕ Ж

**(обязательное)**

**Листинг программы**

from django.shortcuts import render, redirect  
from .models import \*  
from django.contrib.auth.models import User  
from django.conf import settings  
  
# Create your views here.  
  
  
def index(request):  
 courses = Courses.objects.all()  
 img = Image.objects.all()  
 subject = Subjects.objects.all()  
 auth = settings.MY\_GLOBAL\_VAR  
 context = {  
 'auth': auth,  
 'courses': courses,  
 'img': img,  
 'subject': subject  
 }  
 return render(request, 'main/index.html', context)  
  
  
def entrace(request):  
 auth = settings.MY\_GLOBAL\_VAR  
  
 if request.method == 'POST':  
 username = request.POST.get('username')  
 password = request.POST.get('password')  
  
 user = User.objects.filter(username=username, password=password)  
 if user:  
 settings.MY\_GLOBAL\_VAR = {'auth': 1}  
 auth = settings.MY\_GLOBAL\_VAR  
 return redirect('/', auth=auth)  
 else:  
 return redirect('/entrace')  
 # return render(request, 'main/index.html', auth)  
  
 return render(request, 'main/entrace.html', auth)  
  
  
def registration(request):  
 if request.method == 'POST':  
 username = request.POST.get('username')  
 password = request.POST.get('password')  
 repeatPassword = request.POST.get('repeatPassword')  
  
 user = User.objects.filter(username=username, password=password)  
  
 if not user:  
 if password != repeatPassword:  
 return redirect('/registration')  
 else:  
 user = User(username=username, password=password)  
 user.save()  
 settings.MY\_GLOBAL\_VAR = {'auth': 1}  
 auth = settings.MY\_GLOBAL\_VAR  
 return redirect('/', auth=auth)  
 else:  
 return redirect('/registration')  
  
 return render(request, 'main/registration.html')  
  
  
def exit(request):  
 return render(request, 'main/exit.html', {'auth' : 1})  
  
  
def logout(request):  
 settings.MY\_GLOBAL\_VAR = {'auth': 2}  
 auth = settings.MY\_GLOBAL\_VAR  
 return redirect('/', auth=auth)  
 # return render(request, 'main/index.html', auth)  
  
  
def courseDetail(request, course\_id):  
 course = Courses.objects.get(id=course\_id)  
  
 lectures = Lecture.objects.filter(courses\_\_id=course\_id)  
 # lecturesInfo = Lecture.objects.filter(id=lectures.idLecture)  
  
 # lessons = LessonsLectures.objects.filter(lectures\_\_id=lectures.id)  
 # # lessonsInfo = Lessons.objects.filter(id=lessons.idLessons)  
 #  
 # subject = Subjects.objects.filter(id=course.idSubject)  
 # img = Image.objects.filter(id=subject.idImage)  
 context = { 'course': course,  
 'auth': settings.MY\_GLOBAL\_VAR,  
 'lectures': lectures,  
 }  
 return render(request, 'main/courseDetail.html', context)  
  
def full\_course(request, course\_id):  
 if settings.MY\_GLOBAL\_VAR == {'auth': 2}:  
 # доделать с ошибкой  
 return redirect('/entrace')  
 course = Courses.objects.get(id=course\_id)  
  
 lectures = Lecture.objects.filter(courses\_\_id=course\_id)  
 context = {'course': course,  
 'auth': settings.MY\_GLOBAL\_VAR,  
 'lectures': lectures,  
 }  
 return render(request, 'main/full\_course.html', context)