Учреждение образования
«Гродненский государственный политехнический колледж»

ОТЧЁТ
ПО
ТЕХНИКА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Учащегося 3 курса, группы ПЗТ-40

специальности 2 - 40 01 01 «Программное обеспечение информационных технологий»

Тема проекта: «Разработка Интернет-ресурса «EducatITon»

Ссылка на проект: https://github.com/DanNikonovich/EducatITon.git

Данные для входа: email: ggpkspps@gmail.com,
пароль: ggpkspps232151

Ссылка на UX-прототипы:https://www.figma.com/design/9UyXwON5gV76DD2sCTOmvY/EdocatITon?node-id=116-39&t=IlA6SJmG5QPPfobN-1

Ссылка на UI-прототипы: https://www.figma.com/design/9UyXwON5gV76DD2sCTOmvY/EdocatITon?node-id=212-80&t=oHJjmSdgfHCcnLIY-1

Выполнил  Д.А. Никонович
(инициалы, фамилия)

Руководитель от колледжа _____ Е.В. Заяц
(инициалы, фамилия)

2024

Содержание

Введение

На учебной практике была поставлена задача, разработать электронное средство обучения на тему: «EducatiTon».

Цель учебной практики заключается в создании платформы для предоставления возможности обучаться различным курсам в сети Интернет у массового пользователя.

Создаваемое электронное средство будет рассчитано для любого рода пользователей, в особенности для студентов. Применить данные электронное средство смогут все люди, заинтересовавшиеся в предоставленных курсах на платформе.

Приведем описание разделов пояснительной записки.

Первый раздел «Анализ задачи» посвящается постановке задачи, выбору модели жизненного цикла программного обеспечения, а также в этом разделе представлена диаграмма вариантов использования. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе «Инструменты разработки» будет рассмотрена среда, в которой создается данный проект.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки интернет-ресурса. Здесь можно узнать об организации данных в контексте среды разработки. В данном разделе будет описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации.

«Реализация» – это третий раздел отчета, в котором описываются все элементы и объекты, которые будут использованы при реализации данного сайта.

Четвёртый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данной программы. Будут смоделированы все возможные действия пользователя при работе с web-ресурсом, начиная от входа на сайт заканчивая закрытием вкладки.

В разделе «Руководство пользователя» будет описано назначение, область применения, среда функционирования данного программного продукта.

В «Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств.

В «Списке использованных источников» приведен список используемых при разработке источников с информацией.

В приложениях к пояснительной записке будет приведены UX и UI – прототипы страниц сайта, UML - диаграммы и тест-кейсы.

В приложении А «Диаграмма вариантов использования».

В приложении Б «Структура сайта».

В приложении В «Прототипы».

В приложении Г «Модель данных».

В приложении Д «Функциональная модель».
В приложении Е «Диаграмма последовательности».
В приложении Ж «Диаграмма деятельности».
В приложении И «Диаграмма классов».
В приложении К «Диаграмма объектов».
В приложении Л «Листинг программы».
В приложении М «Тест-кейсы».

1 Анализ задачи

1.1 Постановка задачи

1.1.1 Организационно-экономическая сущность задачи

Наименование задачи: сайт “EducatiTon”

Цель разработки: Создать платформу в виде сайта для размещения и просмотра онлайн курсов с встроенным интерпретатором(компилятором) кода.

Назначение: Данный сайт разрабатывается для людей любого возраста, желающих пройти обучение онлайн или разместить свой курс на платформе.

Периодичность использования: При необходимости и-или желании, в порядке появления новой информации.

Источники и способы получения данных: Сторонние преподаватели/онлайн школы, курсы которых одобрили администраторами.

Обзор существующих аналогичных ПП: skillbox, html academy, xyz school, google class.

1.1.2 Функциональные требования

Описание перечня функций и задач, которые должен выполнять будущий сайт:

Гость:

1. Просмотр сайта
2. Поиск продуктов по названию
3. Выбор языка
4. Сортировка по названию, стоимости
5. Просмотр имеющихся на платформе курсов
6. Добавление курсов в корзину
7. Регистрация

Студент:

1. Все функции незарегистрированного пользователя
2. Возможность присоединиться к курсу
3. Способность писать комментарии под данными в курсе
4. Обращение в техподдержку
5. Внесение денежных средств
6. Получение сертификата
7. Авторизация

Преподаватель:

1. Все функции незарегистрированного пользователя
2. Заявка на размещение курса, курирование курса
3. Возможность отвечать на комментарии
4. Предоставление личных данных для связи вне курса
5. Обращение в техподдержку
6. Проверка домашних заданий/тестов
7. Снятие денежных средств
8. Отслеживание прогресса обучения
9. Выдача сертификата
10.Авторизация

Администратор:
1. Предоставление разрешения на размещение курса
2. Добавление/удаление/редактирование курсов
3.Работа с пользователями: ответы на вопросы, помощь с техническими проблемами, обратная связь и поддержка.
4.Авторизация

Подробности реализации структуры курса:
1. Курс включает в себя модули, представляющие из себя множество отдельных прикреплённых файлов, текстовых, мультимедийных материалов, практики и контроля знаний(уроков). Модули размещаются с левой части страницы в виде списка.
2. Контроль знаний представляет собой краткие тесты, задачи, за решение которых выставляется оценка.
3. Практика включает в себя проверяемые домашние задания, проверка осуществляется посредством онлайн компилятора/интерпретатора. Домашние задания размещаются в виде текстовой/мультимедийной информации снизу основных материалов модуля.
4. На основе контроля знаний/практики выставляются баллы, которые позже считаются в среднеарифметическую оценку за курс.

1.1.3 Эксплуатационные требования

Требования к применению: позволяет в качественном и удобном порядке получить информацию из курса/разместить курс на платформе и монетизировать его. Сайт был выбран из-за большей мобильности и, вследствие этого большего охвата аудитории.

Требования к реализации: Для реализации сайта должен использоваться frontend: html, css; backend: c#, javascript, а так же фреймворк asp.net[1][2].

Требования к надежности: система не должна долго тормозить, виснуть, выдавать ошибки, быть недоступной не дольше 10 часов подряд, позволять напрямую скачивать видео с платформы, обеспечивается минимальный уровень защиты материала от кражи. Обеспечение сохранности введённых личных данных пользователей.

Требования к интерфейсу: Сайт должен быть адаптивным к разным разрешениям экрана/устройствам. Минималистичный дизайн, включающий в себя пару разделов (курсы, личная страница, информация о курсе, страница курса, список заданий, служба поддержки).

Подробности требований к компилятору/интерпретатора:

1 компилятора/интерпретатор будет внедрён в сайт. Это будет сделано посредством выбора создателя курса.

2 в компилятора/интерпретаторе будут доступны только некоторые, самые популярные языки программирования.

3 вывод компилятора/интерпретатора ограничен: пользователю может быть возвращена информация об ошибке, результат выполнения или оповещение об успешном/частичном/провальном выполнении unit тестов.

4 доступ и полная работоспособность с мобильных и десктопных браузеров.

Подробности реализации страницы с курсом:

1 Теория

1.1 Каждый урок включает в себя краткое текстовое описание (тема, краткий конспект), видео или иное представление данных, понятное студенту.

1.2 Минимальное требование по качеству для видео - 1920*1080. Видео располагается в/после текста.

1.3 Каждый прикреплённый файл не должен представлять опасности для студента.

1.4 Каждый текстовый материал должен чётко излагать мысль, не содержать грубых грамматических ошибок.

2 Практика

2.1 Решение практических(домашних) заданий, выданных преподавателем.

Сдача заданий при помощи онлайн компилятора/интерпретатора или прикрепления файла с домашним заданием.

3 Контроль знаний

3.1 Может включать в себя простые текстовые вопросы с вариантами/без вариантов ответа.

3.2 Может включать в себя решение задач в компиляторе/интерпретаторе на оценку с заранее определённым количеством попыток.

4 Сертификация

В конце курса, после сдачи всех желаемых материалов пользователь в праве запросить сертификат, подтверждающий прохождение курса с его текущим баллом.

Входная информация

Для авторизации/регистрации: Имя пользователя, адрес электронной почты, пароль, тип учётной записи.

Для создания курса: видео, текст, прикреплённые файлы.

Выходная информация

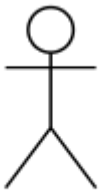
Обновление в базе данных пользователей при регистрации нового, обновление базы купленных курсов для каждого пользователя при покупке.

1.2 Диаграмма вариантов использования

Диаграмма вариантов использования — диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне[3].

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне (рисунок 1). Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.



Пользователь

Рисунок 1 – Графическое обозначение актёра

Вариант использования является стандартным языком UML и применяется для спецификации общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название (рисунок 2).

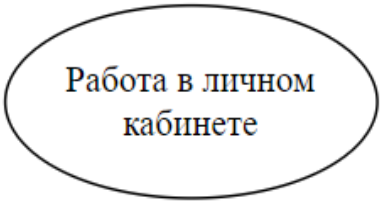


Рисунок 2 – Графическое обозначение варианта использования

Отношение ассоциации является главным понятием языка UML и используется при построении всех графических моделей. Оно служит для обозначения роли актера в отдельном варианте использования. На диаграмме отношение ассоциации обозначается сплошной линией между актером и вариантом использования (рисунок 3).

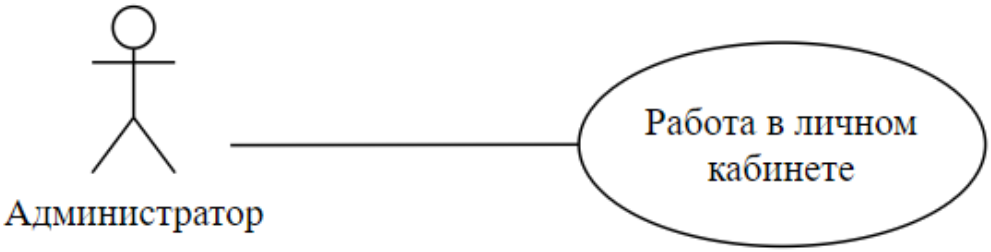


Рисунок 3 – Графическое обозначение отношения ассоциации

Для отображения взаимосвязи экземпляров отдельного варианта использования с общим вариантом, используется отношение расширения, обозначаемое направленный пунктирной линией со стрелкой от исходного варианта. Данная линия помечается ключевым словом <<extend>> (рисунок 4).

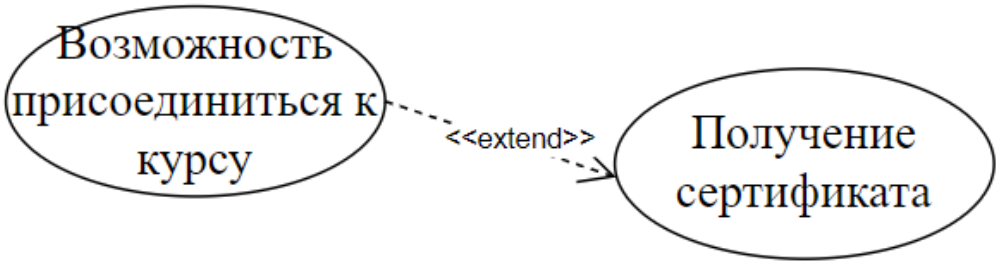


Рисунок 4 – Графическое обозначение отношения расширения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <<include>> (рисунок 5).

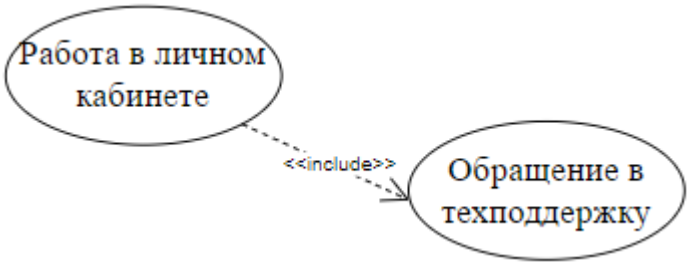


Рисунок 5 – Графическое обозначение отношения

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов использования, ее можно увидеть в Приложении А.

1.3 Выбор стратегии разработки и модели жизненного цикла

Обосновать выбор моделей ЖЦ для разработки проекта в соответствии с вашим индивидуальным заданием.

При выполнении задания использовать методику подбора модели ЖЦ, предложенную в пособии Бахтизин, В. В. Б30 Технология разработки программного обеспечения : учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск: БГУИР, 2010. – 267 с. : ил. ISBN 978-985-488-512-4 (предоставляется в электронном виде). Раздел 3 стр. 72-81[4].

Таблица 1 – Выбор стратегии разработки и модели жизненного цикла на основе характеристики требований

Критерии категории типов проекта и рисков	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
Являются ли требования к проекту легко определимыми и реализуемыми?	Да	Да	Да	Нет	Нет	Нет

Продолжение таблицы 1

1	2	3	4	5	6	7
Могут ли требования быть сформулированы в начале ЖЦ?	Да	Да	Да	Да	Нет	Нет
Часто ли будут изменяться требования на протяжении ЖЦ?	Нет	Нет	Нет	Нет	Да	Да
Нужно ли демонстрировать требования с целью их определения?	Нет	Нет	Да	Нет	Да	Да
Требуется ли проверка концепции программного средства или системы?	Нет	Нет	Да	Нет	Да	Да
Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ?	Нет	Нет	Нет	Да	Да	Да
Нужно ли реализовать основные требования на ранних этапах разработки?	Нет	Нет	Да	Да	Да	Да

Вычисления: Каскадная – 1, V-образная – 1, RAD – 2, Инкрементная – 4, Быстрого прототипирования – 6, Эволюционная – 6.

Итог: на основе результатов заполнения табл. 1 подходящей является Быстрого прототипирования и Эволюционная модель.

Таблица 2 – Выбор стратегии разработки и модели жизненного цикла на основе команды разработчиков

Критерии категории команды разработчиков проекта	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Нет	Нет	Да	Да
Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков?	Да	Да	Нет	Нет	Нет	Да
Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет	Нет	Нет	Да	Да	Да

Продолжение таблицы 2

1	2	3	4	5	6	7
Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	Да	Да	Нет	Да	Нет	Нет
Важна ли легкость распределения человеческих ресурсов проекта?	Да	Да	Да	Да	Нет	Нет
Приемлет ли команда разработчиков оценки, проверки, стадии разработки?	Да	Да	Нет	Да	Да	Да

Вычисления: Каскадная – 2, V-образная – 2, RAD – 3, Инкрементная – 4, Быстрого прототипирования – 5, Эволюционная – 4.

Итог: на основе результатов заполнения табл. 2 подходящими являются Быстрого прототипирования модели.

Таблица 3 – Выбор стратегии разработки и модели жизненного цикла на основе характеристики коллектива пользователей

Критерии категории коллектива пользователей	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
Будет ли присутствие пользователей ограничено в ЖЦ разработки?	Да	Да	Нет	Да	Нет	Да
Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки?	Нет	Нет	Нет	Да	Да	Да
Будут ли пользователи вовлечены во все фазы ЖЦ разработки?	Нет	Нет	Да	Нет	Да	Нет
Будет ли заказчик отслеживать ход выполнения проекта?	Нет	Нет	Нет	Нет	Да	Да

Вычисления: Каскадная – 2, V-образная – 2, RAD – 0, Инкрементная – 3, Быстрого прототипирования – 2, Эволюционная – 4.

Итог: на основе результатов заполнения табл. 3 подходящей является Эволюционная модель.

Таблица 4 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

Критерии категории типов проекта и рисков	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
Разрабатывается ли в проекте продукт нового для организации направления?	Нет	Нет	Нет	Да	Да	Да
Будет ли проект являться расширением существующей системы?	Да	Да	Да	Да	Нет	Нет
Будет ли проект крупно- или среднемасштабным?	Нет	Нет	Нет	Да	Да	Да
Ожидается ли длительная эксплуатация продукта?	Да	Да	Нет	Да	Нет	Да
Необходим ли высокий уровень надежности продукта проекта?	Нет	Да	Нет	Да	Нет	Да
Предполагается ли эволюция продукта проекта в течение ЖЦ?	Нет	Нет	Нет	Да	Да	Да
Велика ли вероятность изменения системы (продукта) на этапе сопровождения?	Нет	Нет	Нет	Да	Да	Да
Является ли график сжатым?	Нет	Нет	Да	Да	Да	Да
Предполагается ли повторное использование компонентов?	Нет	Нет	Да	Да	Да	Да
Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет	Нет	Нет	Нет	Да	Да

Вычисления: Каскадная – 2, V-образная – 3, RAD – 3, Инкрементная – 9, Быстрого прототипирования – 7, Эволюционная – 9.

Итог: на основе результатов заполнения табл. 4 подходящей является Эволюционная модель.

По итогам четырёх таблиц вышло: Каскадная – 7, V-образная – 8, RAD – 7, Инкрементная – 20, Быстрого прототипирования – 19, Эволюционная – 22.

Таким образом для реализации выбранного проекта больше всего подходит Эволюционная модель.

2 Проектирование задачи

2.1 Разработка пользовательского интерфейса

Важным при выполнении проекта является организация диалога между пользователем и сайтом. Во многом это зависит от того, как программист разработает данный сайт, какие компоненты будут использованы и какие методы будут автоматизированы. Особое внимание следует уделить интерфейсу. Разработчик должен так организовать внешний вид своей программы, что бы пользователь понял, что от него требуется. Для организации эффектной работы пользователя нужно создать сайт данной предметной области, в которой все компоненты сайта будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя. Таким образом, для успешной работы всего проекта в целом следует обеспечить интуитивно понятный интерфейс с приятными цветами и шрифтами.

2.1.1 Структура сайта

В ходе разработки была спроектирована структура сайта – схема расположения страниц и разделов относительно друг друга. Она определяет, какие категории будут присутствовать на ресурсе и как они будут связаны между собой. Разработанная структура сайта расположена в Приложении Б на рисунке 1.

2.1.2 UX-прототипы пользовательского интерфейса

Прототип – это черновой вариант IT-продукта, на создание которого требуется меньше времени и профессиональных знаний, но по самому продукту можно перемещаться как по уже запущенному сайту или приложению.

Технически прототип выглядит как система страниц или экранов, соединённых общей логикой и дизайном.

UX – это функционал интерфейса.

Разработанные UX-прототипы пользовательского интерфейса представлены в приложении В.

Ссылка на Figma (макеты интерфейсов): https://www.figma.com/design/9UyXwON5gV76DD2sCTOmvY/EdocatITon?node-id=116-39&t=IlA6SJmG5QPPfobN-1[5].

2.1.3 UI-прототипы пользовательского интерфейса

UI – это пользовательский интерфейс (оформление сайта: сочетания цветов, шрифты, иконки и кнопки). UI – внешний вид интерфейса.

Разработанные UI-прототипы пользовательского интерфейса представлены в приложении В.

Ссылка на Figma (макеты интерфейсов): https://www.figma.com/design/9UyXwON5gV76DD2s CTOmvY/EdocatITon?node-id=212-80&t=oHJjmSdgfHCcnLIY-1[5].

2.2 Разработка UML-диаграмм

UML-диграммы нужны для создания «чертежей» программы, схем, которые показывают, как будет устроено программное обеспечение изнутри, то есть для проектирования. В данном проекте будет представлено 7 UML диаграммы: диаграмма вариантов использования (описана в разделе 1 Анализ задачи), модель данных, функциональная модель, диаграмма последовательности, диаграмма деятельности, диаграмма классов, диаграмма объектов [3].

2.2.1 Модель данных

Модель данных — это абстрактное представление структуры данных и взаимосвязей между ними, которое используется для организации и управления данными в информационных системах.

Цель использования модели данных заключается в упрощении понимания и анализа данных, обеспечении целостности и согласованности информации, а также оптимизации процессов хранения и извлечения данных. Кроме того, модели данных служат основой для разработки баз данных и помогают в коммуникации между участниками проектов, включая разработчиков, аналитиков и бизнес-пользователей.

Модель данных представлена в Приложении Г на рисунке 1.

2.2.2 Функциональная модель

Функциональная модель – это модель, которая показывает, какие функции у проектируемой модели и как они взаимодействуют между собой.

Для построения функциональной модели предназначена методология функционального моделирования DFD.

На диаграмме отображен процесс входа на сайт.

Функциональная модель представлена в Приложении Д на рисунке 1.

2.2.3 Диаграмма последовательности

Диаграмма последовательности — это тип UML-диаграммы, который иллюстрирует взаимодействие между объектами в системе по времени, показывая порядок сообщений, которые они обмениваются.

Целью использования диаграммы последовательности является визуализация динамики системы, что помогает разработчикам понять, как объекты взаимодействуют друг с другом в различных сценариях. Она также способствует выявлению потенциальных проблем в логике взаимодействия и упрощает процесс проектирования, обеспечивая ясное представление о последовательности действий. Кроме того, диаграммы последовательности могут служить документацией для будущих изменений и улучшений в системе.

Диаграмма последовательности представлена в Приложении Е на рисунке 1.

2.2.4 Диаграмма деятельности

Диаграмма деятельности — это инструмент визуального моделирования, созданный на UML (унифицированном языке моделирования) для представления последовательности действий или управления потоками в системе. Она используется как выполнение процессов или сценариев использования, отображение переходов между действиями, альтернативные пути, параллельные действия и условия. Основное назначение диаграмм деятельности — описание поведения системы, процессов в бизнесе или алгоритмов, помогая анализировать и проектировать функциональность. Диаграмма включает в себя такие элементы, как действие, переходы, состояния, начало, завершение и узлы решений. Она помогает разработчикам, аналитикам и пользователям понять, как данные или задачи перемещаются в рамках системы или процесса.

Диаграмма деятельности представлена в Приложении Ж на рисунке 1.

2.2.5 Диаграмма классов

Диаграмма классов — это структурный инструмент визуального моделирования в UML, который описывает статическую структуру системы, отображая классы, их свойства, методы и отношения между ними. Она используется для проектирования и анализа объектно-ориентированных систем, обеспечивая детализированный взгляд на компоненты систем и их взаимодействие. Диаграмма помогает понять архитектуру приложений, определяя, какие классы существуют, как они взаимодействуют и как взаимодействуют. Основными элементами диаграмм являются классы, связи, агрегаты, состав, зависимости и направления. Диаграмма классов поставщиков программного обеспечения, упрощающая кодирование и связь между разработчиками.

Диаграмма классов представлена в Приложении И на рисунке 1.

2.2.6 Диаграмма объектов

Диаграмма объектов — это вид диаграммы UML, который отображает экземпляры классов (объектов) и их связи в конкретный момент времени. Она является статическим представлением системы, показывающим состояние объектов и их влияние на определенный этап выполнения программы. Основное назначение диаграммы объектов — проиллюстрировать структуру данных системы в динамике, указать, как объекты взаимодействуют в рамках определенного сценария или состояния. Диаграмма объектов полезна для анализа и проектирования, помогает визуализировать и уточнять связи между объектами и их атрибутами. Это делает ее эффективной для документирования системы, устранения ошибок и понимания ее работы.

Диаграмма объектов представлена в Приложении К на рисунке 1.

3 Реализация

3.1 Руководство программиста

3.1.1 Организация данных

Так же в проекте имеется база данных MySQL, содержащая в себе две таблицы Users и Courses. База данных имеет название userstoredb. Её содержимое представлено на рисунке 6.

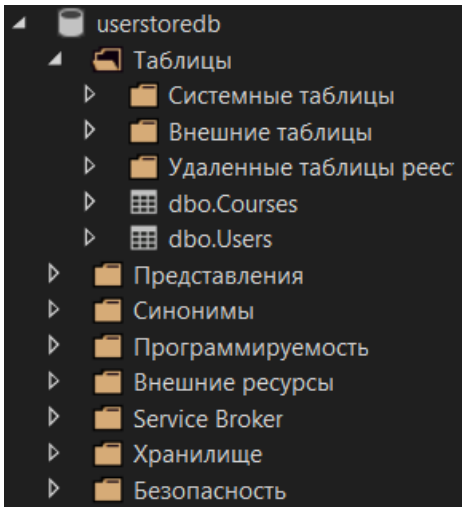


Рисунок 6 – Иерархия базы

Для управления базой данных, получения информации, а так же добавления информации и её обновления, использовался фреймворк Entity Framework.

В таблице Users описаны зарегистрированные пользователи, а в таблице Courses хранится информация о курсах.

Таблица 5 – Users

Атрибут	Тип
ID	INT
Name	NVARCHAR
Login	NVARCHAR
Password	NVARCHAR
Role	INT
WasInMenu	BIT

Таблица 6 – Courses

Атрибут	Тип
ID	INT
NameRu	NVARCHAR

Продолжение таблицы 6

NameEn	NVARCHAR
DescriptionRu	NVARCHAR
DescriptionEn	NVARCHAR
Price	NVARCHAR
Type	INT
ImgPath	NVARCHAR

3.1.2 Архитектура MVC – проекта

Программный продукт разработан с использованием фреймворка на языка C# ASP.Net core. Этот фреймворк позволяет создавать сайты, используя многие модели архитектур, в том числе и MVC, на которой написан данный программный продукт. Для начала работы с ASP.Net core необходимо установить необходимые компоненты, такие как Visual Studio Installer, Visual Studio, а так же компонент ASP.Net и разработка приложений [1][2]. После этого можно создать проект с необходимой архитектурой и разрабатывать его.

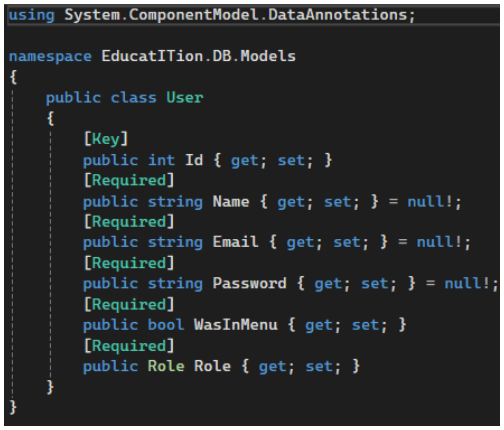
MVC (Model-View-Controller) — это архитектурный паттерн, используемый в разработке программного обеспечения, особенно в веб-приложениях. Он разделяет приложение на модели, представления и контроллеры.

Модель (Model): Этот компонент отвечает за управление данными и бизнес-логикой приложения. Модель взаимодействует с базой данных, обрабатывает данные и отправляет их обратно контроллеру или представлению.

Представление (View): Это компонент, который отвечает за отображение данных пользователю. Представление получает данные от модели и отображает их в удобном для пользователя виде. Оно также может включать в себя элементы интерфейса.

Контроллер (Controller): Контроллер служит связующим звеном между моделью и представлением. Он обрабатывает пользовательские вводы (например, нажатия кнопок, заполнение форм), взаимодействует с моделью для получения или изменения данных и обновляет представление.

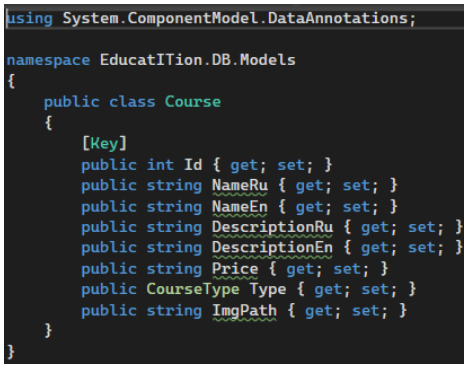
В проекте описаны 2 модели для более удобной связи с базой данных. Это Course и User, таблицы которых описаны выше. Они представлены на рисунках 7 и 8.



```
using System.ComponentModel.DataAnnotations;

namespace EducatITion.DB.Models
{
    public class User
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; } = null!;
        [Required]
        public string Email { get; set; } = null!;
        [Required]
        public string Password { get; set; } = null!;
        [Required]
        public bool WasInMenu { get; set; }
        [Required]
        public Role Role { get; set; }
    }
}
```

Рисунок 7 – Модель данных пользователя



```
using System.ComponentModel.DataAnnotations;

namespace EducatITion.DB.Models
{
    public class Course
    {
        [Key]
        public int Id { get; set; }
        public string NameRu { get; set; }
        public string NameEn { get; set; }
        public string DescriptionRu { get; set; }
        public string DescriptionEn { get; set; }
        public string Price { get; set; }
        public CourseType Type { get; set; }
        public string ImgPath { get; set; }
    }
}
```

Рисунок 8 – Модель данных курса

Так же описаны модели для передачи информации в представления. Например: CombinedCatalogueModel, CombinedIndexModel, ChoiceModel, SearchModel, AuthModel, RegModel, BaseViewModel, CatalogueViewModel, IndexViewModel и MenuViewModel.

MenuViewModel, CatalogueViewModel, IndexViewModel отвечают за передачу локализованной информацию в соответствующие им представления.

BaseViewModel является базовым классом, помогающим описать MenuViewModel, CatalogueViewModel и IndexViewModel.

AuthModel, RegModel нужны для передачи вводимых пользователем данных на сервер и последующей их обработки. AuthModel используется для формы авторизации, а RegModel для формы регистрации, находящихся на главной странице.

SearchModel используется для передачи пользовательского текста, по которому требуется найти и отобразитьь курсы в представлении catalogue.cshtml.

CombinedCatalogueModel, CombinedIndexModel представляют из себя наборы более простых моделей, которые одновременно должны передаваться в представляения. CombinedCatalogueModel включает в себя SearchModel и CatalogueViewModel, а CombinedIndexModel включает в себя IndexViewModel, AuthModel и RegModel.

Имеется 3 представления: index.cshtml, menu.cshtml, catalogue.cshtml. они описаны на языках C# + html. Index отвечает за главную страницу, menu за меню, а catalogue за каталог.

Пример формы регистрации, находящейся на главной страницу, представлен на рисунке 9.

```
<div id="reg-form" class="modal" data-width="45%" data-height="75%">
    <div class="modal-content modal-s vertical-sb-layout">
        <form method="post" class="form-container">
            <img src="/imgs/icons/cross.png" id="reg-close-cross" class="closeModal cross-img">
            <div class="form-els-container">
                <div class="logo-container">
                    <img src="/imgs/icons/logo.png" class="logo">
                    <span class="logo-text">EducatITion</span>
                    <span class="description-text">-WEBSITE WITH IT COURSES-</span>
                </div>
                <div class='form-title'><b>@Model.IndexViewModel.RegLikeText</b></div>
                <div class="reg-inputs-container">
                    <input asp-for="RegModel.Name" type="text" class="user-input" placeholder="@Model.IndexViewModel.PlaceholdersNames["name"]">
                    <input asp-for="RegModel.Email" type="text" class="user-input" placeholder="@Model.IndexViewModel.PlaceholdersNames["login"]">
                    <input asp-for="RegModel.Password" type="text" class="user-input" placeholder="@Model.IndexViewModel.PlaceholdersNames["password"]">
                    <input asp-for="RegModel.RepeatPassword" type="text" class="user-input" placeholder="@Model.IndexViewModel.PlaceholdersNames["repeat password"]">
                </div>
                <button class='continue-btn'>@Model.IndexViewModel.ContinueText</button>
                <span class='reg-agreement-text'>@Model.IndexViewModel.AgreementText</span>
                <button type="button" class='reg-enter-ref' onclick="switchForms(this)">@Model.IndexViewModel.EnterText</button>
            </div>
        </form>
    </div>
</div>
<script src="~/js/modalScript.js"></script>
<script src="~/js/indFormsSwitcher.js"></script>
<script src="~/js/indModalOpener.js"></script>
<script src="~/js/saveUserType.js"></script>
body>
html>
```

Рисунок 9 – Модель данных пользователя

Модели и представления связываются при помощи одного контроллера HomeController, в котором описана логика отклика приложения на запросы.

Пример обработки запроса Get и Post для методов, работающих с представлением index.cshtml представлен на рисунке 10.

```
[HttpGet]
public IActionResult Index(string localization, string direction)
{
    if (localization != null)
        session.SetString("localization", localization);
    if (direction == null)
        direction = "0";

    int lastInd = session.GetInt32("selectedCourseInd") ?? 0;
    session.SetInt32("selectedCourseInd", lastInd + int.Parse(direction));

    var viewModel = new CombinedIndexModel()
    {
        RegModel = new RegModel(),
        AuthModel = new AuthModel(),
        IndexViewModel = new IndexViewModel(session)
    };

    return View(viewModel);
}

[HttpPost]
public IActionResult Index(CombinedIndexModel model)
{
    RegUser(model.RegModel);
    AuthUser(model.AuthModel);

    model = new CombinedIndexModel()
    {
        RegModel = new RegModel(),
        AuthModel = new AuthModel(),
        IndexViewModel = new IndexViewModel(session)
    };

    return View(model);
}
```

Рисунок 10 – Обработка запросов

3.1.3 Спецификация проекта

Таблица 7 – Значение файлов

Имя файла	Назначение
1	2
1 HomeController.cs	Контроллер, отвечающий за все ответы на запросы к сайту
2 ApplicationContext.cs	Представление базы данных, нужно для взаимодействия с бд при помощи Entity Framework
3 Course.cs	Модель курса
4 User.cs	Модель пользователя
5 CombinedCatalogueModel.cs	Комбинированння модель для передачи в представляение каталога
6 CombinedIndexModel.cs	Комбинированння модель для передачи в представляение главной страницы
7 ChoiceModel.cs	Модель для выбора пользователя
8 SearchModel.cs	Модель для пользовательского ввода искомого курса
9 AuthModel.cs	Модель для авторизации
10 RegModel.cs	Модель для регистрации
11 BaseViewModel.cs	Базовый класс модели
12 CatalogueViewModel.cs	Локализированная модель данных для каталога
13 IndexViewModel.cs	Локализированная модель данных для главной страницы
14 MenuViewModel.cs	Локализированная модель данных для меню
15 catalogue.cshtml	Файл с кодом представления каталога
16 menu.cshtml	Файл с кодом представления меню
17 Index.cshtml	Файл с кодом представления главной страницы
18 EducatITion.sln	Файл решения Visual Studio 2022
19 Enums.cs	Код перечислений, которые используются в веб-приложении
20 Program.cs	Файл, в котором определена точка входа в программу
21 EducatITion.csproj.user	Пользовательский файл конфигурации проекта в среде разработки Visual Studio
22 EducatITion.csproj	Файл проекта, используемый в среде разработки .NET
23 appsettings.Development.json	Файл конфигурации в приложениях, разработанных на платформе .NET, который используется для хранения параметров настройки, специфичных для среды разработки

Продолжение таблицы 7

24 appsettings.json	Файл конфигурации, используемый в приложениях на платформе .NET
25 EducatITion.exe	Исполняемый файл проекта
26 EducatITion.dll	Скомпилированный код проекта
27 *.dll	Скомпилированный код, сгенерированный системой и необходимый для запуска веб-приложения
28 EducatITion.pdb	Содержит отладочную информацию, такую как символы и данные о структуре программы
29 EducatITion.runtimeconfig.json	Содержит информацию о среде выполнения (runtime) и зависимостях приложения
30 EducatITion.staticwebassets.endpoints.json	Описывает конфигурацию статических веб-ресурсов, таких как CSS, JavaScript
31 EducatITion.staticwebassets.runtime.json	Содержит информацию о статических веб-ресурсах, таких как CSS, JavaScript
32 *.css	Файлы с описанием стилей отдельной страницы или её части
1	2
33 *.png	Файлы, храняций картинку
34* .js	Файлы с кодом на языке JavaScript
35 launchSettings.json	Файл конфигурации для определения параметров запуска приложения
36 Obj/*	Временные файлы проекта

Листинг всей программы представлен в Приложении Л

4 Тестирование

4.1 Тесты на использование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Разработанные тест-кейсы и статус их выполнения представлены в приложение М.

Расписание работ над проектом представлено в таблице 8.

Таблица 8 – Расписание работ над проектом

Имя	Дата	Деятельность	Продолжительность, ч
Никонович Даниил	21.01.2025	Разработка тестов	2
Никонович Даниил	06.01.2025	Тестирование web-сайта	3
Никонович Даниил	07.01.2025	Составление отчетов о найденных дефектах	3
Никонович Даниил	18.01.2025	Исправление найденных ошибок	1
Никонович Даниил	24.12.2024	Проведение регрессионного тестирования	2
Никонович Даниил	26.12.2024	Составление отчета о результатах тестирования	3

4.2 Отчёт о результатах тестирования

Статистика по всем дефектам представлена в таблице 9.

Таблица 9 – Статистика по всем дефектам

Статус	Количество	Важность			
		Низкая	Средняя	Высокая	Критическая
Найдено	1	1	0	0	0
Исправлено	1	1	0	0	0
Проверено	0	0	0	0	0
Открыто заново	0	0	0		0
Отклонено	0	0	0	0	0

По результатам тестирования все элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные

в процедурах. Так же было зафиксировано, что загрузка страниц сайта усложнена большим количеством изображений, из-за ограниченных возможностей конструктора, но это никак не повлияло на результативность программного продукта. Таким образом, программный продукт можно использовать, не испытывая особых проблем или неудобств, связанных с взаимодействием с программным продуктом [7].

5 Руководство пользователя

Целью данного проекта является создание программного средства «EducatiTon», который предоставит всем желающим, доступ ко всей необходимой информации в курсах.

5.1 Незарегистрированный пользователь

Для того, чтобы открыть сайт необходимо перейти по ссылке.

После открытия сайта загружается главная страница, представленная на рисунке 11. На главной странице располагается общая информация. Также незарегистрированный пользователь может воспользоваться боковыми кнопками для просмотра имеющихся курсов, и их описания, на платформе, изображённых на рисунках 12 и 13.

Рисунок 11 – Первичная страница

Рисунок 12 – Нажатие на кнопку справа

Рисунок 13 – Нажатие на кнопку слева

После нажатия на меню или кнопки «Войти»/«Зарегистрироваться» появляется пак-меню предлагающее пользователю выбрать свою роль в системе, из предоставленных на рисунке 14.

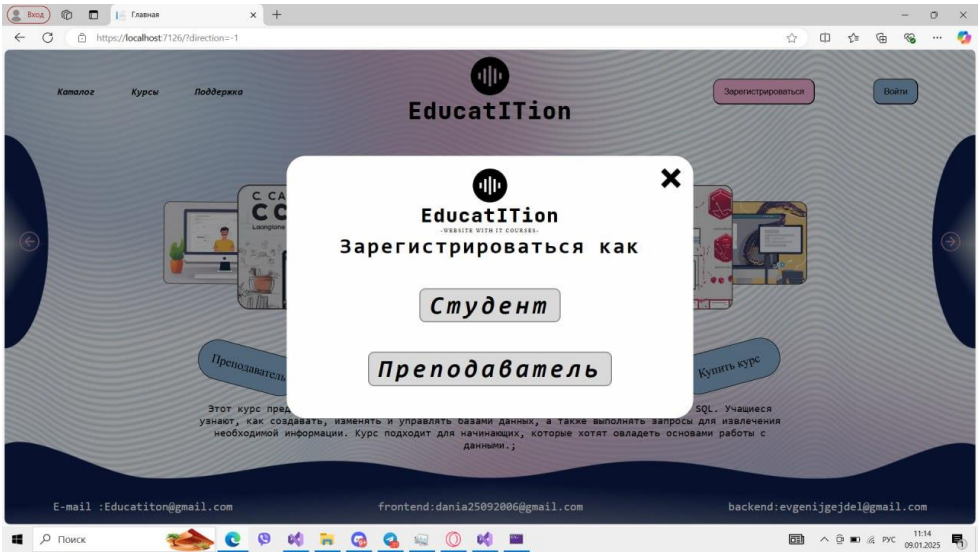


Рисунок 14 – Выбор роли в системе

После выбора пользователю предоставляется форма регистрации, предоставленная на рисунке 15, если же пользователь был однажды зарегистрирован, то он может перейти на форму вход, предоставленную на рисунке 16.

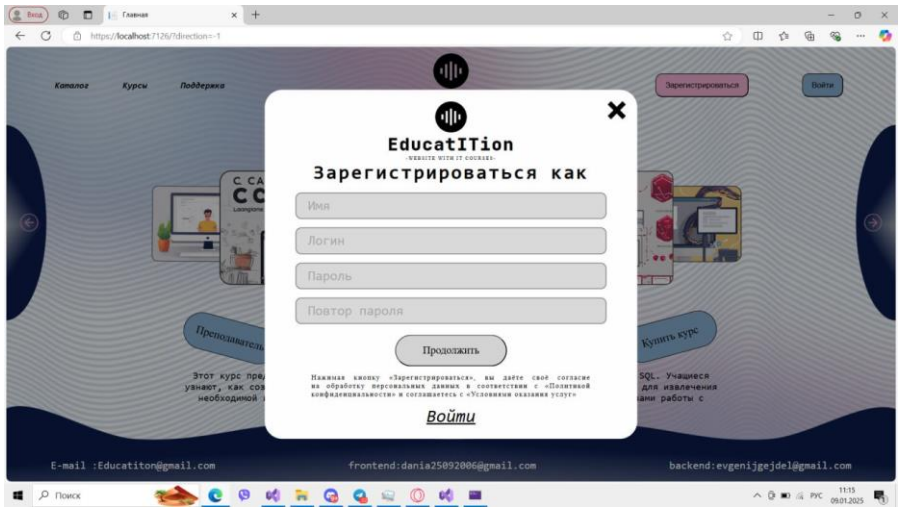


Рисунок 15 – Форма регистрации

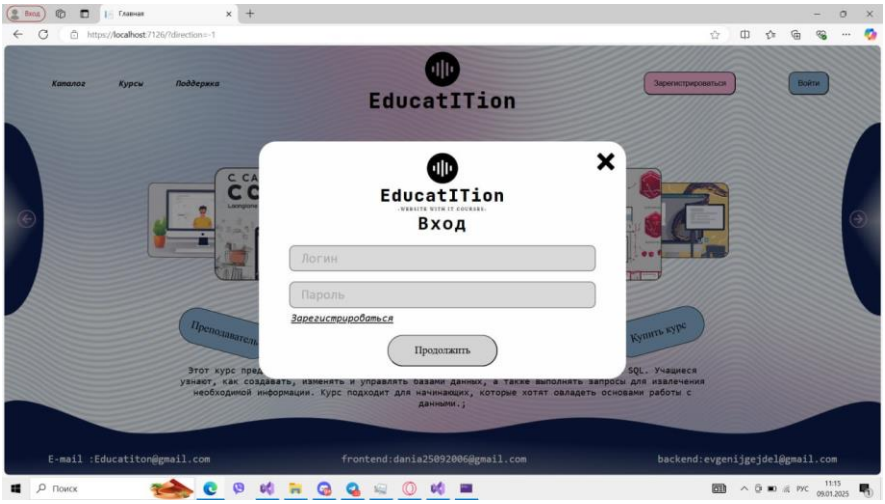


Рисунок 16 – Форма вход

5.2 Студент

Если попытка входа является неудачной и какие-либо вводимые данные были указаны неправильно, то пользователя возвращает на предыдущую страницу, показанную на рисунке 11, если же всё было сделано правильно, то пользователь переходит на главную страницу, показанную на рисунке 17. У студента по-прежнему остаётся возможность скроллинга имеющихся курсов на платформе.

Рисунок 17 – Главная страница

Весь сайт автоматически воспроизводит текст на русском языке, но есть студент захочет, то может нажать на кнопку «EN», после чего сайт переведётся на английский язык, как на рисунке 18, если же студент захочет вернуться назад, то может нажать на кнопку «RU».

Рисунок 18 – Нажатие на кнопку «EN»

При нажатии студентом на меню, ему открывается личный, в котором встречает приветственное сообщение, которое показывается только один раз, предоставлено на рисунке 19.

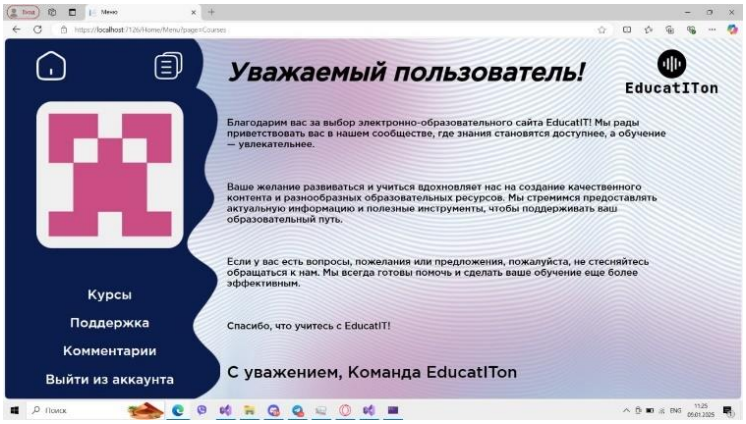


Рисунок 19 – Отображение приветственного сообщения

При нажатии студентом на листики с текстом, то студент переходит в каталог курсов, предоставленный на рисунке 20, также в каталог можно попасть с главной страницы нажав в меню на каталог.

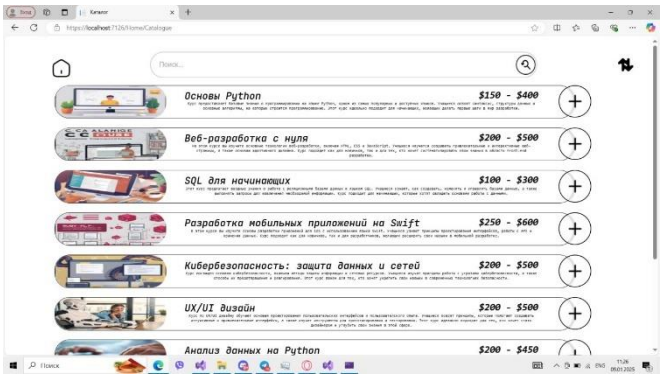


Рисунок 20 – Каталог курсов

Если студент начнёт вводить текст в поисковую строку, то ему будут отображаться все возможные подходящие результаты, показанные на рисунке 21, после нажатия на стрелки все курсы сортируются по алфавитному порядку, представленному на рисунке 22, если же пользователь нажмёт на домик, то он вернётся на главную страницу, показанную на рисунке 17.

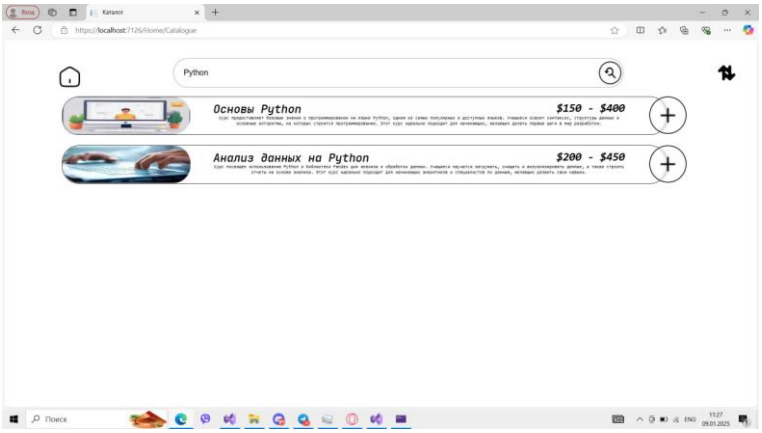


Рисунок 21 – Поиск курсов

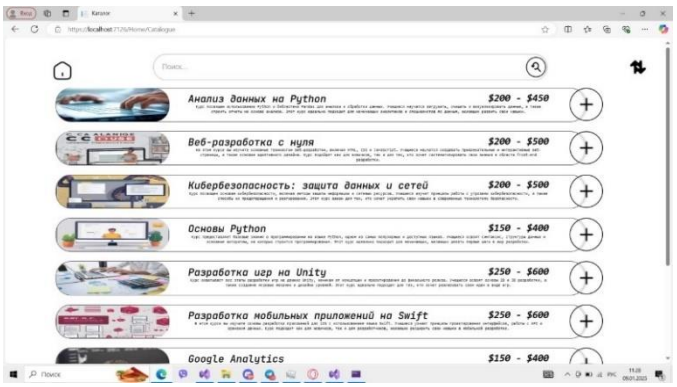


Рисунок 22 – Сортировка курсов по алфавиту

После нажатия на главной странице на кнопку «Курсы» сразу же отображаются все имеющиеся курсы студента у него в личном кабинете, показанный на рисунке 23.

Рисунок 23 – Курсы студента

Если студент захочет нажать на кнопки «Поддержка» или «Комментарии», то он переходит на соответствующие страницы сайта, изображённые на рисунках 24 и 25.

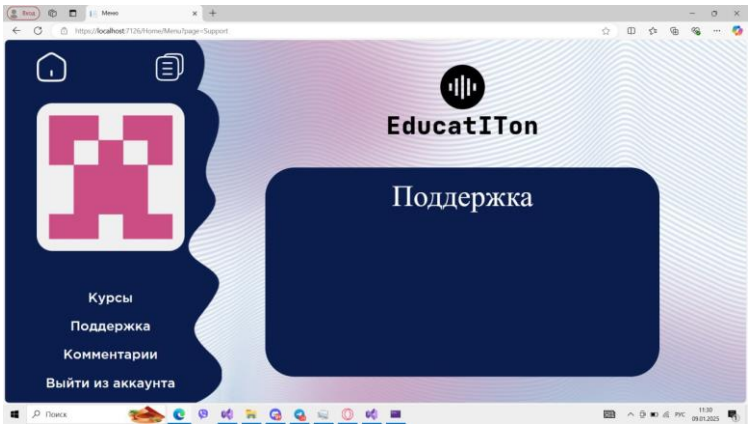


Рисунок 24 – Страница поддержка

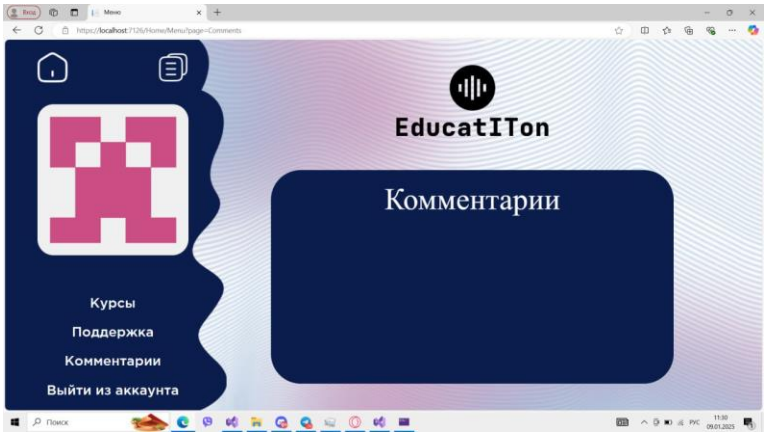


Рисунок 25 – Страница комментарии

При нажатии студентом на кнопку «Выйти из аккаунта», то ему отображается пак-меню, предоставленное на рисунке 26, если студент выберет «Остаться», то он перейдёт на страницу «Курсы» в личном кабинете, представленную на рисунке 23, но если студент выберет «Выйти», то он переместится на первичную страницу, изображённую на рисунке 11, но его данные и роль сохранятся.

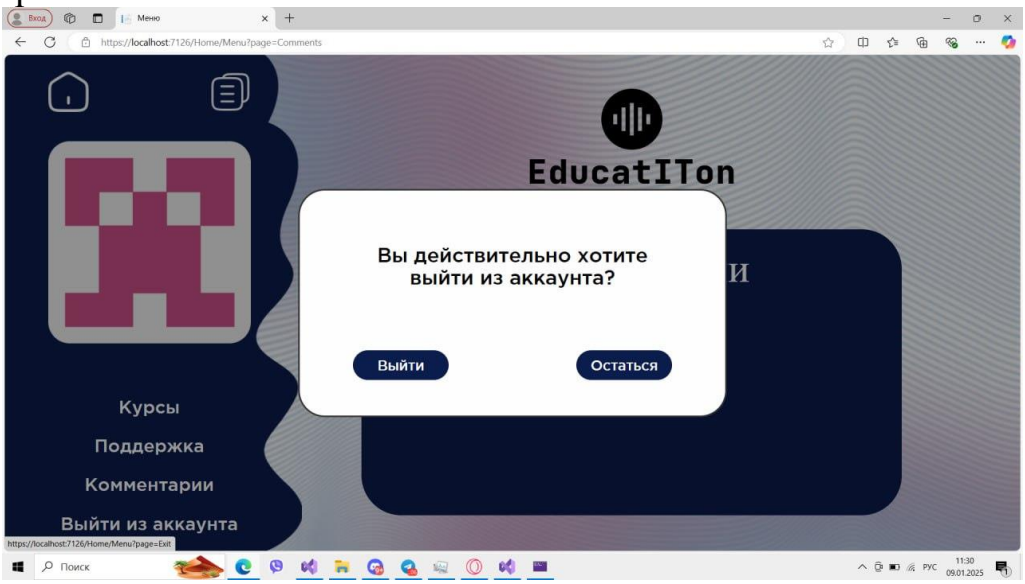


Рисунок 26 – Выход из аккаунта

5.3 Преподаватель

Если же пользователь зашёл в систему под ролью преподавателя, то ему отображается плюс, для создания своего курса, представленного на рисунке 27, все остальные функции остаются такими же, как и у студента.

Рисунок 27 – Курсы преподавателя, с возможностью создания курса

Заключение

Целью данного проекта является создание информационного сайта «EducatiTon», который позволит получить доступ ко всей необходимой информации для самостоятельного образования. Ресурс будет поддерживать различные форматы файлов, фотографии, видеоматериалы.

В ходе реализации поставленной задачи были закреплены знания по использованию фреймворков asp.net и Entity framework.

Поставленная задача выполнена в соответствии со всеми требованиями, созданы и протестированы все необходимые страницы и компоненты проекта.

В ходе тестирования все исключительные ситуации были обработаны. Проект работает без сбоев и ошибок. В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

Исходя из этого, можно сделать вывод, что программа реализована успешно, поставленные цели достигнуты.

Список использованных источников

1. C# для чайников [Электронный ресурс]. – Режим доступа: https://itproger.com/course/csharp– Дата доступа: 04.12.2024.

2. Обзор ASP.NET [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/overview– Дата доступа: 11.12.2024.

3. Типы UML-диаграмм, как их создать, примеры [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/uml-diagrammy/– Дата доступа: 11.11.2024.

4. Выбор стратегии жизненного цикла [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/448008/ – Дата доступа: 10.11.2024.

5. Разбор UI/UX на примере прототипа в Figma и основные принципы [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/508028/ – Дата доступа: 17.11.2024.

6. Руководство по MySQL [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/mysql/ – Дата доступа: 17.11.2024.

7. Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://gb.ru/blog/testirovanie-programmnogo-obespecheniya/ – Дата доступа: 17.11.2024.

Приложение А

Диаграмма вариантов использования

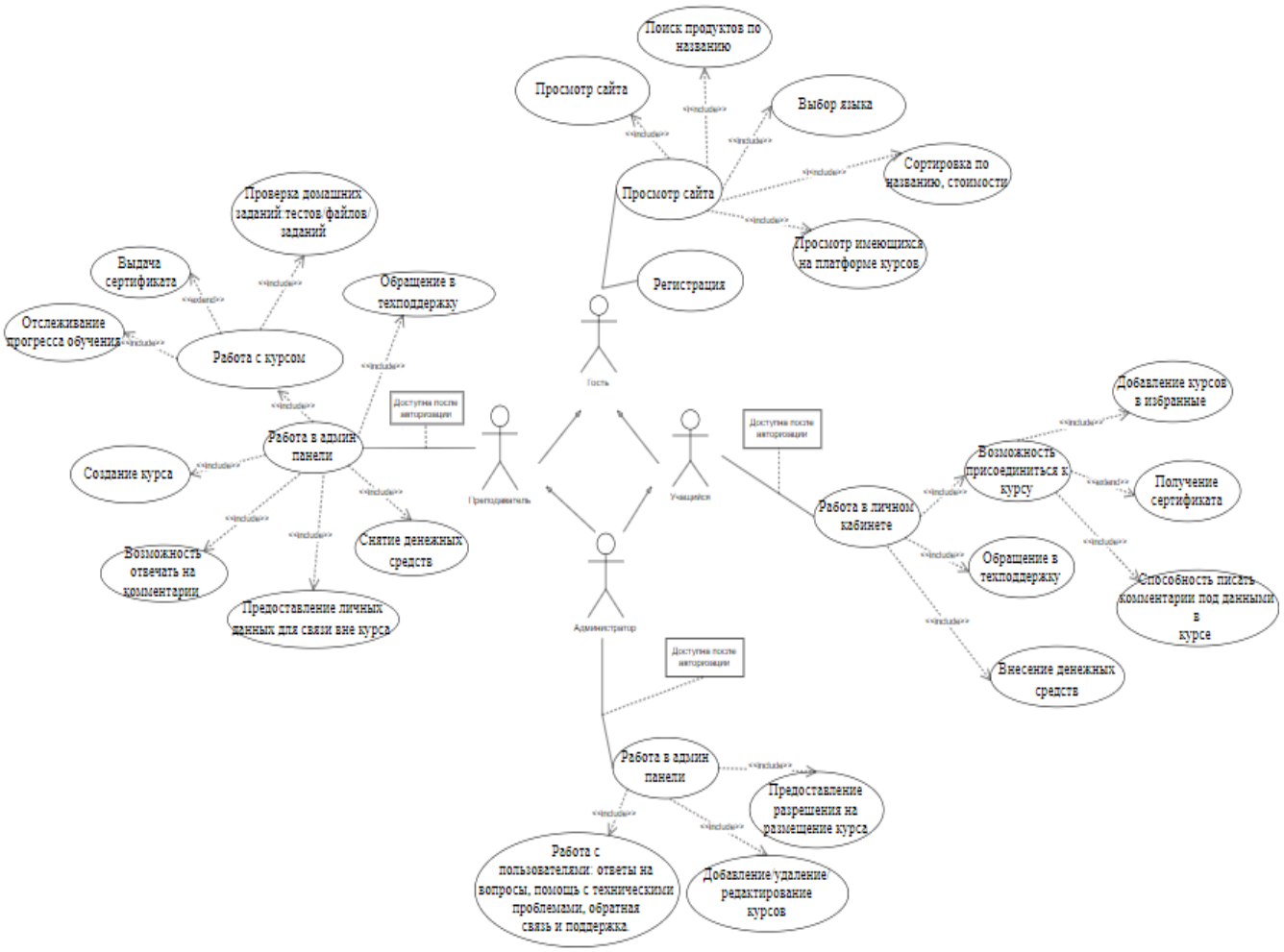


Рисунок А.1 — Диаграмма вариантов использования

Приложение Б
Структура сайта

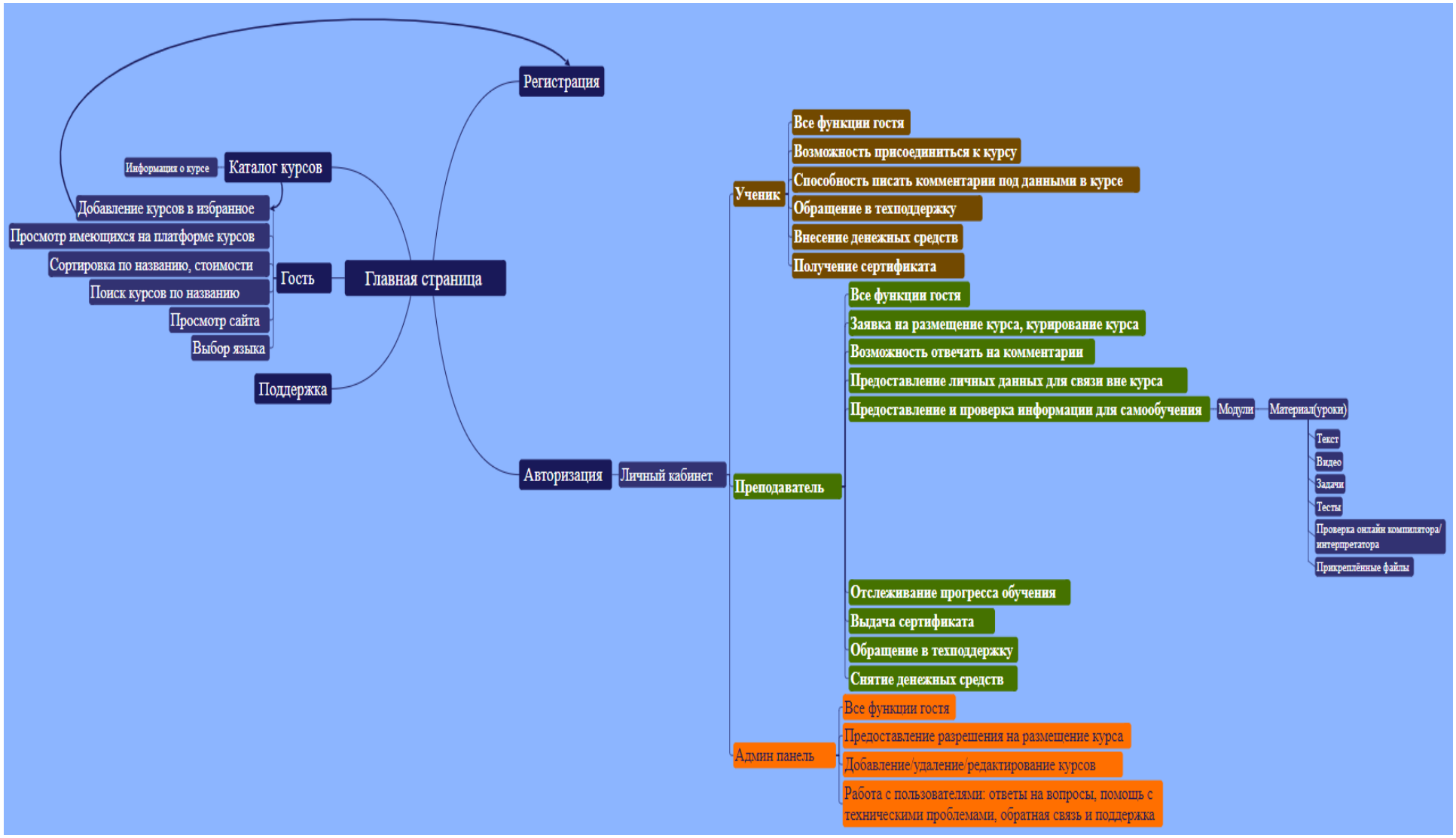


Рисунок Б.1 – Структура сайта

Приложение В
Прототипы

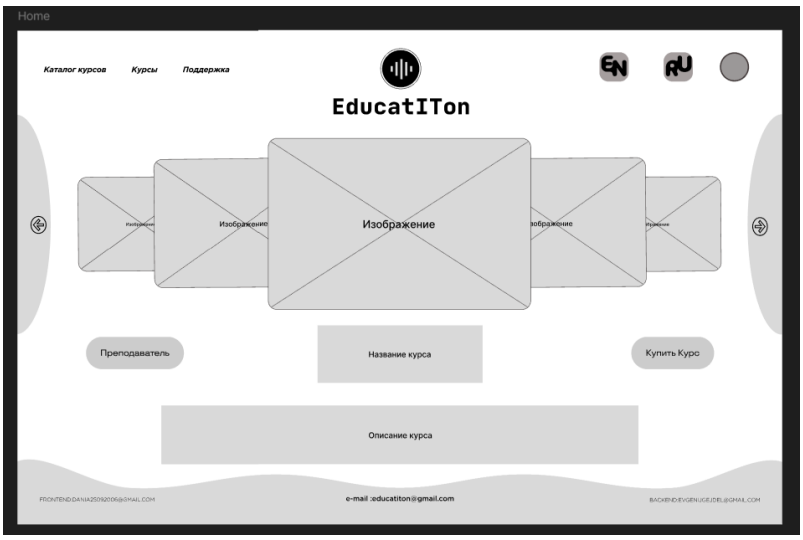


Рисунок В.1 – UX. Главная страница

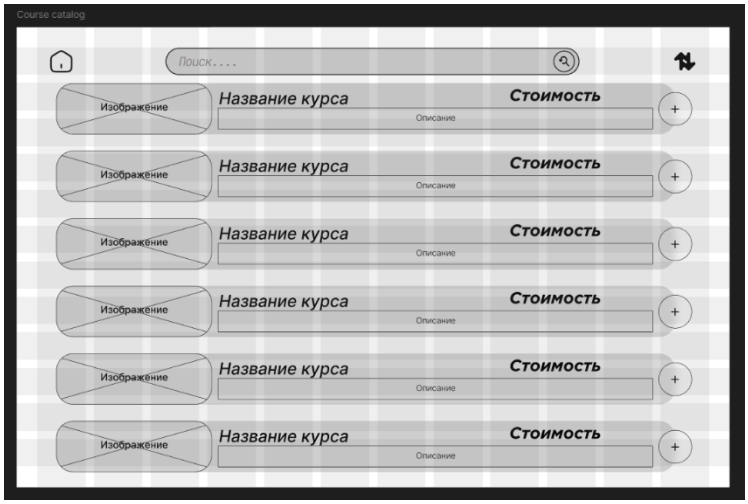


Рисунок В.2 – UX. Каталог курсов

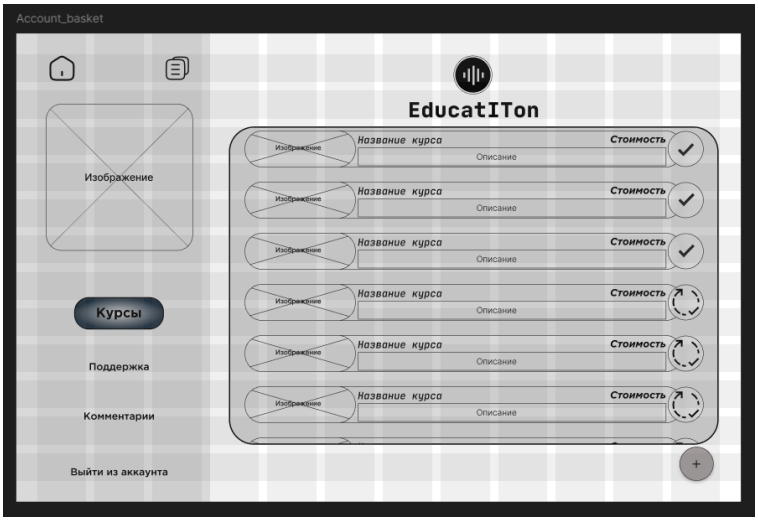


Рисунок В.3 – UX. Личный кабинет

Рисунок В.4 – UI. Главная страница

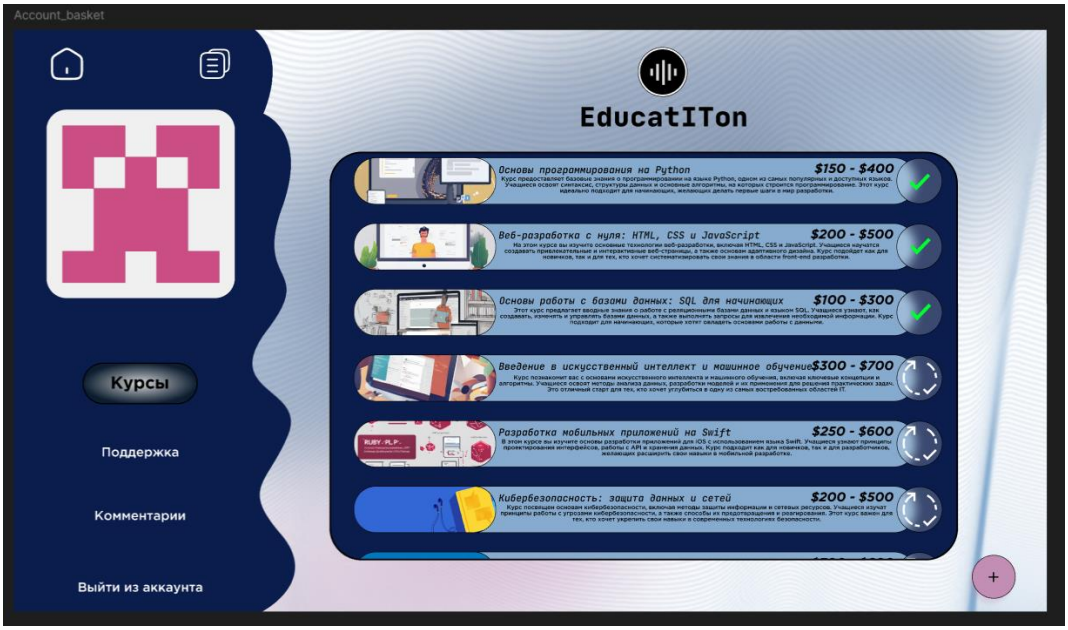


Рисунок В.5 – UI. Личный кабинет

Приложение Г
Модель данных

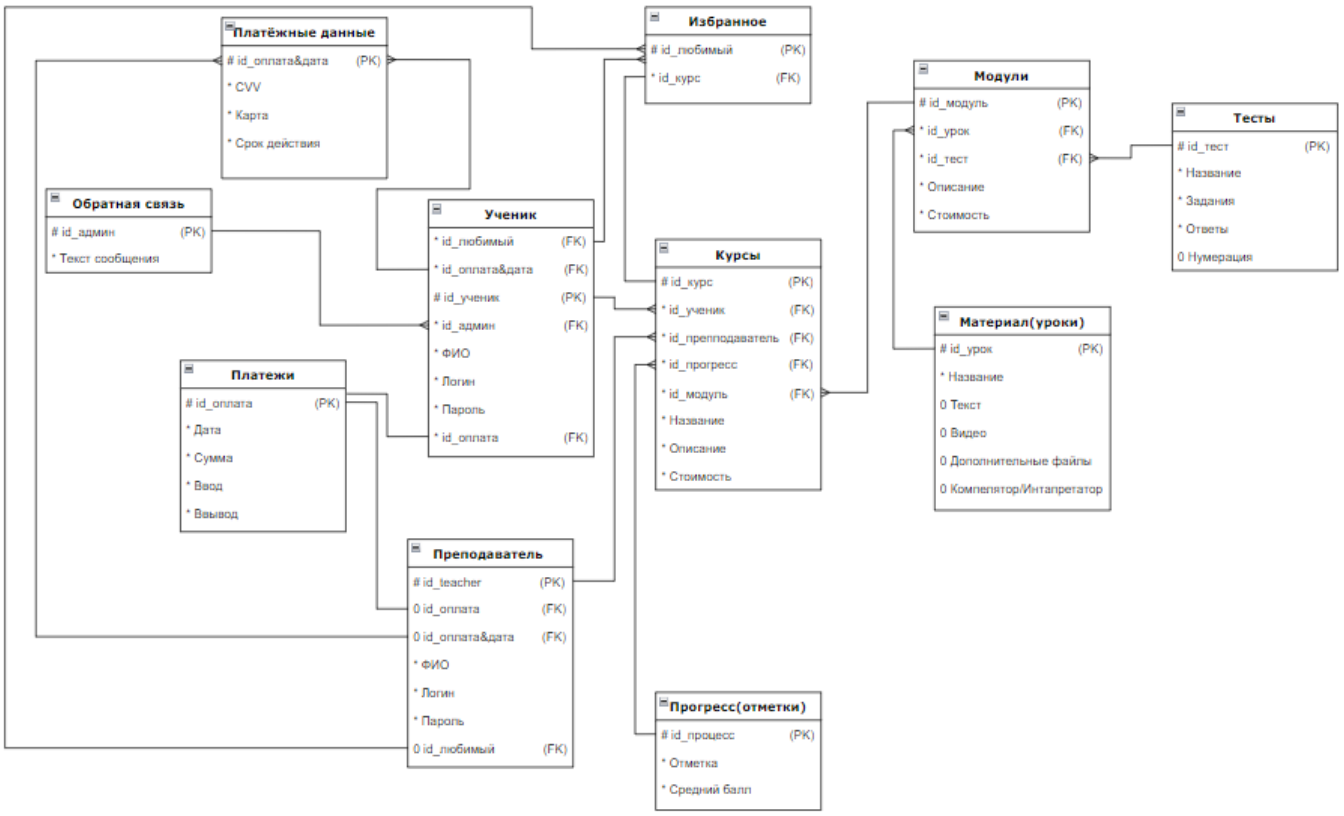


Рисунок Г.1 – Модель данных

Приложение Д
Функциональная модель

Рисунок Д.1 – Функциональная модель. Прохождение курса

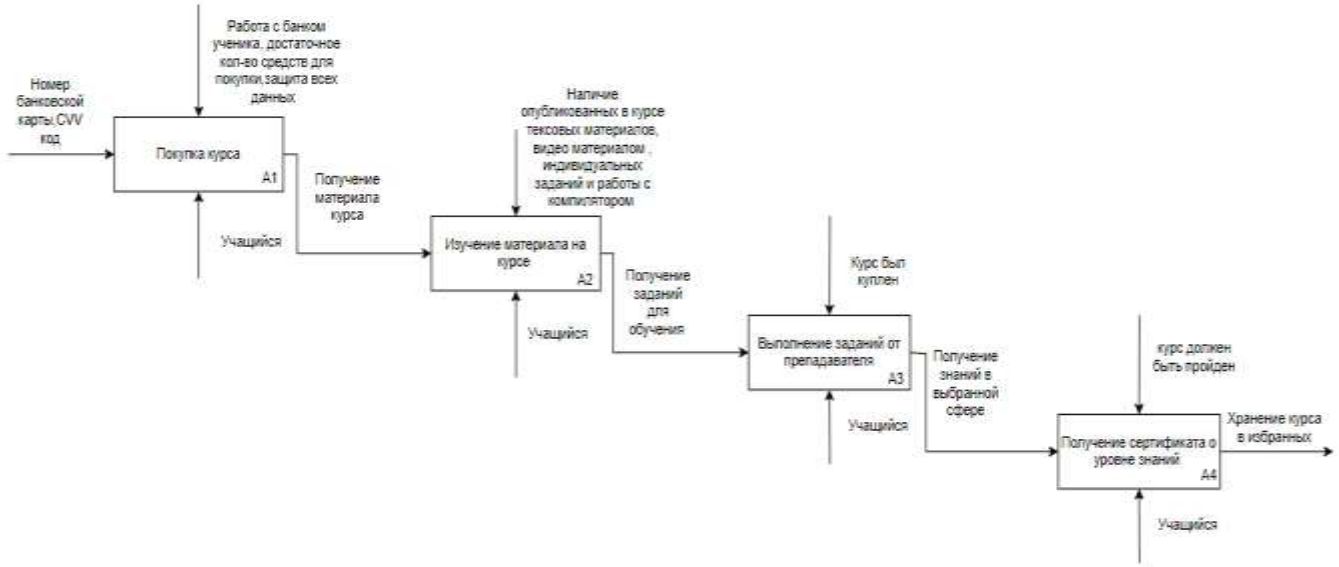


Рисунок Д.2 – Функциональная модель. Функции учащегося

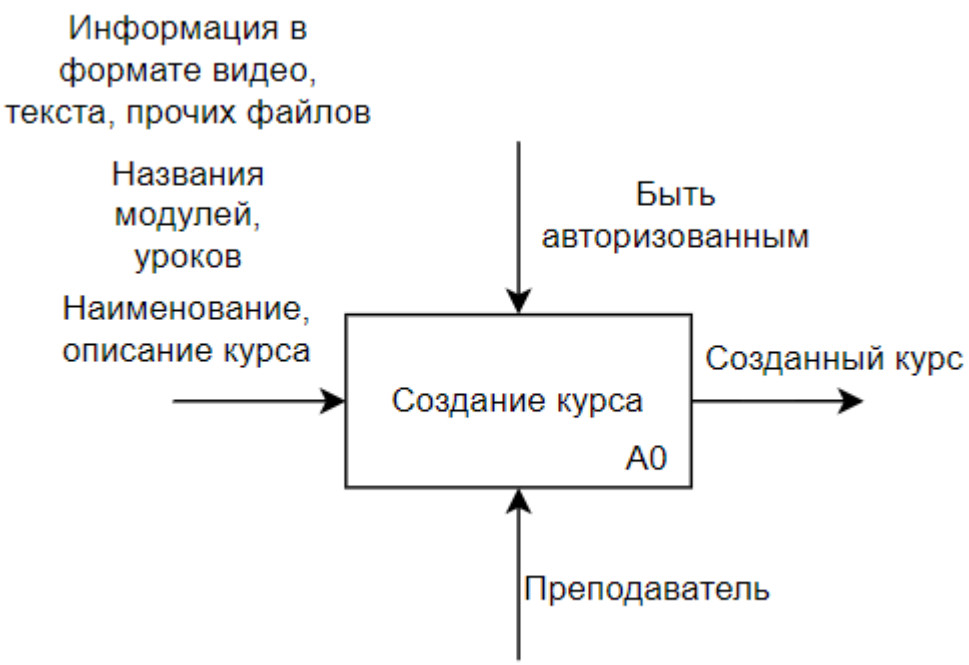


Рисунок Д.3 – Функциональная модель. Создание курса

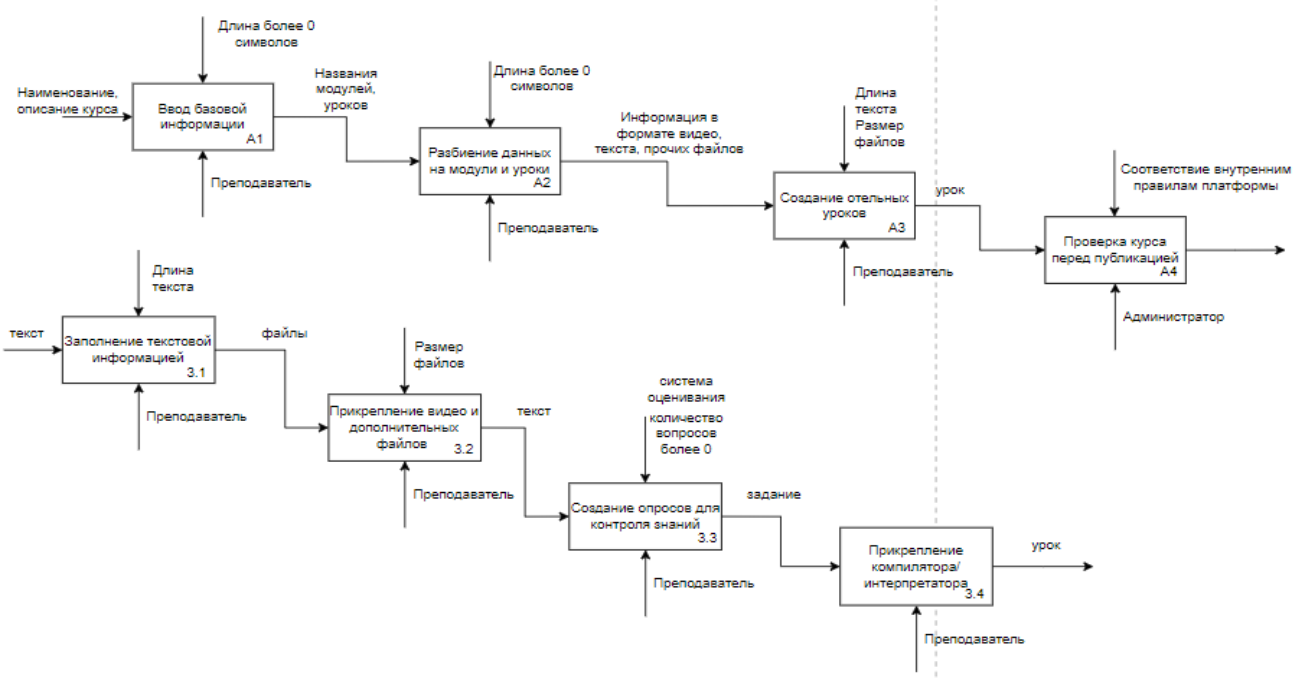


Рисунок Д.4 – Функциональная модель. Функции учащегося

Приложение Е

Диаграмма последовательности

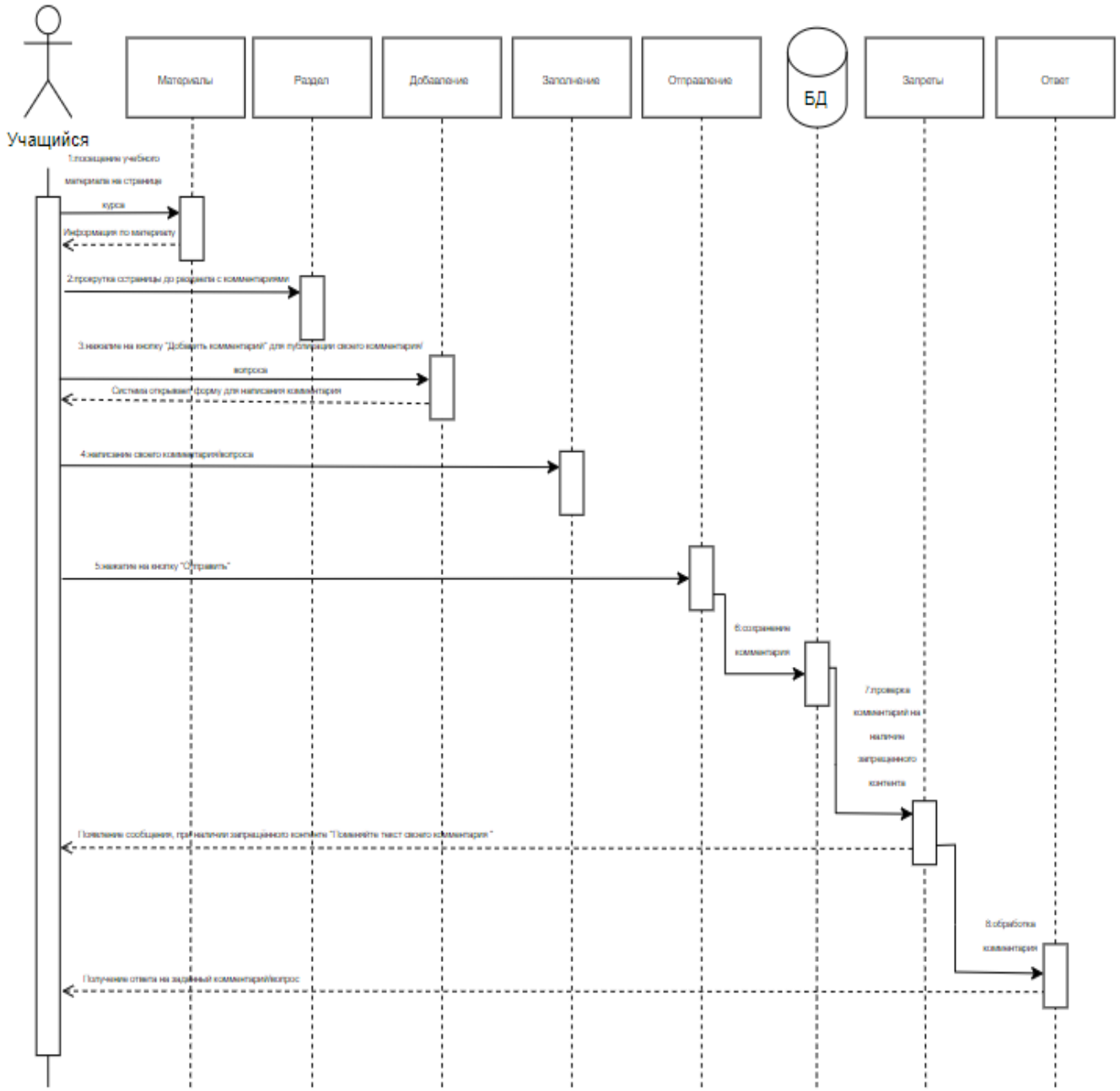


Рисунок Е.1 – Диаграмма последовательности. Работа с комментариями

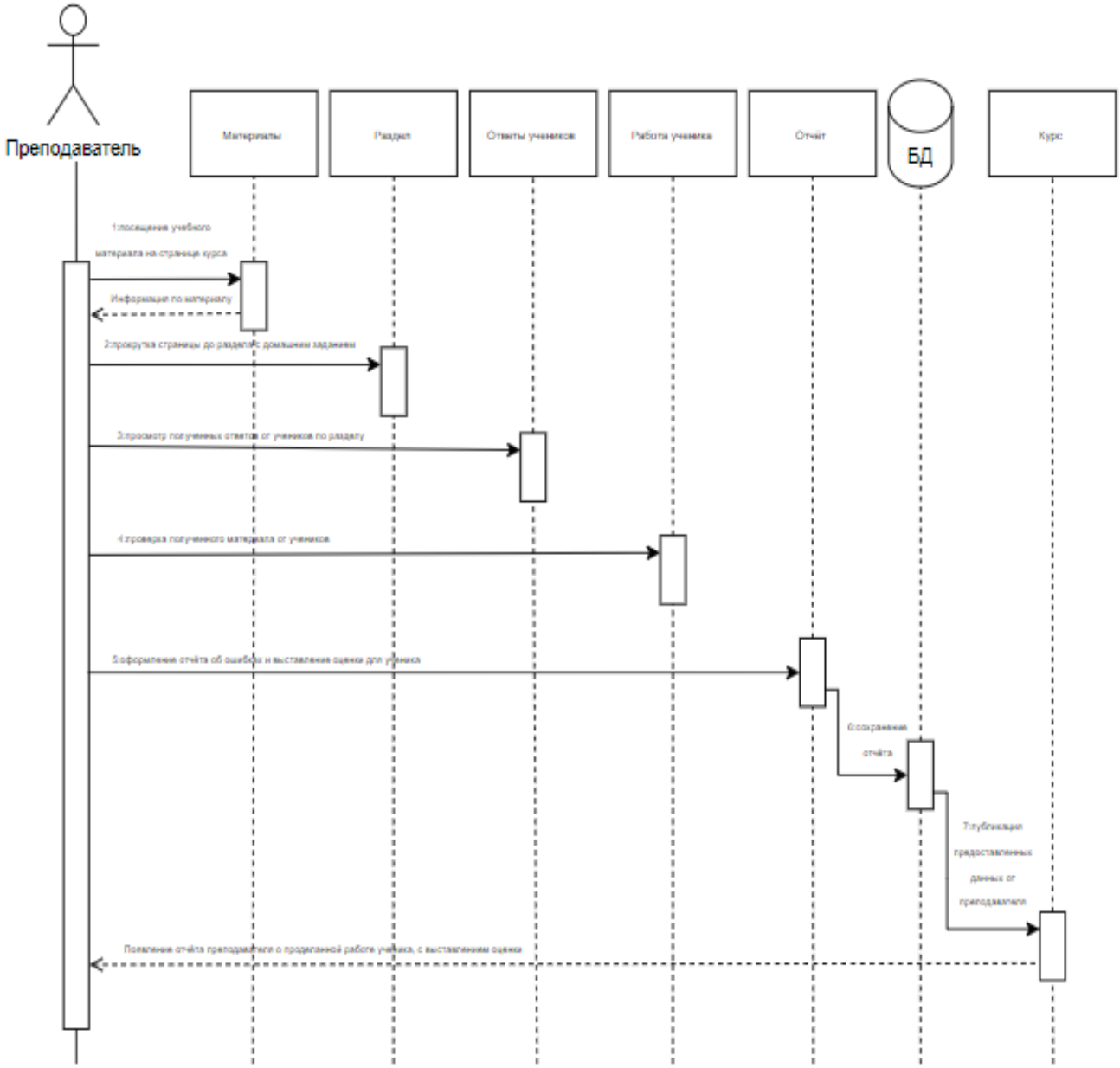


Рисунок Е.2 – Диаграмма последовательности. Проверка заданий и составление отчётов

Приложение Ж

Диаграмма деятельности

Рисунок Ж.1 – Диаграмма деятельности. Работа от лица учащегося

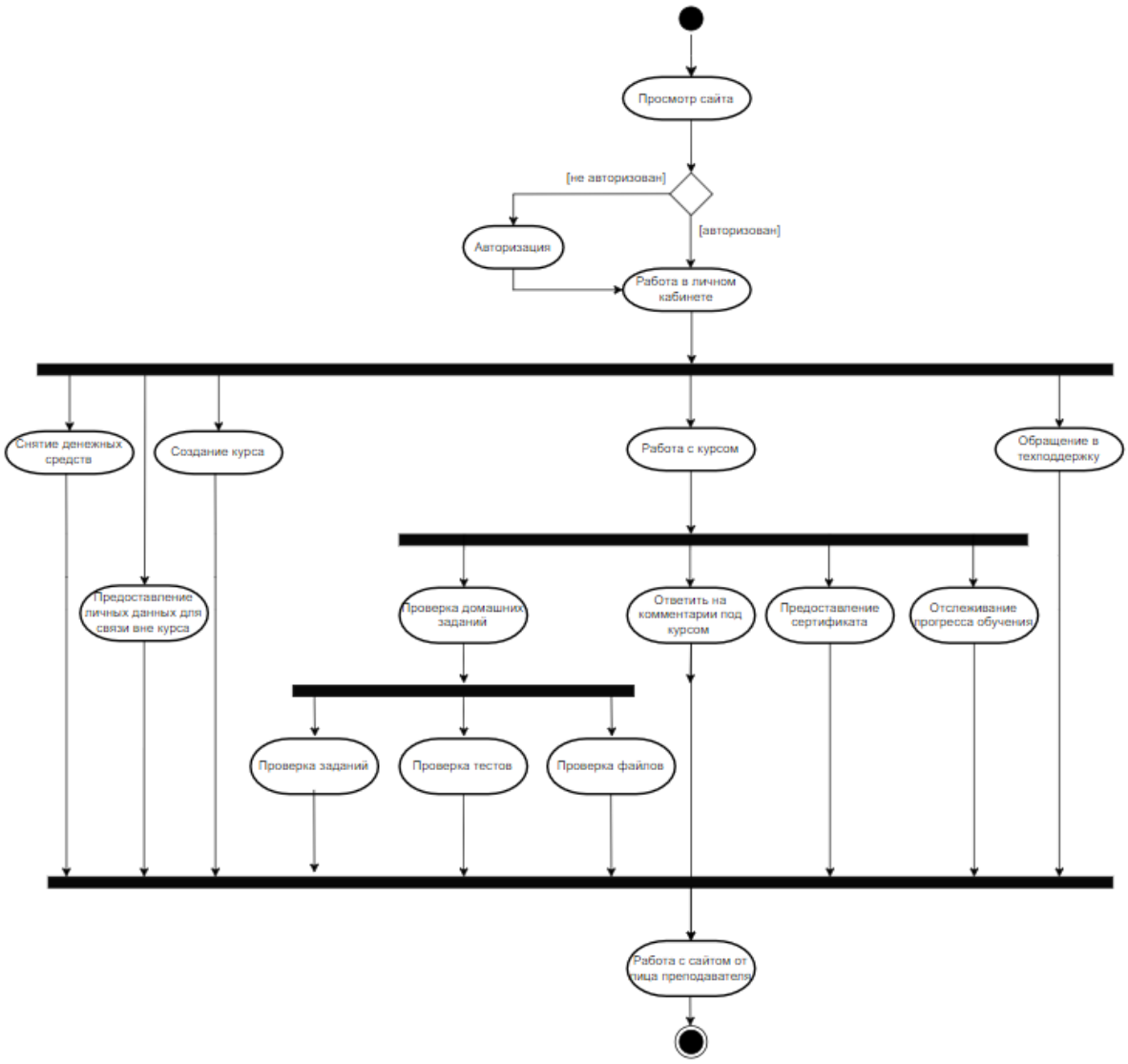


Рисунок Ж.2 – Диаграмма деятельности. Работа от лица преподавателя

Приложение И
Диаграмма классов

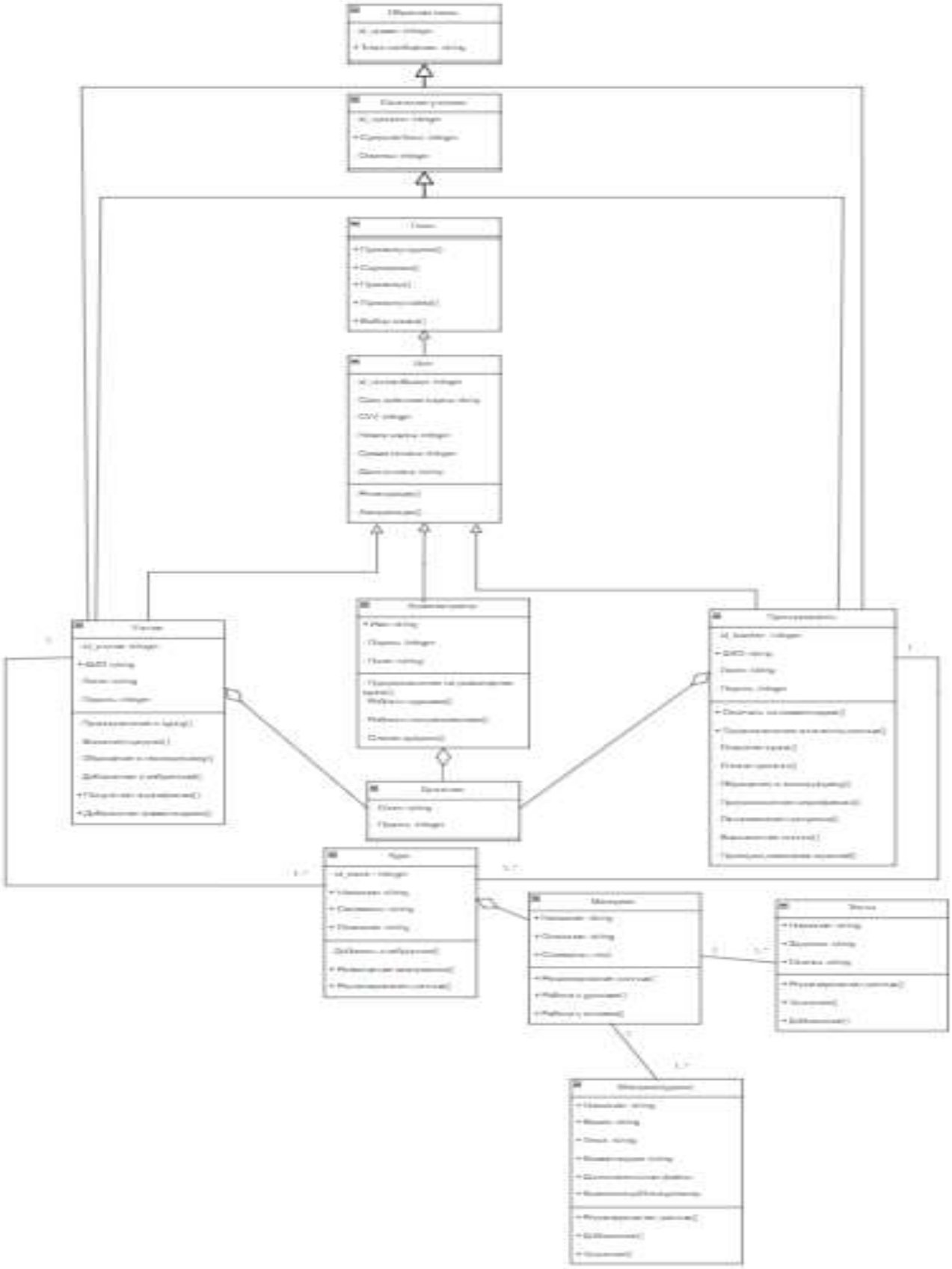


Рисунок И.1 – Диаграмма классов

Приложение К
Диаграмма объектов

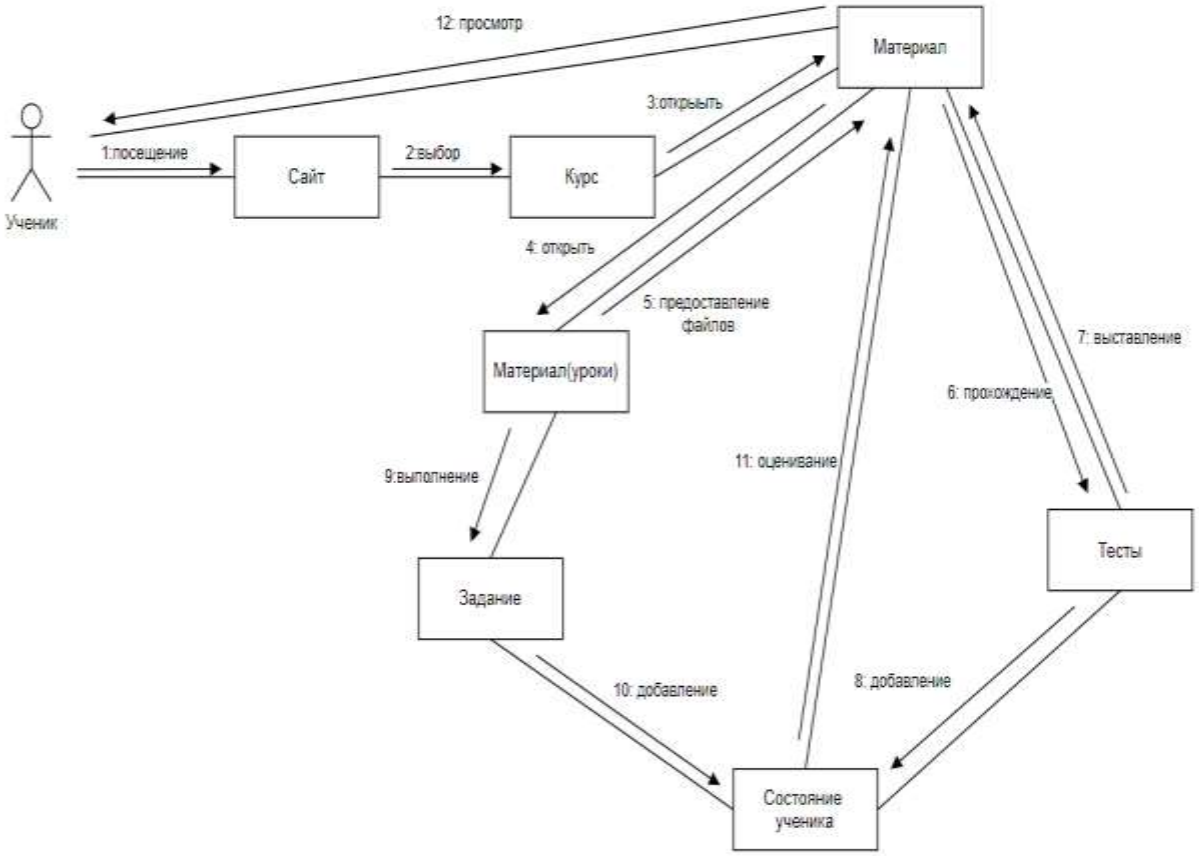


Рисунок К.1 – Диаграмма объектов. Обучение на курсе

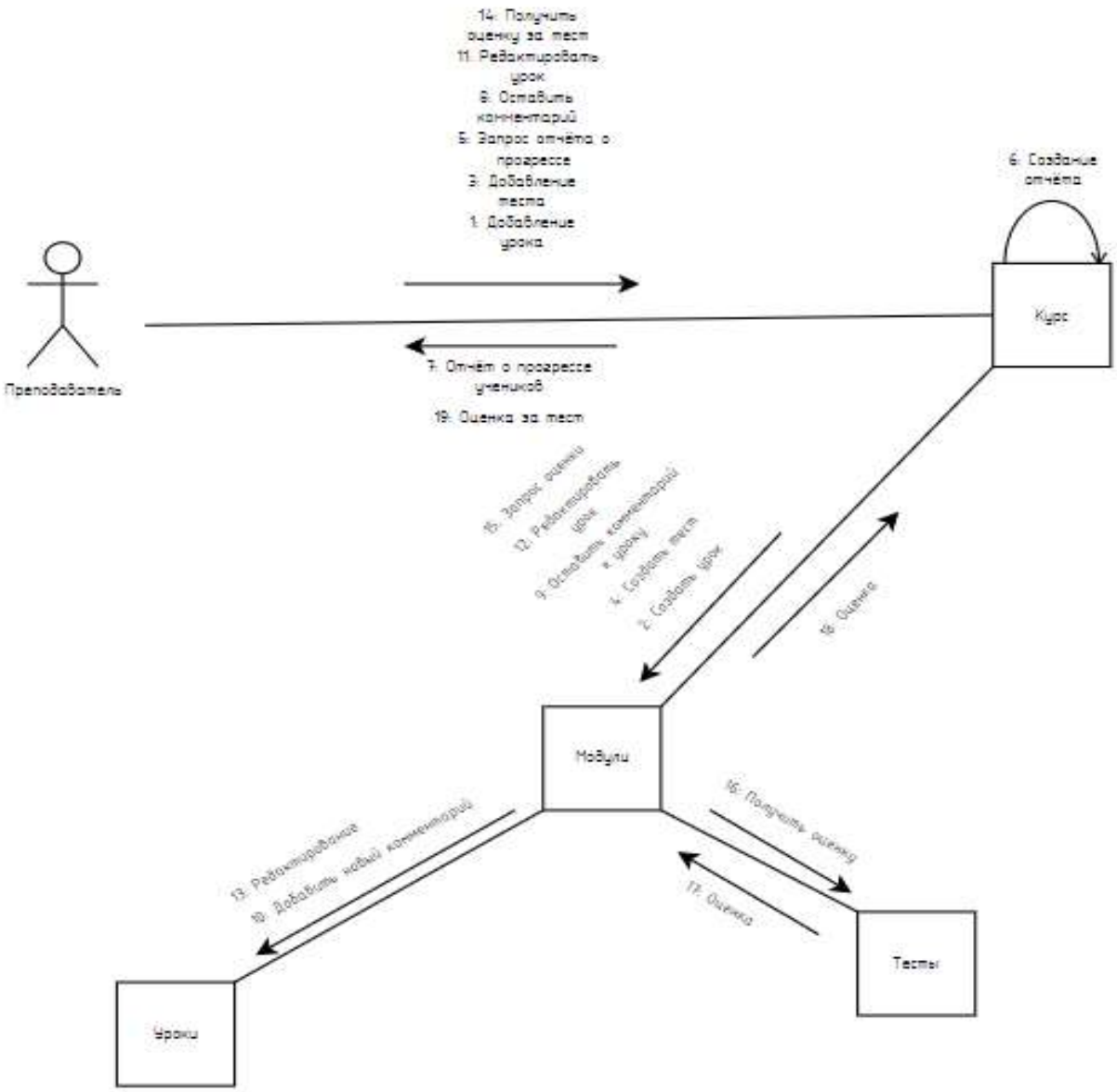


Рисунок К.2 – Диаграмма объектов. Взаимодействие преподавателя с курсом

Приложение Л
Листинг программы

```
using System.ComponentModel.DataAnnotations;

namespace EducatITion.DB.Models
{
        public class Course
        {
                [Key]
                public int Id { get; set; }
                public string NameRu { get; set; }
                public string NameEn { get; set; }
                public string DescriptionRu { get; set; }
                public string DescriptionEn { get; set; }
                public string Price { get; set; }
                public CourseType Type { get; set; }
                public string ImgPath { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;
namespace EducatITion.DB.Models
{
    public class User
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; } = null!;
        [Required]
                public string Email { get; set; } = null!;
        [Required]
        public string Password { get; set; } = null!;
        [Required]
        public bool WasInMenu { get; set; }
        [Required]
        public Role Role { get; set; }
    }
}
using EducatITion.DB.Models;
using Microsoft.EntityFrameworkCore;
namespace EducatITion.DB
{
    public class ApplicationContext : DbContext
        {
                public DbSet<User> Users { get; set; }
                public DbSet<Course> Courses { get; set; }
                public
ApplicationContext(DbContextOptions<ApplicationContext> options)
                        : base(options)
                {
                        Database.EnsureCreated();
                }
                public ApplicationContext()
                        : base()
                {
                        Database.EnsureCreated();
                }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                IConfigurationRoot configuration = new
ConfigurationBuilder()
                    .SetBasePath(Directory.GetCurrentDirectory())
                    .AddJsonFile("appsettings.json")
                    .Build();
                var connectionString =
configuration.GetConnectionString("DefaultConnection");
                optionsBuilder.UseSqlServer(connectionString);
            }
        }
                protected override void
OnModelCreating(ModelBuilder modelBuilder)
                {
                        modelBuilder.Entity<User>()
                                .Property(u => u.Id)
                                .ValueGeneratedOnAdd();
                        modelBuilder.Entity<Course>()
```

```
                .Property(u => u.Id)
                .ValueGeneratedOnAdd();
        }
    }
}
using EducatITion.Models.Send;
namespace EducatITion.Models.Combined
{
    public class CombinedCatalogueModel
    {
        public CatalogueViewModel CatalogueViewModel { get; set; } = null!;
        public SearchModel SearchModel { get; set; } = null!;
    }
}
using EducatITion.Models.User;
namespace EducatITion.Models.Combined
{
  public class CombinedIndexModel
  {
    public AuthModel AuthModel { get; set; } = null!;
    public RegModel RegModel { get; set; } = null!;
    public IndexViewModel IndexViewModel { get; set; } = null!;
  }
}
namespace EducatITion.Models.Send
{
  public class ChoiceModel
  {
    public string choice { get; set; }
  }
}
namespace EducatITion.Models.Send
{
    public class SearchModel
    {
        public string Searched { get; set; } = null!;
    }
}
```

```
}
using System.ComponentModel.DataAnnotations;
namespace EducatITion.Models.User
{
  public class AuthModel
  {
    [Required]
    public string Email { get; set; } = null!;
    [Required]
    public string Password { get; set; } = null!;
  }
}
using System.ComponentModel.DataAnnotations;
namespace EducatITion.Models.User
{
  public class RegModel
  {
    [Key]
    public int Id { get; set; }
    [Required]
    public string Name { get; set; } = null!;
    [Required]
    public string Email { get; set; } = null!;
    [Required]
    public string Password { get; set; } = null!;
    [Required]
    public string RepeatPassword { get; set; } = null!;
  }
}
using EducatITion.DB;
using EducatITion.DB.Models;
namespace EducatITion.Models
{
  public class BaseViewModel
  {
    public string Title { get; protected set; }
    public bool IsLoggedIn { get; }
    public Role Role { get; }
    private Localization Localization { get; }
```

```
        protected ISession Session { get; }
        protected ApplicationContext _context = new ApplicationContext();
        public BaseViewModel(ISession session)
        {
            var localization = session.GetString("localization") ?? Localization.ru.ToString();
            var role = session.GetString("role") ?? Role.student.ToString();
            Session = session;
            Localization = (Localization)Enum.Parse(typeof(Localization), localization);
            Role = (Role)Enum.Parse(typeof(Role), role);
            IsLoggedIn = session.Keys.Contains("username");
            Localize(Localization);
            LoadData(Localization);
        }
        protected string[][] LoadCoursesLocalizedInfo(Localization localization, int? take = null)
        {
                Course[] courses = _context.Courses.ToArray();
            take = take ?? courses.Length;
            courses = courses.Take((int)take).ToArray();
            string[] coursesTitles, coursesDescriptions, coursesPrices, coursesTypes, coursesImgsPaths;
                if (localization == Localization.ru)
                {
                    coursesTitles = courses.Select(course => course.NameRu).ToArray();
                    coursesDescriptions = courses.Select(course => course.DescriptionRu).ToArray();
                }
                else
                {
                    coursesTitles = courses.Select(course => course.NameEn).ToArray();
                    coursesDescriptions = courses.Select(course => course.DescriptionEn).ToArray();
                }
            coursesPrices = courses.Select(course => course.Price).ToArray();
            coursesImgsPaths = courses.Select(course => course.ImgPath).ToArray();
                    coursesTypes = courses.Select(course => course.Type.ToString()).ToArray();
                    return new string[][] { coursesTitles, coursesDescriptions, coursesPrices, coursesTypes, coursesImgsPaths };
                }
        protected virtual void LoadData(Localization localization)
        {
            throw new NotImplementedException();
        }
        protected virtual void Localize(Localization localisation)
        {
            throw new NotImplementedException();
        }
    }
}
namespace EducatITion.Models
{
        public class CatalogueViewModel : BaseViewModel
        {
            public string SearchText;
            public string[] CoursesDescriptions;
            public string[] CoursesTitles;
            public string[] CoursesPrices;
            public CourseType[] CoursesTypes;
            public string[] CoursesImgPaths;
            public CatalogueViewModel(ISession session) : base(session) { }
            protected override void LoadData(Localization localization)
            {
                var data = LoadCoursesLocalizedInfo(localization);
                var searched = Session.GetString("searched")?.ToLower() ?? String.Empty;
```

```
                var sort = Session.GetString("sort") ??
String.Empty;
                int[] indices = new
int[data[0].Length];
                for (int i = 0; i < indices.Length; i++)
                    indices[i] = i;
                if (sort != String.Empty)
                    Array.Sort(data[0],
indices);
                string[]
                    c1 = data[1].ToArray(),
                    c2 = data[2].ToArray(),
                    c3 = data[3].ToArray(),
                    c4 = data[4].ToArray();
                for (int i = 0; i < indices.Length; i++)
                {
                    data[1][i] = c1[indices[i]];
                    data[2][i] = c2[indices[i]];
                    data[3][i] = c3[indices[i]];
                    data[4][i] = c4[indices[i]];
                }
                if (searched != String.Empty)
                {
                    var indexisToClaim =
data[0]
                        .Select((x,
index) => new { Value = x, Index = index })
                        .Where(item
=> item.Value.ToLower().Contains(searched))
                        .Select(item =>
item.Index)
                        .ToArray();
                    CoursesTitles =
indexisToClaim.Select(index => data[0][index]).ToArray();
                    CoursesDescriptions =
indexisToClaim.Select(index => data[1][index]).ToArray();
                    CoursesPrices =
indexisToClaim.Select(index => data[2][index]).ToArray();
                    CoursesTypes =
indexisToClaim.Select(index =>
Enum.Parse<CourseType>(data[3][index])).ToArray();
                    CoursesImgPaths =
indexisToClaim.Select(index => data[4][index]).ToArray();
                    return;
                }
                CoursesTitles = data[0];
                CoursesDescriptions = data[1];
                CoursesPrices = data[2];
                CoursesTypes = data[3].Select(course
=> Enum.Parse<CourseType>(course)).ToArray();
                CoursesImgPaths = data[4];
            }
            protected override void Localize(Localization
localisation)
            {
                if (localisation == Localization.ru)
                {
                    Title = "Каталог";
                    SearchText = "Поиск...";
                }
                else if (localisation ==
Localization.en)
                {
                    Title = "Catalogue";
                    SearchText = "Search...";
                }
            }
        }
    }
    using EducatITion.DB;
    using EducatITion.DB.Models;
    using EducatITion.Models;
    using EducatITion.Models.Combined;
    using EducatITion.Models.Send;
    using EducatITion.Models.User;
    using Microsoft.AspNetCore.Mvc;
    using Microsoft.AspNetCore.Mvc.ModelBinding;
```

```
using System.Diagnostics;
using System.Reflection;
namespace EducatITion.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private ISession session { get => HttpContext.Session; }
        private ApplicationContext context = new ApplicationContext();
                        private string localization { get =>
session.GetString("localization"); }
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }
        [HttpGet]
        public IActionResult Index(string localization, string direction)
        {
            if (localization != null)
                session.SetString("localization", localization);
            if (direction == null)
                direction = "0";
            int lastInd = session.GetInt32("selectedCourseInd") ?? 0;

                session.SetInt32("selectedCourseInd",  lastInd  +
int.Parse(direction));
                                var  viewModel  =  new
CombinedIndexModel()
            {
                RegModel = new RegModel(),
                AuthModel = new AuthModel(),
                IndexViewModel = new IndexViewModel(session)
            };
            //DEBUG_DB_ADD_DATA();
            return View(viewModel);
        }
        [HttpPost]
        public IActionResult Index(CombinedIndexModel model)
        {
            RegUser(model.RegModel);
            AuthUser(model.AuthModel);
            model = new CombinedIndexModel()
            {
                RegModel = new RegModel(),
                AuthModel = new AuthModel(),
                IndexViewModel = new IndexViewModel(session)
            };

            return View(model);
        }
        private bool RegUser(RegModel model)
        {
            bool isValid = IsValid(model);
            if (isValid)
            {
                if (model.Password != model.RepeatPassword)
                    return false;
                if (GetUser(model.Email) != null)
                    return false;
                var user = new User()
                {
                    Name = model.Name,
                    Email = model.Email,
                    Password = model.Password,
                    Role  =  (Role)Enum.Parse(typeof(Role),
session.GetString("role"))
                };
                using (var context = new ApplicationContext())
                {
                    context.Users.Add(user);
                    context.SaveChanges();
                }
                WriteUserToSession(user);
            }
            return isValid;
        }
        private bool AuthUser(AuthModel model)
        {
```

```
        bool isValid = IsValid(model);
        if (isValid)
        {
            var user = GetUser(model.Email);
            if (user != null && user.Password == model.Password)
                WriteUserToSession(user);
            else
                return false;
        }
        return isValid;
    }
    private User? GetUser(string email)
    {
        return context.Users.Where(x => x.Email == email).FirstOrDefault();
    }
    private void WriteUserToSession(User user)
    {
        session.SetString("username", user.Name);
        session.SetString("login", user.Email);
        session.SetString("role", user.Role.ToString());
    }
    private bool IsValid<T>(T model)
    {
        if (model == null)
            return false;
        Type modelType = typeof(T);
        bool isValid = true;
        var propertiesNames = modelType.GetProperties(BindingFlags.Public | BindingFlags.Instance)
            .Where(x => x.Name != "Id")
            .Select(x => $"{modelType.Name}.{x.Name}")
            .ToList();
        foreach (var name in propertiesNames)
            isValid &= ModelState[name]?.ValidationState == ModelValidationState.Valid;
        return isValid;
    }
    [HttpPost]
    [Route("Home/SaveChoice")]
    public IActionResult SaveChoice([FromBody] ChoiceModel model)
    {
        session.SetString("role", model.choice);
        return Ok();
    }
    private void DEBUG_DB_ADD_DATA()
    {
        var users = new List<User>
        {
            new User { Email = "Doe1@gmail.com", Password = "123", Role = Role.teacher },
            new User { Email = "Doe2@gmail.com", Password = "123", Role = Role.student },
            new User { Email = "Doe3@gmail.com", Password = "123", Role = Role.student }
        };
        var courses = new List<Course>
        {
            new Course { NameRu = "PythonRu", NameEn = "PythonEn", DescriptionRu = "описание1", DescriptionEn = "desc1", Price = "$100-$151" },
            new Course { NameRu = "CssRu", NameEn = "CssEn", DescriptionRu = "описание2", DescriptionEn = "desc2", Price = "$100-$152" },
            new Course { NameRu = "HtmlRu", NameEn = "HtmlEn", DescriptionRu = "описание3", DescriptionEn = "desc3", Price = "$100-$153" },
            new Course { NameRu = "CssRu", NameEn = "CssEn", DescriptionRu = "описание4", DescriptionEn = "desc2", Price = "$100-$154" },
            new Course { NameRu = "HtmlRu", NameEn = "HtmlEn", DescriptionRu = "описание5", DescriptionEn = "desc3", Price = "$100-$155" }
        };
        using (var context = new ApplicationContext())
        {
            context.Users.AddRange(users);
```

```
            context.Courses.AddRange(courses);
            context.SaveChanges();
        }
    }
    public IActionResult Menu(string page)
    {
        page = page ?? "Courses";
        MenuPage type = (MenuPage)Enum.Parse(typeof(MenuPage),
page);
        var currentUser = GetUser(session.GetString("login"));
        if (type == MenuPage.Exit)
        {
            session.Clear();
            return RedirectToAction("Index");
        }
                                if (!currentUser.WasInMenu)
        {
            type = MenuPage.Greetings;
            currentUser.WasInMenu = true;
            context.SaveChanges();
        }
                                var model = new
MenuViewModel(session, type);
        return View(model);
    }
                        [HttpGet]
                        public IActionResult Catalogue()
                        {
        var viewModel = new CombinedCatalogueModel
        {
            SearchModel = new SearchModel(),
            CatalogueViewModel = new CatalogueViewModel(session)
        };
                                return View(viewModel);
                        }
    [HttpPost]
    public IActionResult Catalogue(CombinedCatalogueModel model,
string sort)
    {
        session.SetString("sort", sort ?? "");
        session.SetString("searched", model.SearchModel?.Searched ??
"");
                                var viewModel = new
CombinedCatalogueModel
                                {
                                        SearchModel = new
SearchModel(),
                                        CatalogueViewModel =
new CatalogueViewModel(session)
                                };
                                return View(viewModel);
        }
        [ResponseCache(Duration = 0, Location =
ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
namespace EducatITion.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }
        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
namespace EducatITion.Models
{
    public class IndexViewModel : BaseViewModel
    {
        public Dictionary<string, string> HeaderTextBtnsNames = new
Dictionary<string, string>
        {
            { "catalogue", "" },
            { "courses", "" },
```

```
        { "support", "" }
    };
                public Dictionary<string, string>
PlaceholdersNames = new Dictionary<string, string>
                {
                        { "name", "" },
                        { "login", "" },
                        { "password", "" },
                        { "repeat password", "" },
                };
                public Dictionary<string, string> BtnsNames =
new Dictionary<string, string>
    {
        { "signIn", "" },
        { "signUp", "" },
        { "teacher", "" },
        { "buyCourse", "" },
    };
    public string RegText;
    public string RegLikeText;
    public string StudentText;
    public string TeacherText;
    public string ContinueText;
    public string EnterText;
    public string AgreementText;
    public string[] CoursesTitles;
    public string[] CoursesDescriptions;
    public string[] CoursesImgsPaths;
    private int offset;
    public int SelectedCourseInd
    {
        get => GetCycledIndex(2);
    }
    public string SelectedCourseTitle;
    public string SelectedCourseDescription;
    public IndexViewModel(ISession session) : base(session) { }
    public string GetImgPath(int index) =>
CoursesImgsPaths[GetCycledIndex(index)];
    private int GetCycledIndex(int index)
    {
        int res;
        var cycled = (index + offset) % CoursesTitles.Length;
        if (cycled < 0)
            res = CoursesTitles.Length + cycled;
        else
            res = cycled;
        return res;
    }
    protected override void LoadData(Localization localization)
                {
                        var data =
LoadCoursesLocalizedInfo(localization);
                        CoursesTitles = data[0];
                        CoursesDescriptions = data[1];
        CoursesImgsPaths = data[4];
        offset = (int)Session.GetInt32("selectedCourseInd");
        SelectedCourseTitle = CoursesTitles[SelectedCourseInd];
        SelectedCourseDescription =
CoursesDescriptions[SelectedCourseInd];
                }
                protected override void Localize(Localization
localisation)
    {
        if (localisation == Localization.ru)
        {
            Title = "Главная";
            RegText = "Зарегистрироваться";
            RegLikeText = "Зарегистрироваться как";
            StudentText = "Студент";
            TeacherText = "Преподаватель";
            ContinueText = "Продолжить";
            EnterText = "Войти";
            AgreementText = "Нажимая кнопку «Зарегистрироваться»,
вы даёте своё согласие на обработку персональных данных
в соответствии с «Политикой конфиденциальности»
и соглашаетесь с «Условиями оказания услуг»";
            PlaceholdersNames["name"] = "Имя";
```

```
            PlaceholdersNames["login"] = "Логин";

            PlaceholdersNames["password"] = "Пароль";

            PlaceholdersNames["repeat password"] = "Повтор пароля";

            HeaderTextBtnsNames["catalogue"] = "Каталог";
            HeaderTextBtnsNames["support"] = "Поддержка";
            HeaderTextBtnsNames["courses"] = "Курсы";
            BtnsNames["signIn"] = "Войти";
            BtnsNames["signUp"] = "Регистрация";
            BtnsNames["teacher"] = "Преподаватель";
            BtnsNames["buyCourse"] = "Купить курс";
        }
        else if (localisation == Localization.en)
        {
            Title = "Main";
            RegText = "Register";
                                        RegLikeText = "Register
like";
            StudentText = "Student";
            TeacherText = "Teacher";
            ContinueText = "Continue";
                                        EnterText = "Enter";
            AgreementText = "By clicking the «Register» button, you
consent to the processing of personal data in accordance with the
«Privacy Policy» and agree to the «Terms of Service»";
            PlaceholdersNames["name"] = "Name";

            PlaceholdersNames["login"] = "Login";

            PlaceholdersNames["password"] = "Password";

            PlaceholdersNames["repeat password"] = "Repeat
password";

            HeaderTextBtnsNames["catalogue"] = "Catalogue";

            HeaderTextBtnsNames["support"] = "Support";

            HeaderTextBtnsNames["courses"] = "Courses";
                                        BtnsNames["signIn"] =
"Sign In";
                                        BtnsNames["signUp"] =
"Register";
                                        BtnsNames["teacher"] =
"Teacher";
                                        BtnsNames["buyCourse"]
= "Buy Course";
                                    }
        }
    }
}
namespace EducatITion.Models
{
        public class MenuViewModel : BaseViewModel
        {
                public Dictionary<string, string>
MenuSectionsNames = new Dictionary<string, string>
                {
                        { "support", "" },
                        { "courses", "" },
                        { "comments", "" },
                        { "exit", "" },
                };
                public Dictionary<int, string> GreetingInfo =
new Dictionary<int, string>
                {
                        { 1, "" },
                        { 2, "" },
                        { 3, "" },
                        { 4, "" },
                        { 5, "" },
                        { 6, "" },
                };
                public string ExitButtonName;
```

```
        public string StayButtonName;
        public string AreYouSureTextPart1, AreYouSureTextPart2;
        public string[] CoursesDescriptions;
        public string[] CoursesTitles;
        public string[] CoursesPrices;
        public CourseType[] CoursesTypes;
        public string[] CoursesImgPaths;
        public MenuPage PageType;
        public MenuViewModel(ISession session, MenuPage type) : base(session)
        {
            PageType = type;
        }
        protected override void LoadData(Localization localization)
        {
            var data = LoadCoursesLocalizedInfo(localization, 3);
            CoursesTitles = data[0];
            CoursesDescriptions = data[1];
            CoursesPrices = data[2];
            CoursesTypes = data[3].Select(course => Enum.Parse<CourseType>(course)).ToArray();
            CoursesImgPaths = data[4];
        }

        protected override void Localize(Localization localisation)
        {
            if (localisation == Localization.ru)
            {
                Title = "Меню";
                ExitButtonName = "Выйти";
                StayButtonName = "Остаться";
                AreYouSureTextPart1 = "Вы действительно хотите";
                AreYouSureTextPart2 = "выйти из аккаунта?";

                MenuSectionsNames["courses"] = "Курсы";

                MenuSectionsNames["comments"] = "Комментарии";

                MenuSectionsNames["support"] = "Поддержка";

                MenuSectionsNames["exit"] = "Выйти из аккаунта";
                GreetingInfo[0] = "Уважаемый пользователь!";
                GreetingInfo[1] = "Благодарим вас за выбор электронно-образовательного сайта EducatIT! Мы рады приветствовать вас в нашем сообществе, где знания становятся доступнее, а обучение — увлекательнее.";
                GreetingInfo[2] = "Ваше желание развиваться и учиться вдохновляет нас на создание качественного контента и разнообразных образовательных ресурсов. Мы стремимся предоставлять актуальную информацию и полезные инструменты, чтобы поддерживать ваш образовательный путь.";
                GreetingInfo[3] = "Если у вас есть вопросы, пожелания или предложения, пожалуйста, не стесняйтесь обращаться к нам. Мы всегда готовы помочь и сделать ваше обучение еще более эффективным.";
                GreetingInfo[4] = "Спасибо, что учитесь с EducatIT!";
                GreetingInfo[5] = @"С уважением, Команда EducatITon";
            }
            else if (localisation == Localization.en)
            {
                Title = "Menu";
                ExitButtonName = "Exit";
                StayButtonName = "Stay";
                AreYouSureTextPart1 = "Are you really want";
```

```
                AreYouSureTextPart2 =
"to exit from account?";

        MenuSectionsNames["courses"] = "Courses";

        MenuSectionsNames["comments"] = "Comments";

        MenuSectionsNames["support"] = "Support";

        MenuSectionsNames["exit"] = "Logout";
                GreetingInfo[0] = "Dear
user!";
                GreetingInfo[1] = "Thank
you for choosing the EducatIT electronic educational site! We are glad
to welcome you to our community, where knowledge becomes more
accessible and learning becomes more exciting.";
                GreetingInfo[2] = "Your
desire to grow and learn inspires us to create quality content and a variety
of educational resources. We strive to provide relevant information and
useful tools to support your educational journey.";
                GreetingInfo[3] = "If you
have any questions, requests or suggestions, please do not hesitate to
contact us. We are always ready to help and make your training even
more effective.";
                GreetingInfo[4] = "Thank
you for studying with EducatIT!";
                GreetingInfo[5] =
@"Sincerely, EducatITon Team";
                    }
                }
        }
}
```

Приложение М
Тест-кейсы

Таблица 1 – Тестирование

Идентификатор	Приоритет	Заглавие и шаги выполнения	Ожидаемый результат	Фактический результат	Статус
1	2	3	4	5	6
T_1	средний	**Свайп влево** 1. Открыть страницу https://EducatITon.com 2. Нажать на кнопку перелистывания слева.	1. Загружается главная страница сайта с информацией. 2. Нажимается кнопка перелистывания слева. 3. Все фотографии сместились на один влево. 4. Изменяется название и описание центрального курса.	1. Загружена главная страница сайта с информацией. 2. Нажата кнопка перелистывания слева. 3. Сместились все фотографии влево. 4. Изменилось название, описание центрального курса.	Пройдено
T_2	средний	**Свайп вправо** 1. Открыть страницу https://EducatITon.com 2. Нажать на кнопку перелистывания справа.	1. Загружается главная страница сайта с информацией. 2. Нажимается кнопка перелистывания справа. 3. Все фотографии сместились на один вправо. 4. Изменяется название и описание центрального курса.	1. Загружена главная страница сайта с информацией. 2. Нажата кнопка перелистывания справа. 3. Сместились все фотографии вправо. 4. Изменилось название, описание центрального курса.	Пройдено
T_3	крайне высокий	**Выбор роли в системе** 1. Нажать на меню, любые представленные кнопки или на «Зарегистрироваться». 2. Выбор интересующей роли. 3. Нажатие на соответствующую роль.	1. Нажимается меню, представленные кнопки или кнопка «Зарегистрироваться». 2. Отображается всплывающее окно с предоставленными ролями в системе. 3. Выбирается интересующая роль в системе. 4. Отображается форма для регистрации.	1. Нажато меню, представленные кнопки или кнопка «Зарегистрироваться». 2. Отображено всплывающее окно с предоставленными ролями в системе. 3. Выбрана интересующая роль в системе. 4. Отображена форма для регистрации.	Пройдено
T_4	крайне высокий	**Вход в систему** 1. Нажать на «Войти». 2. Заполнить все поля. 3. Нажать на «Продолжить».	1. Открывается форма входа. 2. Корректно заполняются поля ввода. 3. Демонстрируется главная страница.	1. Открыта форма входа. 2. Корректно заполнены поля ввода. 3. Главная страница отображена.	Пройдено
T_5	крайне высокий	**Регистрация в системе** 1. Нажать на «Зарегистрироваться». 2. Заполнить все поля. 3. Нажать на «Продолжить».	1. Открывается форма регистрации. 2. Корректно заполняются поля ввода. 3. Демонстрируется главная страница.	1. Открыта форма регистрации. 2. Корректно заполнены поля ввода. 3. Главная страница отображена.	Пройдено
T_6	минимальный	**Переход из «Войти» в «Зарегистрироваться»** 1. Нажать на «Войти». 2. Нажать на «Зарегистрироваться».	1. Открывается форма входа. 2. Нажимается кнопка «Зарегистрироваться». 3. Открывается форма регистрации.	1. Открыта форма входа. 2. Нажата кнопка «Зарегистрироваться». 3. Открыта форма регистрации.	Пройдено

Продолжение таблицы 1

1	2	3	4	5	6
T_7	минимальный	**Переход из «Войти» в «Зарегистрироваться»** 1. Нажать на «Зарегистрироваться». 2. Нажать на «Войти».	1. Открывается форма регистрации. 2. Нажимается кнопка «Войти». 3. Открывается форма входа в систему.	1. Открыта форма регистрации. 2. Нажата кнопка «Войти». 3. Открыта форма входа в систему.	Пройдено
T_8	крайне высокий	**Проверка вводимых данных** 1. Нажать на «Войти». 2. Заполнить все поля. 3. Нажать на «Продолжить».	1. Открывается форма входа. 2. Некорректно заполняются поля ввода. 3. Форма пропадает, демонстрируется первичная страница.	1. Открыта форма входа. 2. Некорректно заполнены поля ввода. 3. Форма пропала, демонстрируется первичная страница.	Пройдено
T_9	крайне высокий	**Проверка вводимых данных** 1. Нажать на «Зарегистрироваться». 2. Заполнить все поля. 3. Нажать на «Продолжить».	1. Открывается форма регистрации. 2. Заполняются поля ввода данными, которые раньше были введены. 3. Форма пропадает, демонстрируется первичная страница.	1. Открыта форма входа. 2. Заполнены поля ввода данными, которые раньше были введены. 3. Форма пропала, демонстрируется первичная страница.	Пройдено
T_10	средний	**Выбор языка** 1. Открыть страницу https://EducatITon.com, авторизировавшись. 2. Нажать на кнопку «RU»/ «EN».	1. Нажимается кнопка «RU»/ «EN». 2. Сменяется текста на сайте на русский/английский язык.	1. Нажата кнопка «RU»/ «EN». 2. Текст на сайте сменился на русский/английский язык.	Пройдено
T_11	средний	**Купить курс** 1. Нажать на кнопку «Купить курс», на главной странице. 2. Нажать на плюс напротив курса.	1. Нажимается кнопка «Купить курс». 2. Нажимается плюс напротив курса. 3. Курс добавляется в «Курсы».	1. Нажата кнопка «Купить курс». 2. Нажат плюс напротив курса. 3. Курс добавлен в «Курсы».	Пройдено
T_12	средний	**Переход на «Каталог»** 1. Нажать на кнопку «Каталог», на главной странице.	1. Нажимается кнопка «Каталог». 2. Открывается страница с каталогом курсов.	1. Нажата кнопка «Каталог». 2. Открыта страница с каталогом курсов.	Пройдено
T_13	средний	**Сортировка** 1. Нажать на стрелочки.	1. Нажимаются стрелочки сортировки. 2. Все курсы в каталоге сортируются в алфавитном порядке.	1. Нажаты стрелочки сортировки. 2. Все курсы в каталоге отсортированы в алфавитном порядке.	Пройдено
T_14	средний	**Поиск** 1. Ввести текст в поисковую строку. 2. Нажать значок поиска.	1. Вводится текст в поисковую строку. 2. Нажимается значок поиска. 3. Показываются курсы, содержащие схожий текст.	1. Введён текст в поисковую строку. 2. Нажат значок поиска. 3. Показаны курсы, содержащие схожий текст.	Пройдено
T_15	средний	**Скроллинг** 1. Нажать на полосу прокрутки.	1. Тянется полоса прокрутки. 2. Курсы изменяют своё положение.	1. Тянулась полоса прокрутки. 2. Курсы изменили своё положение.	Пройдено

Продолжение таблицы 1

T_16	средний	**Возврат на главную страницу** 1. Нажать на домик.	1. Нажимается значок домик. 2. Демонстрируется главная страница.	1. Нажат значок домик. 2. Главная страница отображена.	Пройдено
T_17	средний	**Переход на «Курсы»** 1. Нажать на кнопку «Курсы» на главной странице.	1. Нажимается кнопка «Курсы». 2. Открывается страница с личными курсами.	1. Нажата кнопка «Курсы». 2. Открыта страница с личными курсами.	Пройдено
T_18	средний	**Переход на «Каталог»** 1. Нажать на значок бумажек.	1. Нажимается значок бумажек. 2. Открывается страница с каталогом курсов.	1. Нажата значок бумажек. 2. Открыта страница с каталогом курсов.	Пройдено
T_19	средний	**Переход на «Поддержка»** 1. Нажать на кнопку «Поддержка» на главной странице.	1. Нажимается кнопка «Поддержка». 2. Открывается страница с поддержкой.	1. Нажата кнопка «Поддержка». 2. Открыта страница с поддержкой.	Пройдено
T_20	средний	**Переход в личный кабинет** 1. Нажать иконку своего изображения на главной странице.	1. Нажимается иконка своего изображения. 2. Открывается личный кабинет.	1. Нажата иконка своего изображения. 2. Открыт личный кабинет.	Пройдено
T_21	средний	**Переход на «Курсы»** 1. Нажать на кнопку «Курсы» в личном кабинете.	1. Нажимается кнопка «Курсы». 2. Открывается страница с личными курсами.	1. Нажата кнопка «Курсы». 2. Открыта страница с личными курсами.	Пройдено
T_22	средний	**Переход на «Поддержка»** 1. Нажать на кнопку «Поддержка» в личном кабинете.	1. Нажимается кнопка «Поддержка». 2. Открывается страница с поддержкой.	1. Нажата кнопка «Поддержка». 2. Открыта страница с поддержкой.	Пройдено
T_23	средний	**Переход на «Комментарии»** 1. Нажать на кнопку «Комментарии» в личном кабинете.	1. Нажимается кнопка «Комментарии». 2. Открывается страница с комментариями.	1. Нажата кнопка «Комментарии». 2. Открыта страница с комментариями.	Пройдено
T_24	высокий	**Выход из аккаунта** 1. Нажать на кнопку «Выйти из аккаунта» в личном кабинете.	1. Нажимается кнопка «Выйти из аккаунта» 2. Нажимается кнопка «Выйти» 3. Отображается первичная страница	1. Нажата кнопка «Выйти из аккаунта» 2. Нажата кнопка «Выйти» 3. Отобразилась первичная страница	Пройдено
T_25	высокий	**Выход из аккаунта** 1. Нажать на кнопку «Выйти из аккаунта» в личном кабинете.	1. Нажимается кнопка «Выйти из аккаунта» 2. Нажимается кнопка «Остаться» 3. Отображается личный кабинет	1. Нажата кнопка «Выйти из аккаунта» 2. Нажата кнопка «Остаться» 3. Отобразился личный кабинет	Пройдено