МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Кемеровский государственный университет»

Институт цифры

Кафедра Цифровых технологий

*Выпускная квалификационная работа*

*бакалаврская работа*

**«Информационная система тестирования знаний обучающихся»**

Паздерин Никита Игоревич

Тимошенко Никита Романович

направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

направленность (профиль) подготовки «Web и мобильная разработка»

Научный руководитель:

| кандидат техн. наук, доцент | |
| --- | --- |
| Завозкин С.Ю. | |
|  |  |

Работа защищена с оценками:

| Паздерин Н.И |  |  |  | Протокол ГЭК №\_\_\_\_ от « \_\_\_ »\_\_\_\_\_\_\_\_ 2024 г. |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Тимошенко Н.Р. |  |  |  | Протокол ГЭК №\_\_\_\_ от « \_\_\_ »\_\_\_\_\_\_\_\_ 2024 г. |
|  |  |  |  |  |

Секретарь ГЭК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Кемерово 2024

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 3**](#_gjdgxs)

[**ОБЗОР ЛИТЕРАТУРЫ 5**](#_30j0zll)

[**ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6**](#_1fob9te)

[1.1.2 Socrative 7](#_tyjcwt)

[1.1.3 «Информационная система контроля уровня знания SQL» 9](#_3dy6vkm)

[1.1.4 «Визуальная студия тестирования» 9](#_1t3h5sf)

[1.1.5 Moodle 11](#_4d34og8)

[1.2 Пользовательские требования 12](#_2s8eyo1)

[1.2.1 Пользовательские требования, реализованные в «Информационная система контроля уровня знания SQL». 12](#_17dp8vu)

[1.2.2. Пользовательские требования «Информационная система тестирования знаний обучающихся» . 13](#_3rdcrjn)

[**ГЛАВА 2. ПРОЕКТИРОВАНИЕ 14**](#_26in1rg)

[2.1 Диаграмма структуры данных 14](#_lnxbz9)

[2.2 UML диаграммы 18](#_1ksv4uv)

[2.2.1 Диаграммы вариантов использования 18](#_44sinio)

[2.2.2 Диаграмма последовательности 19](#_2jxsxqh)

[2.2.3 Диаграмма развертывания 22](#_3j2qqm3)

[**ГЛАВА 3. РАЗРАБОТКА 24**](#_1y810tw)

[3.1 Средства реализации 24](#_4i7ojhp)

[3.2 Архитектура 25](#_2xcytpi)

[3.2 Веб-сервисы 29](#_1ci93xb)

[3.2.1 Сервисы для работы с результатами: 29](#_3whwml4)

[3.2.2 Сервисы для работы пользователя “Студент” : 30](#_2bn6wsx)

[3.2.3 Сервисы для преподавателей: 30](#_qsh70q)

[3.3 Пользовательский интерфейс 32](#_3as4poj)

[3.4 Тестирование веб-сервисов 37](#_1pxezwc)

[3.5 Тестирование пользовательского интерфейса 39](#_49x2ik5)

[**ЗАКЛЮЧЕНИЕ 41**](#_2p2csry)

[**СПИСОК ЛИТЕРАТУРЫ 43**](#_147n2zr)

# ВВЕДЕНИЕ

Оценка успеваемости в учебном процессе является неотъемлемой частью обучения. Эффективный контроль на всех этапах обучения не только обеспечивает продуктивность учебной работы, но и обеспечивает обратную связь между обучающимся и преподавателем. Этот вид контроля оказывает значительное влияние на уровень подготовки специалистов, позволяет выявить пробелы в знаниях у студентов. Такая хорошо налаженная система помогает преподавателям оценить эффективность своей работы и определить основные направления в учебном процессе[8].

В Кемеровском государственном университете разработана Электронная информационно-образовательная среда КемГУ (ЭИОС), которая объединяет электронные ресурсы и технологии для обеспечения процесса обучения. Одним из её элементов является система компьютерного адаптивного тестирования (СКАТ). СКАТ предлагает студентам проходить тесты с классическими видами вопросов, такими как выбор одного или нескольких вариантов ответа, заполнение пропусков, упорядочивание ответов и другие. Однако система не предоставляет возможности проверять умение писать код на различных языках программирования. Для развития этих навыков студентам приходится обращаться к другим ресурсам и самостоятельно практиковаться в написании кода. У СКАТ есть ряд недостатков помимо этого:

* СКАТ устарел визуально.
* СКАТ не адаптивен, что не позволяет комфортно работать с ним на мобильном устройстве.
* Стек технологий на котором работает система не поддерживается, документация утеряна. Развивать систему крайне затруднительно.
* Система привязана к СУБД ORACLE доступ к которой ограничен на территории РФ и официальная поддержка отсутствует.

Для контроля уровня знаний учащихся по работе с запросами к базе данных на языке SQL в КемГУ в рамка выпускной квалификационной работы 2023 года обучающихся Корбина Е.К. и Оспищева А.А. был разработан прототип "Информационной системы контроля уровня знания SQL" [29]. Информационная система (ИС) позволяет создать тест и принимать запрос SQL, который сравнивается с заданным верным ответом. Прототип также имеет ряд недостатков:

* Нет системы мониторинга результатов тестирования.
* Не эргономичный интерфейс.
* Нет возможности проведения тестирования помимо SQL запросов.

Цель работы – Информационная система тестирования знаний обучающихся, предоставляющая возможности как проведения классического тестирования знаний, так и проверку знания языка SQL.

Основываясь на цели работы, можно определить задачи для участников выпускной квалификационной работы.

Задачи Паздерин Н.И.:

* анализ предметной области,
* составление требований к «Информационной системе тестирования знаний обучающихся»,
* построение модели структуры данных,
* реализация серверной части приложения:

1. реализация просмотра результатов тестирования
2. реализация создания теста стандартного тестирования преподавателя
3. реализация прохождения стандартного тестирования студентом
4. реализация проверки стандартного тестирования

* тестирование серверной части.

Задачи Тимошенко Н.Р.:

* анализ предметной области,
* составление требований к «Информационной системе тестирования знаний обучающихся»,
* построение диаграмм в нотации UML,
* реализация клиентской части приложения:

1. реализация клиентского интерфейса результатов тестирования
2. реализация клиентского интерфейса создания теста стандартного тестирования преподавателя
3. реализация клиентского интерфейса прохождения стандартного тестирования студентом
4. улучшения внешнего вида клиентского интерфейса

* тестирование ИС.

# 

# ОБЗОР ЛИТЕРАТУРЫ

Для реализации научно-исследовательской работы были проанализированы литературные источники, а также различная документация для разработчиков на интернет-ресурсах.

Для составления введения были использованы следующие источники: [8].

При выполнении анализа предметной области были использованы следующие источники: [9,10,11].

Информационные системы, использованные для обзора, представлены в: [2, 3, 4, 5, 6].

При построении диаграмм в нотации UML используются источники [7, 12].

# 

# ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

**1.1 Обзор ИС контроля уровня знания**

Целью текущего обзора является изучение ряда ИС контроля уровня знаний, в том числе на которой базируется работа. Далее, на полученных сведениях и особенностях обучающего процесса в рассмотренных ИС, реализовать требования к ИС контроля уровня знания. В обзоре внимание будет акцентироваться на:

* возможных типах вопросов,
* процессе проверки ответов,
* возможность создания контекста тестирования,

СКАТ уже внедрен в учебный процесс КемГУ, основываясь на нем можно определить основные критерии для анализа предметной области. «Информационная система контроля уровня знания SQL» является прототипом нашей системы и планировался к внедрению. Так же были взяты еще 3 системы тестирования для сравнения и определения пользовательских требований. Для обзора были выбраны следующие ИС:

* Система компьютерного адаптивного тестирования (СКАТ) КемГУ,
* Socrative,
* «Информационная система контроля уровня знания SQL»,
* «Визуальная студия тестирования»,
* Moodle.

**1.1.1 Система компьютерного адаптивного тестирования (СКАТ) КемГУ**

Информационная система, которая предоставляет управления тестами, тестовыми заданиями, темами, дисциплинами и другими данными, которые необходимы для проведения контрольных испытаний у студентов в виде теста. Данная ИС использует метод адаптивного и стандартного тестирования, предназначенный для проведения контрольных испытаний у студентов в тестовой форме.

Вопросы в конкретном тесте с точки зрения ответа на него могут быть разнородными. Рассмотрим возможные визуальные постановки вопросов и способы ответа на них:

* текстовый вопрос с выбором одного правильного варианта ответа,
* текстовый вопрос с выбором нескольких правильных вариантов ответа,
* расставление вариантов ответа в правильной последовательности,
* соотношение утверждения с правильным вариантом ответа.

Критерий оценивания теста заключается в успешном прохождении всех заданий. Процесс проверки ответа происходит автоматически – с помощью сверки с правильным ответом, подготовленным заранее автором задания.

Пример прохождения тестирования:

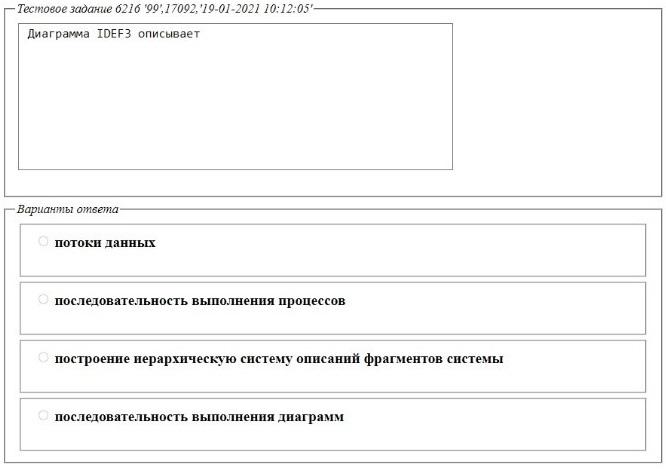


Рис 1. – Тестирование СКАТ

Достоинства:

* Преподаватель может создавать тесты по разным дисциплинам и добавить учебный материал по дисциплине.
* Система интегрирована с ЭИОС КемГУ.

Недостатки:

* Стек технологий на котором работает система не поддерживается, документация утеряна. Развивать систему крайне затруднительно.
* Не адаптивность системы не позволяет комфортно работать с мобильными устройствами.
* Не эргономичный интерфейс
* Система привязана к СУБД ORACLE доступ к которой ограничен на территории РФ и официальная поддержка отсутствует.

## 

## 1.1.2 Socrative

Socrative – это инструмент для онлайн-тестирования и оценки знаний студентов. Тесты имеет название и описание, они не привязаны к предмету или теме.

Вопросы в конкретном тесте с точки зрения ответа на него могут быть разнородными. При этом вариантов вопросов, в силу направленности этой ИС на быстрое тестирования, очень разнообразные. Некоторые из них:

* Текстовый вопрос с выбором одного правильного варианта ответа.
* Текстовый вопрос с выбором нескольких правильных вариантов ответа.
* Вопрос с ответом в свободной форме.
* Вопрос с краткие ответом.
* Вопросы правда/ложь.

Критерий оценивания теста заключается в успешном прохождении всех или части заданий. Представленные студентом ответы в динамичном режиме сравниваются с верными. Мониторинг прохождения теста отображается для преподавателя в реальном времени.

Пример прохождения тестирования:

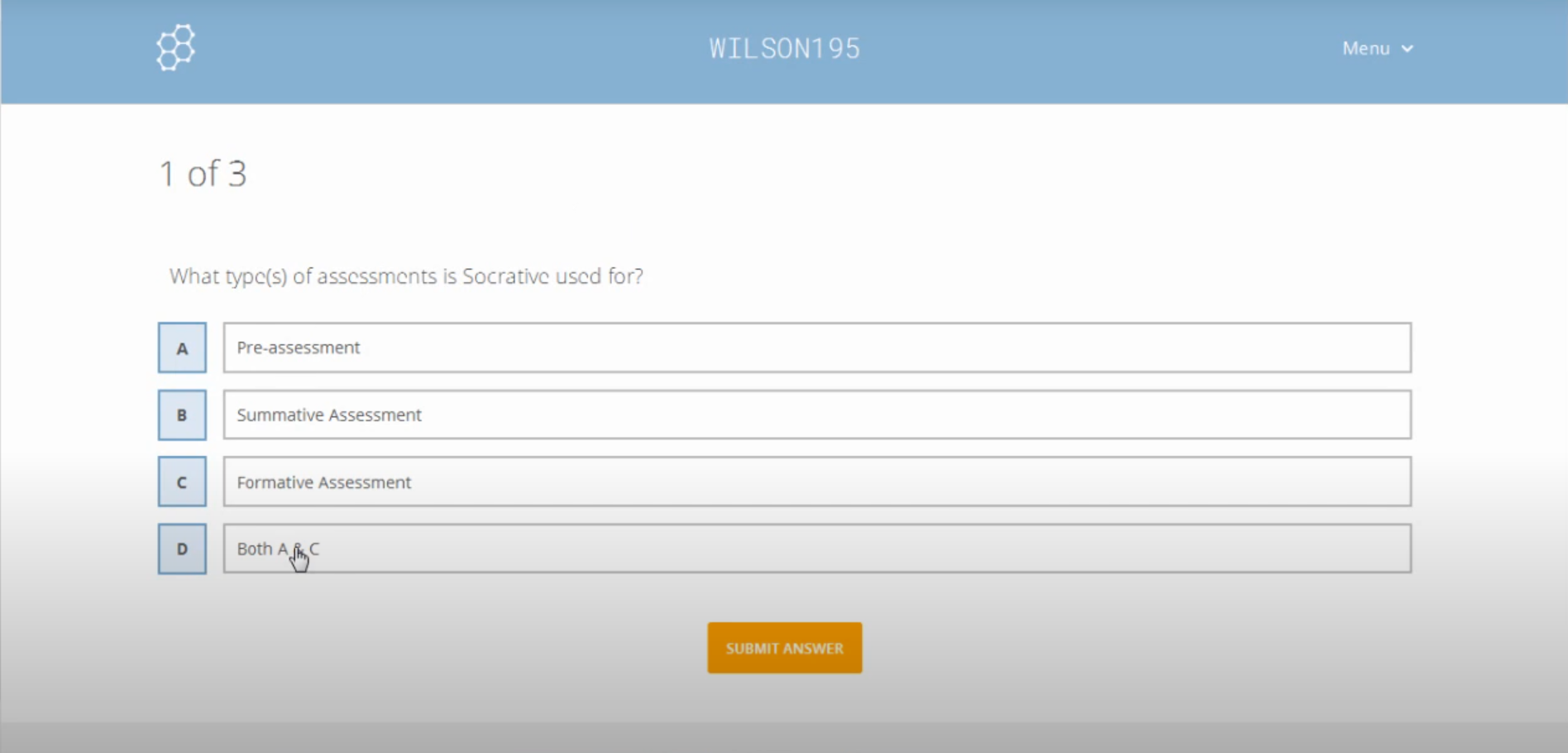


Рис 2. – Тестирование Socrative

Достоинства:

* Интерактивность: Socrative позволяет проводить интерактивные тесты и опросы в режиме реального времени, что способствует активному участию студентов в учебном процессе.
* Мгновенная обратная связь: Преподаватели могут немедленно видеть ответы студентов и результаты тестов, что позволяет им реагировать на понимание материала и корректировать лекцию в соответствии с потребностями аудитории.
* Разнообразие типов вопросов, что позволяет создавать разносторонние тесты.
* Простота использования: Интерфейс Socrative интуитивно понятен и легок в освоении, как для преподавателей, так и для студентов.
* Мобильность: Socrative доступен на разных устройствах, включая смартфоны и планшеты, что делает его удобным для использования вне аудитории.
* Доступность для прохождения теста может осуществляться QR- кодом.

Недостатки:

* Не всегда бесплатен. Хотя Socrative предоставляет бесплатную версию, существует платная версия с дополнительными функциями.
* Невозможно связать с БРС КемГУ. Систему нельзя интегрировать в уже имеющиеся ресурсы ВУЗа.
* Socrative является продуктом компании Showbie, которая базируется в Канаде. Это может впоследствии повлиять на доступ к работе граждан РФ.

## 

## 1.1.3 «Информационная система контроля уровня знания SQL»

Выпускная квалификационная работа Корбина Е.К. и Оспищева А.А. “Информационная система контроля уровня знания SQL” выполненная под руководством к.т.н., доцента С.Ю. Завозкина, направление 02.03.03, 2023 год.

Вопросы в тесте с точки зрения ответа едины. Все они принимают запрос SQL в поле ответа.

Проверка происходит по окончанию тестирования, сравнивая ответы с заранее подготовленным верным ответом автора теста.

Пример прохождения тестирования:

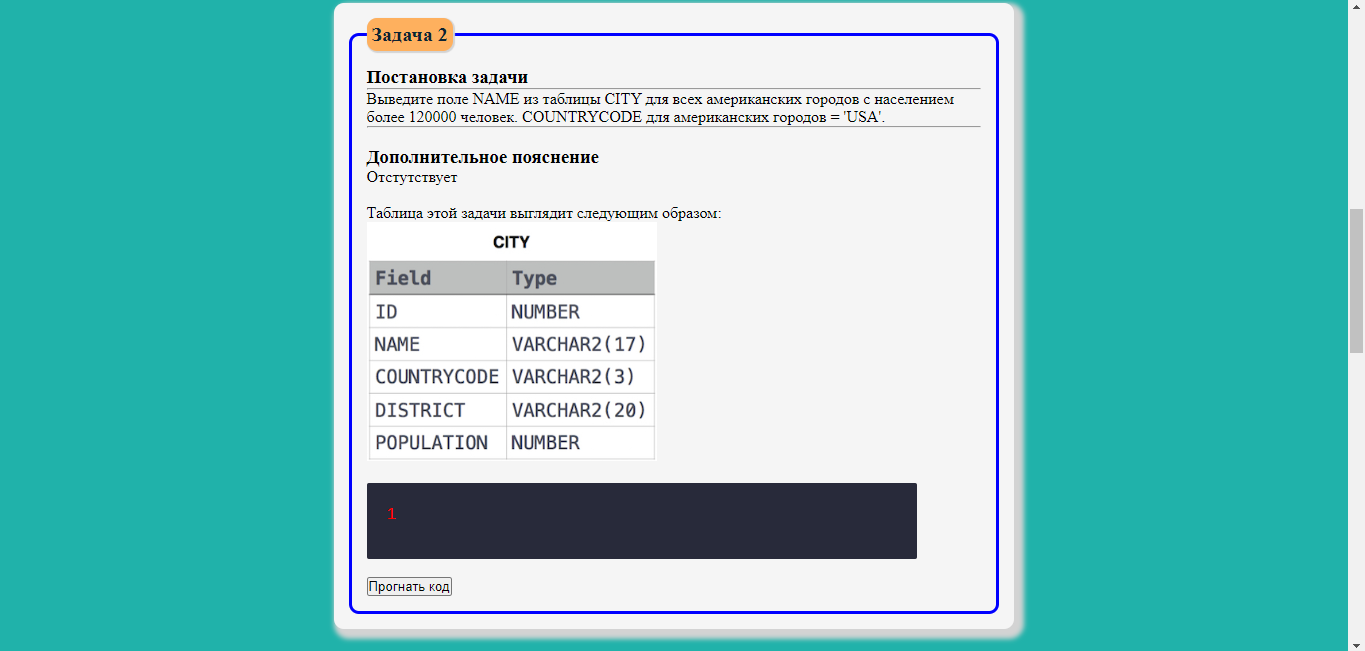


Рис 3. – Тестирование «Информационная система контроля уровня знания SQL»

Достоинства:

* Тестирования уровня знаний SQL запросов, что является, важной частью учебной программы ВУЗа.

Недостатки:

* Нет системы мониторинга результатов тестирования.
* Не эргономичный интерфейс.
* Поддерживается только один тип вопросов.

## 

## 1.1.4 «Визуальная студия тестирования»

Программный комплекс «Студия визуального тестирования» позволяет автоматизировать контроль знаний студентов, включая создание набора тестовых заданий, проведение тестирования студентов и анализ результатов.

Вопросы в конкретном тесте с точки зрения ответа на него могут быть разнородными. Рассмотрим возможные визуальные постановки вопросов и способы ответа на них:

* Да/Нет,
* выбор одного или нескольких правильных ответов,
* ввод числа или слова,
* установление последовательности и соответствия.

Пример прохождения тестирования:

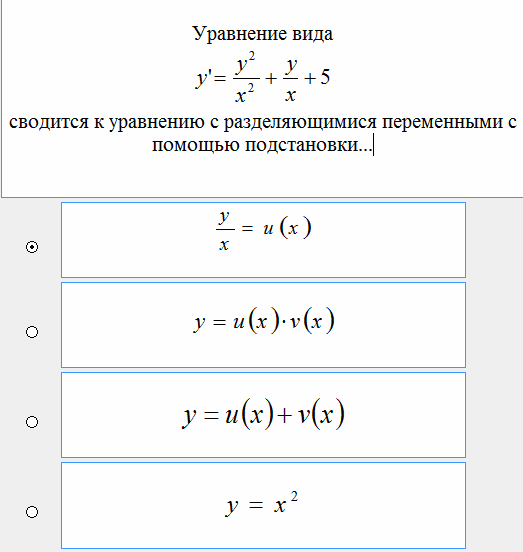


Рис 4. – Тестирование «Визуальная студия тестирования»

Для определения оценки могут использоваться два алгоритма, один из которых учитывает статистическую погрешность угадывания правильного варианта ответа. Единая база данных хранит задания и накапливаемую статистику тестирования, которую можно использовать для оценки качества тестовых заданий и совершенствования теста. Итогом проведения тестирования является отчет с результатами контроля. При необходимости можно просмотреть, на какие вопросы был дан неправильный ответ.

**Достоинства:**

* При создании текста можно использовать формулы, рисунки и сложное форматирование.
* В редакторе сценариев можно выбрать, какие задания использовать в тестировании из одного или нескольких тестов, задать время и количество заданий, определить режим тестирования.
* На основе созданного теста можно провести тестирование как на компьютерах, так и на бумажных бланках, автоматически сформированных программой.
* Система тестирования может использоваться как отдельная система, так и в связке с другими системами автоматизации. В этом случае автоматически загружаются списки студентов из ИС «Деканат» и результаты тестирования могут экспортироваться в ИС «Электронные ведомости».

**Недостатки:**

* Система привязана к ORACLE доступ к которой ограничен на территории РФ. Официальной поддержки ПО не присутствует в наше время.
* Полная версия ИС является платной, в демо версии функционал сильно ограничен.

## 

## 1.1.5 Moodle

Свободная платформа для управления курсами, предназначенная для создания онлайн-курсов и виртуальных образовательных сред. Она была разработана с целью облегчения процесса обучения и управления образовательными ресурсами.

Moodle поддерживает различные типы вопросов в тестах, что предоставляет преподавателям разнообразные инструменты для создания тестов. Вот некоторые из типов вопросов, которые поддерживаются:

* Одиночный выбор,
* Множественный выбор,
* Ложь/Истина,
* Заполнение пропусков,
* Сопоставление,
* Числовой ответ,
* Задание с кратким ответом.

Процесс проверки вопросов в тестах Moodle может быть автоматизированным или требовать ручной проверки, в зависимости от типа вопроса:

* Автоматическая проверка: Вопросы таких типов, как Одиночный выбор, Множественный выбор, Ложь/Истина, Числовой ответ и другие, могут быть проверены автоматически. Преподаватель определяет правильные ответы и критерии оценки, и система автоматически присваивает баллы.
* Ручная проверка: Открытые вопросы, задания с коротким ответом и другие типы, требующие оценки текстовых ответов, обычно требуют ручной проверки со стороны преподавателя. Преподаватель оценивает ответы и присваивает баллы в соответствии с установленными критериями.

Пример прохождения тестирования:

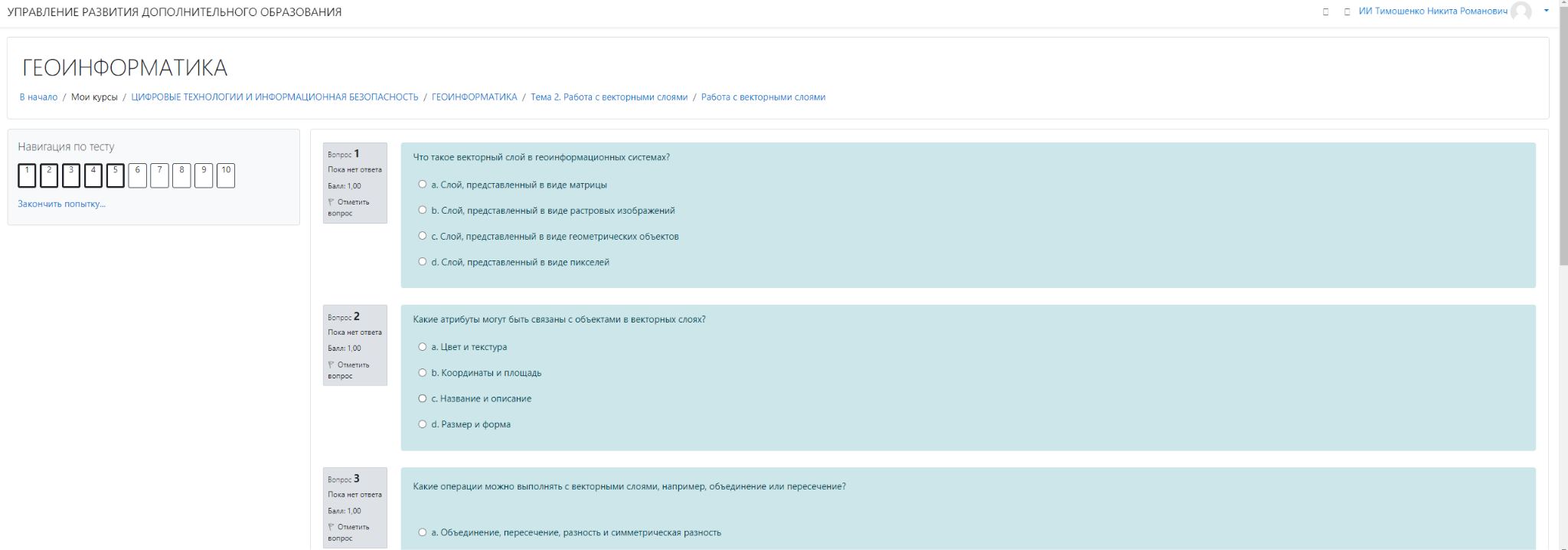


Рис 5. – Тестирование Moodle

**Достоинства:**

* Гибкость и настраиваемость: Платформа позволяет настраивать различные типы тестов и опросов, включая множество вариантов вопросов и форматов ответов.
* Удобство для обучающихся: Пользовательский интерфейс прост в использовании, что облегчает студентам процесс прохождения тестов.
* Мониторинг прогресса: Система предоставляет возможность отслеживать прогресс студентов, результаты тестов и их успеваемость.

**Недостатки:**

* Moodle является продуктом компании Moodle Pty Ltd, она базируется в Перте, Западная Австралия. Это может впоследствии повлиять на доступ к работе граждан РФ.
* Система использует стандартный способы тестирования, не имея возможности тестировать учащихся на знания запросов к базе данных SQL.

## 

## 1.2 Пользовательские требования

## 

## 1.2.1 Пользовательские требования, реализованные в «Информационная система контроля уровня знания SQL»

«Информационная система контроля уровня знания SQL» была взята за основу «Информационная система тестирования знаний обучающихся». Пользовательские требования «Информационная система контроля уровня знания SQL» представлены ниже.

Для роли «Студент»:

* регистрация и авторизация по логину и паролю,
* восстановление пароля через электронную почту,
* редактирование и просмотр своей учетной записи,
* выбор языка запросов БД для тестирования,
* прохождение тестирования на знания SQL,

Для роли «Преподаватель»:

* авторизация по логину и паролю,
* редактирование и просмотр своей учетной записи,
* создание тестов в конструкторе тестов,
* редактирование тестов,
* удаление тестов,
* добавление для задачи постановки вопроса,
* добавление дополнительного пояснения,
* добавление скрипта с правильным решением,
* добавление графика со структурой таблицы,
* добавление скрипта для создания таблицы,
* добавление скрипта для генерации данных в таблице,
* настройка времени тестирования.

Для роли «Администратор»:

* авторизация,
* выдача и изъятие роли преподавателя,
* создание и удаление пользователей.

## 1.2.2. Пользовательские требования «Информационная система тестирования знаний обучающихся»

«Информационная система тестирования знаний обучающихся» предоставляет возможность прохождения классического тестирования, тестирования состоит из 3 типов вопросов:

* «Единственный ответ»,
* «Множественный ответ»,
* «Открытый ответ».

Добавленные требования подчеркнутый пунктиром и представлены ниже.

Для роли «Студент»:

* регистрация и авторизация по логину и паролю,
* восстановление пароля через электронную почту,
* редактирование и просмотр своей учетной записи,
* просмотр информации о тесте,
* прохождение тестирования знания SQL,
* выбор языка запросов БД для тестирования,
* прохождение стандартного тестирования,

Для роли «Преподаватель»:

* авторизация по логину и паролю,
* редактирование и просмотр своей учетной записи,
* создание тестов в конструкторе тестов SQL,
* создание тестов в конструкторе стандартного тестирования,
* редактирование тестов,
* удаление тестов,
* выбор типа теста,
* выбор типа вопроса,
* добавление для задачи постановки вопроса,
* добавление скрипта с правильным решением,
* добавление графика со структурой таблицы,
* добавление скрипта для создания таблицы,
* добавление дополнительного пояснения,
* добавление скрипта для генерации данных в таблице,
* настройка времени тестирования.
* просмотр результатов тестирования.

# 

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ

## 2.1 Модель данных

Разрабатываемая ИС для хранения данных использует MongoDB – документоориентированную систему управления базами данных. Хранение данных реализовано в соответствии с рисунком 1. Между коллекциями Users и replyStud установлена связь «один-ко-многим»: одному студенту может принадлежать много историй прохождения теста, однако одна история прохождения теста принадлежит одному студенту. Между коллекциями replyStud и Tests установлена связь «один-к-одному»: одна история прохождения теста принадлежит одному тесту и только один тест может храниться в истории прохождения. В ходе проектирования были добавлены следующие изменения:

В коллекцию **Tests**:

* type - содержит тип теста,
* lector\_id- идентификатор преподавателя, которому принадлежит тест.
* answerOptions - массив хранящий ответы и Boolean значения правильности каждого из ответов,
* problemStatement, problemPreview, problemSolution, scriptTable, scriptTableData теперь могут быть равны NULL.

В коллекцию **replyStud**:

* lector\_id - идентификатор преподавателя, которому принадлежит тест.
* Data - информация о дате прохождения теста в формате ISOData.

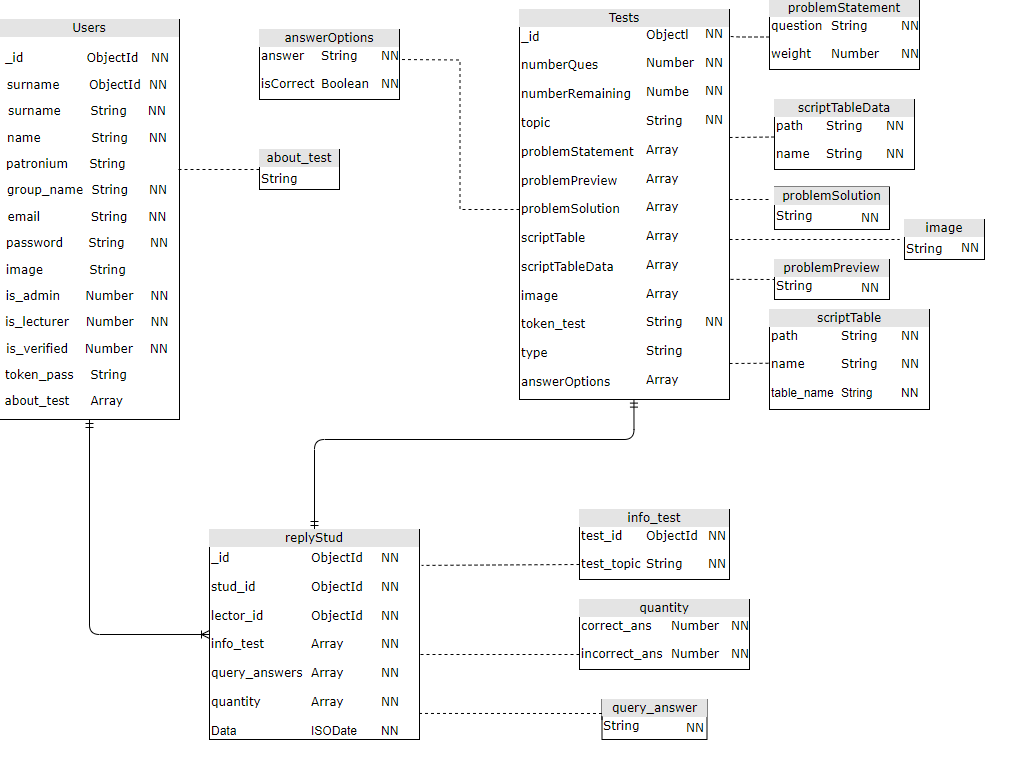


Рис 6. – Модель данных

Коллекция **Users** хранит информацию о пользователях:

* \_id – идентификатор пользователя, первичный ключ,
* surname – фамилия,
* name – имя,
* patronim – отчество,
* group\_name – группа студента,
* email – электронная почта,
* password – пароль,
* image – картинка учетной записи,
* is\_admin – параметр, устанавливающий роль администратора,
* is\_lecturer – параметр, устанавливающий роль преподавателя,
* is\_verified – параметр, показывающий статус учётной записи (подтверждена или нет),
* token\_pass – токен пароля. Когда задействован веб-сервис восстановления пароля этому полю присваивается токен, в обычном состоянии поле пустое,
* about\_test – изначально пустой массив. Хранит идентификаторы всех пройденных тестов.

Коллекция **Tests** хранит информацию о созданных тестах:

* \_id – идентификатор теста, первичный ключ,
* numberQues – количество вопросов,
* numberRemaining – количество минут на тест,
* topic – тема теста,
* problemStatement – массив, хранящий постановки задач, и вес от вопроса, для задний с несколькими вариантами ответов,
* problemPreview – массив, хранящий дополнительную информацию к самой задаче,
* problemSolution – массив, хранящий пути к скриптам и файлам с правильным ответом, если тест не на знание SQL, то хранит массив из 1 элемента,
* scriptTable – массив объектов. Параметр path хранит путь к скрипту, который генерирует таблицу. Параметр name хранит имя скрипта. Параметр table\_name хранит имя таблицы. Если тест не на проверку уровня знаний SQL, то null,
* scriptTableData – массив объектов. Параметр path хранит путь к скрипту, который генерирует данные в таблице. Параметр name хранит имя скрипта. Если тест не на проверку уровня знаний SQL, то null,
* image – массив, хранящий пути к картинкам таблиц,
* token\_test – токен теста,
* type - содержит тип вопроса
* answerOptions - массив хранящий ответы и Boolean значения правильность этих ответов.

Коллекция **replyStud** хранит историю с информацией о результатах тестирования для конкретного студента:

* \_id – идентификатор тестирования, первичный ключ,
* stud\_id – идентификатор студента,
* lector\_id - идентификатор преподавателя, которому принадлежит тест.
* test\_topic – массив объектов. Параметр test\_id хранит идентификатор пройденного теста. Параметр test\_topic хранит тему пройденного теста,
* query\_answers – массив, хранящий ответы студента на этот тест,
* quantity – массив объектов. Параметр correct\_ans хранит число правильных ответов. Параметр incorrect\_ans хранит число неправильных ответов.
* Data - информация о дате прохождения теста в формате ISOData.

## 

## 2.2 UML диаграммы

Для создания UML-диаграмм использовалась программа Draw.io.

## 

## 2.2.1 Диаграммы вариантов использования

Диаграмма вариантов использования – это один из видов диаграмм UML, предназначенных для моделирования динамических аспектов систем [8]. Она обеспечивает доступность и понятность систем, подсистем и классов за счет внешнего представления того, как эти элементы могут быть использованы в контексте. В соответствии с рисунками 2, 3, 4 отображены три роли в ИС: студент, преподаватель и администратор.



Рис 7. - Диаграмма вариантов использования для роли “Студент”



Рис 8. -Диаграмма вариантов использования для роли “Преподаватель”



рис 9. - Диаграмма вариантов использования для роли “Администратор”

## 2.2.2 Диаграмма последовательности

Диаграмма последовательности представляет сценарий или поток событий в одном единственном случае использования. Поток сообщений диаграммы последовательностей основан на описании конкретного случая использования. Для визуализации данной диаграммы в качестве примеров используются следующие сценарии: авторизация студента в соответствии с рисунком 5, процесс создания задачи преподавателем в соответствии с рисунком 6 и процесс проверки результатов тестирования в соответствии с рисунком 7.

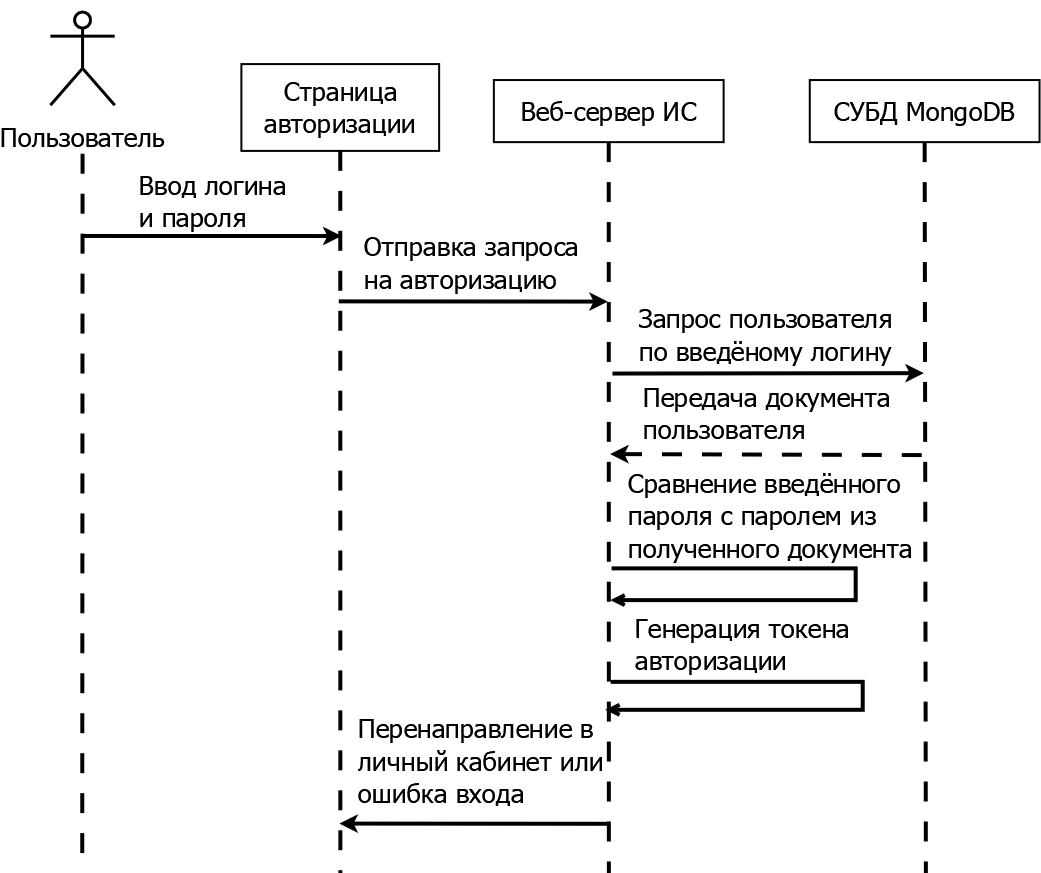


Рис 10. – Диаграмма последовательности для сценария “авторизация”

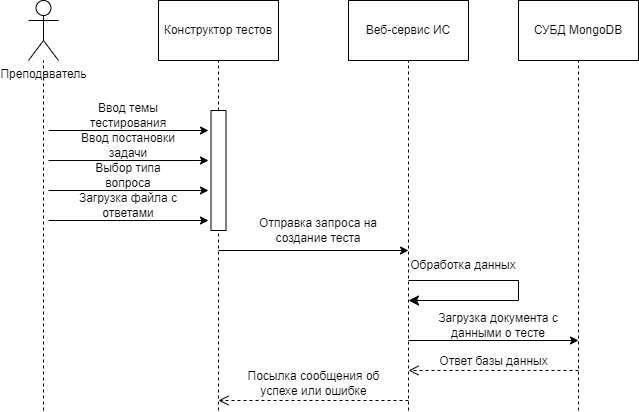


Рис 11. –Диаграмма последовательности для сценария “процесс создания задачи преподавателем”



Рис 12. –Диаграмма последовательности для сценария “процесс проверки результатов тестирования”

## 2.2.3 Диаграмма развертывания

Диаграммы развертывания используются для визуализации аппаратных процессоров/узлов/устройств системы, каналов связи между ними и размещения программных файлов на этом аппаратном обеспечении. В связи с доработкой информационной системы, используются программно аппаратные средства «Информационная система контроля уровня знания SQL», диаграмма развертывания остается такой же. Визуализация этого вида UML-диаграммы представлена в соответствии с рисунком 8.

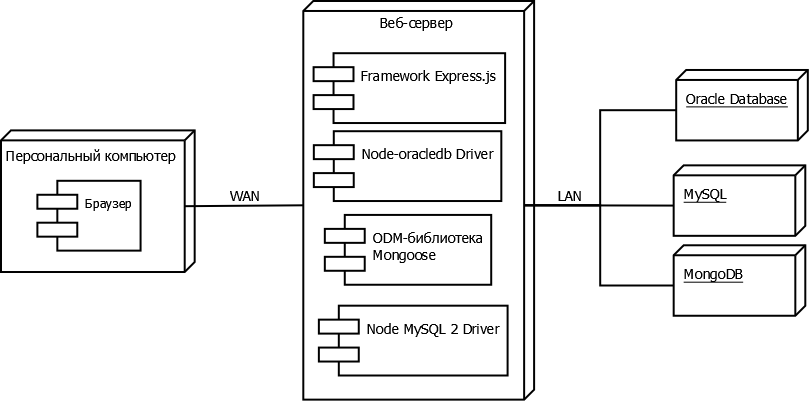


Рис 13. – Диаграмма развёртывания.

Для развертывания системы тестирования требуется следующее оборудование и программное обеспечение:

**Клиентская часть**

Клиентская часть системы представляет собой веб-приложение, которое может быть запущено на любом персональном компьютере с установленным браузером. Основные требования:

* Персональный компьютер
* Любой современный веб-браузер (Google Chrome, Mozilla Firefox, Microsoft Edge и т.д.)

**Серверная часть**

Серверная часть системы включает в себя несколько компонентов, которые взаимодействуют друг с другом для обеспечения корректной работы приложения. Основные требования и компоненты серверной части:

* Среда выполнения Node.js
  + Версия: Node.js v20.11.0 или выше
  + Node.js используется как основная среда выполнения серверного кода. Установка Node.js включает в себя npm (Node Package Manager), который используется для управления зависимостями и библиотеками.
* Необходимые библиотеки JS:
  + Express.js: библиотека для создания и управления веб-серверами и маршрутизацией HTTP-запросов.
  + Node-oracledb Driver: драйвер для взаимодействия с базой данных Oracle.
  + Mongoose: библиотека для взаимодействия с MongoDB и управления схемами данных.
  + Node MySQL 2 Driver: драйвер для работы с базой данных MySQL.
* Системы управления базами данных (СУБД):
  + Oracle: используется для выполнения скриптов и проверки знаний студентов по языку SQL. Oracle необходима для обеспечения поддержки специфических функций и синтаксиса SQL.
  + MySQL: также используется для выполнения SQL-скриптов и проверки знаний студентов. MySQL необходима для поддержки другого набора специфических функций и синтаксиса SQL.
  + MongoDB: используется как основная база данных для хранения данных о пользователях, тестах и результатах тестирования.

# ГЛАВА 3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЯ

## 3.1 Средства реализации

Разрабатываемая информационная система основана на паттерне Model-View-Controller (MVC). Этот выбор обеспечивает гибкость при разработке, позволяя, например, изменять веб-форму, не затрагивая при этом логику обработки данных. MVC также поддерживает концепцию разделения, отделяя бизнес-логику от интерфейса, что позволяет изменять один компонент без влияния на другие.

Для разработки был использован архитектурный стиль REST API, обеспечивающий разделение клиента и сервера, где взаимодействие с ресурсами осуществляется через методы HTTP.

Для создания информационной системы использовались следующие инструменты:

* Visual Studio Code – редактор исходного кода для кроссплатформенной разработки веб- и облачных приложений.
* JavaScript – мультипарадигменный язык программирования.
* Node.js – кросс-платформенная среда выполнения JavaScript с открытым исходным кодом.
* Express.js – веб-фреймворк для Node.js, написанный на JavaScript.
* Node-oracledb – драйвер базы данных для высокопроизводительных приложений с использованием Oracle Database на JavaScript или TypeScript.
* MongoDB – документоориентированная система управления базами данных, не требующая описания схемы таблиц.
* Oracle Database – объектно-реляционная система управления базами данных.
* SQL – декларативный язык программирования для создания, обработки и хранения данных в реляционных базах данных.
* Node.js делает процесс разработки гибким и менее затратным по времени. Использование одного языка на стороне клиента и сервера обеспечивает быструю синхронизацию, что особенно полезно для приложений реального времени, основанных на событиях. Благодаря асинхронной однопоточной архитектуре, Node.js отлично подходит для проектов, требующих постоянного обновления данных. Наличие менеджера пакетов (NPM) позволяет легко добавлять разнообразные библиотеки.

## 

## 3.2 Архитектура

Проект основан на архитектурном стиле REST API, который предполагает использование определённых принципов и методов для организации взаимодействия между клиентом и сервером через протокол HTTP. REST, что расшифровывается как Representational State Transfer (передача репрезентативного состояния), представляет собой архитектурный подход, разработанный для создания масштабируемых и эффективных веб-сервисов. Этот стиль обеспечивает структурированное и предсказуемое взаимодействие, где каждая операция соответствует определённому HTTP-методу (например, GET, POST, PUT, DELETE) и ресурсы идентифицируются через уникальные URL. Основная цель REST заключается в том, чтобы обеспечить простоту и универсальность обмена данными между системами, делая их легко расширяемыми и поддерживаемыми [25].

Информационная система состоит из двух основных компонентов: Front-end и Back-end. В Front-end используется шаблонизатор EJS[26] для создания пользовательского интерфейса, что позволяет динамически отображать данные и обеспечивать удобство использования для конечного пользователя. В то время как Back-end включает в себя серверную инфраструктуру, которая обрабатывает запросы от Front-end, управляет данными и выполняет основную логику приложения. Эти две части работают вместе, обеспечивая полноценное функционирование информационной системы.

В приложении используется подход чистой архитектуры (Clean Architecture) [27], который направлен на создание гибких и легко поддерживаемых программных систем.

Этот архитектурный стиль, предложенный Робертом Мартиным, позволяет разделять код на четкие слои, каждый из которых имеет свою собственную ответственность и уровень абстракции. Основные слои включают в себя слой приложений (Application Layer), слой домена (Domain Layer), слой интерфейсов (Interface Adapters), и слой фреймворков и драйверов (Frameworks and Drivers). В центре находится слой домена, который содержит бизнес-логику и сущности, независимые от внешних систем. Вокруг него располагается слой приложений, где находятся случаи использования (use cases), которые определяют поведение системы в ответ на пользовательские действия. Далее идет слой интерфейсов, где реализованы интерфейсы для взаимодействия с пользователями и другими системами, включая API и пользовательские интерфейсы. Самый внешний слой включает фреймворки и драйверы, такие как базы данных, веб-серверы и другие инфраструктурные компоненты. Такой подход позволяет легко изменять и тестировать отдельные части приложения, улучшает его масштабируемость и упрощает добавление новых функций, поскольку изменения в одном слое минимально влияют на другие слои.

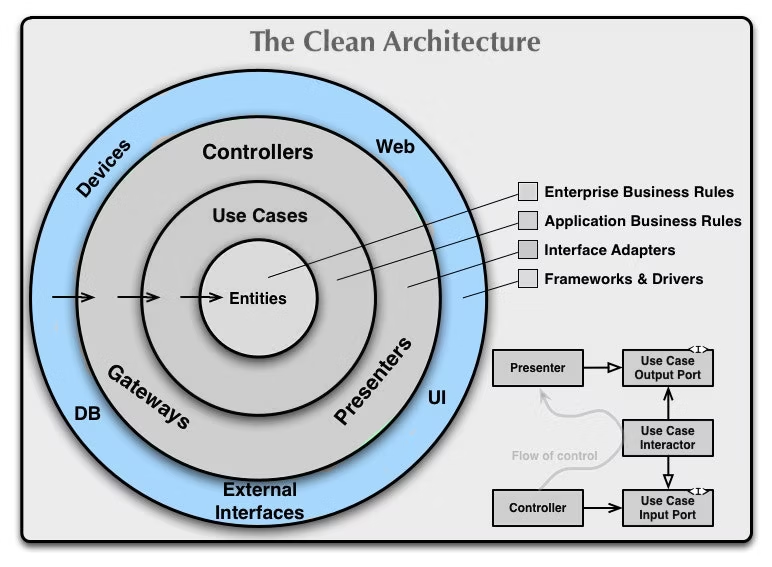


рис. 14 - Диаграмма чистой архитектуры.

Система разделена на несколько архитектурных слоев:

**Контроллеры (controllers)**:

* Отвечают за прием HTTP-запросов от клиентов.
* Обрабатывают входные данные и вызывают соответствующие методы из слоя use\_case.
* Подготавливают и отправляют ответы обратно клиентам.
* Не содержат бизнес-логику, а лишь координируют взаимодействие между HTTP и слоем use\_case.

**Слой use\_case**:

* Содержит конкретные случаи использования (use cases) приложения.
* Реализует бизнес-логику и операции, которые могут быть выполнены в системе.
* Независим от внешних фреймворков и библиотек.
* Включает в себя операции, такие как добавление результатов, получение списка тестов и другие важные функции приложения.

**Слой доступа к данным (data-access-layer)**:

* Содержит репозитории, которые отвечают за взаимодействие с базой данных или другими источниками данных.
* Реализует методы для выполнения операций CRUD (Create, Read, Update, Delete) с объектами предметной области.
* Обеспечивает абстракцию от конкретной реализации базы данных, что облегчает изменение и масштабирование системы.

**Сущности (entity)**:

* Представляют собой основные объекты бизнес-логики приложения, такие как тесты и пользователи.
* Содержат состояние и поведение объектов в системе.
* Не зависят от других слоев приложения и содержат только логику, специфичную для предметной области.

**Промежуточное программное обеспечение (middleware)**:

* Обрабатывает промежуточные этапы обработки запросов перед их передачей контроллерам или после получения ответов от контроллеров.
* Выполняют такие задачи, как аутентификация пользователей, обработка исключений, логирование и другие операции, необходимые для обеспечения безопасности и стабильности системы.

**Слой транспорта (transport)**:

* Определяет маршруты для обработки HTTP-запросов и их направления к соответствующим контроллерам.
* Преобразует внешние запросы в формат, который может быть обработан контроллерами и другими частями приложения.
* Отвечает за взаимодействие с внешними системами и клиентами, обеспечивая интерфейс для работы с приложением.

## 3.2 Веб-сервисы

Приложение на Node.js с использованием фреймворка Express.js предоставляет REST-сервисы для взаимодействия с приложением через протокол HTTP. При обработке запросов на Front-end используются шаблоны EJS для динамической генерации HTML-страниц, что обеспечивает создание интерактивных пользовательских интерфейсов. Каждый маршрут определяет соответствующие обработчики запросов, взаимодействующие с частями приложения для обеспечения необходимого функционала.

## 3.2.1 Сервисы для работы с результатами:

* Получение результатов по теме теста:
  + Метод: GET
  + URL: /result/:topic
  + Права: авторизация лектора
  + Описание: Получает результаты тестов определенной темы.
* Получение всех результатов:
  + Метод: GET
  + URL: /result
  + Права: авторизация лектора
  + Описание: Получает все результаты тестов.

## 3.2.2 Сервисы для работы пользователя “Студент” :

* Получение всех тестов:
  + Метод: GET
  + URL: /tests
  + Права: авторизация пользователя
  + Описание: Получает список всех доступных тестов.
* Получение конкретного теста по идентификатору:
  + Метод: GET
  + URL: /test/:id
  + Права: авторизация пользователя
  + Описание: Получает информацию о конкретном тесте по его идентификатору.
* Сохранение результата теста:
  + Метод: POST
  + URL: /test
  + Права: авторизация пользователя
  + Описание: Сохраняет результат теста для конкретного пользователя.

## 3.2.3 Сервисы для преподавателей:

* Отображение формы создания теста:
  + Метод: GET
  + URL: /test
  + Права: авторизация лектора
  + Описание: Отображает форму для создания нового теста.
* Отображение классического конструктора тестов:
  + Метод: GET
  + URL: /constructor
  + Права: авторизация лектора
  + Описание: Отображает классический конструктор тестов.
* Создание классического теста:
  + Метод: POST
  + URL: /test
  + Права: авторизация лектора
  + Описание: Создает новый классический тест.
* Обновление теста по идентификатору:
  + Метод: PUT
  + URL: /test/:id
  + Права: авторизация лектора
  + Описание: Обновляет информацию о существующем тесте.
* Удаление теста по идентификатору:
  + Метод: DELETE
  + URL: /test/:id
  + Права: авторизация лектора
  + Описание: Удаляет указанный тест.

## 

## 3.3 Пользовательский интерфейс

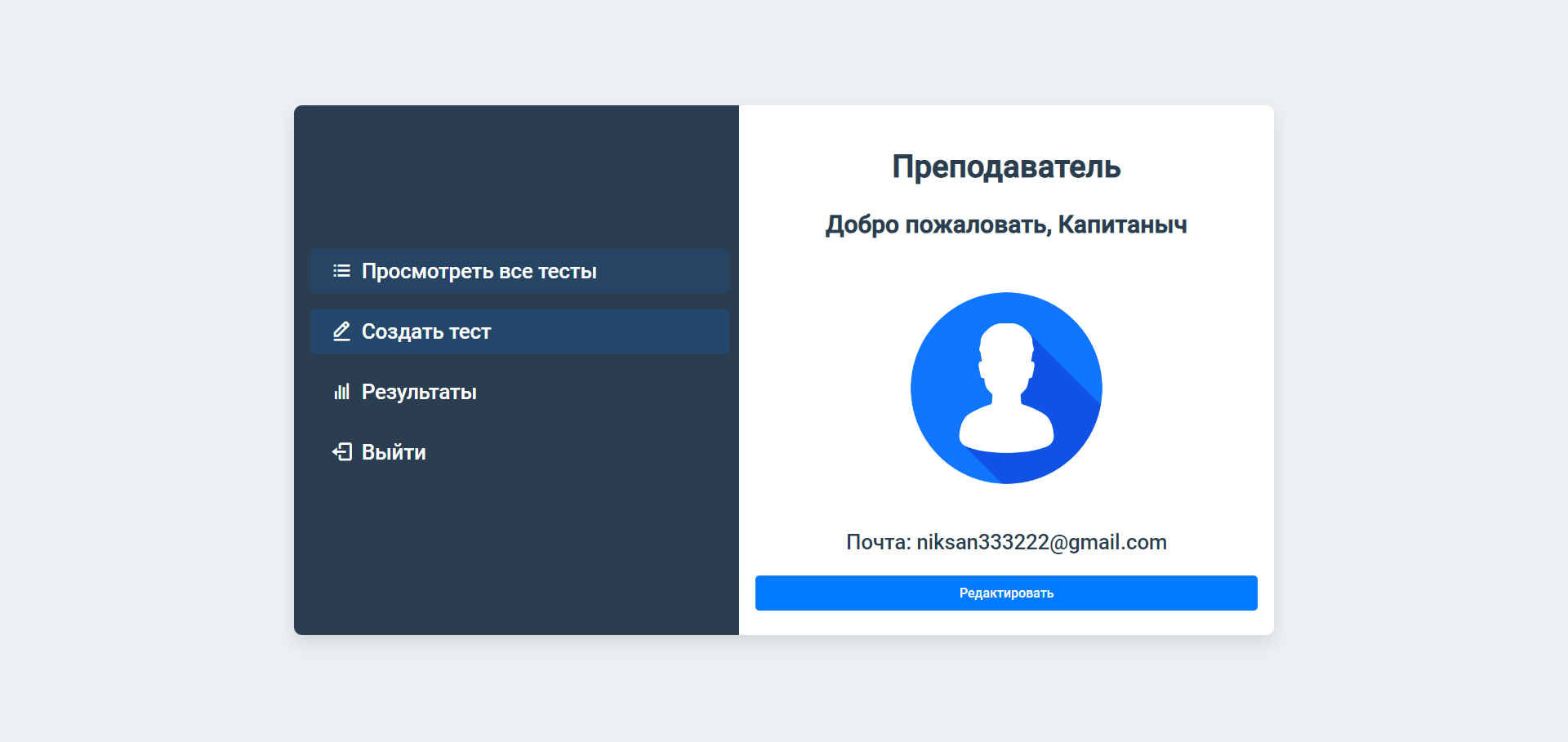
Итоговый интерфейс приложения, включая переработку страниц прототипа:****

Рис. 15 - Меню для роли «Преподаватель»

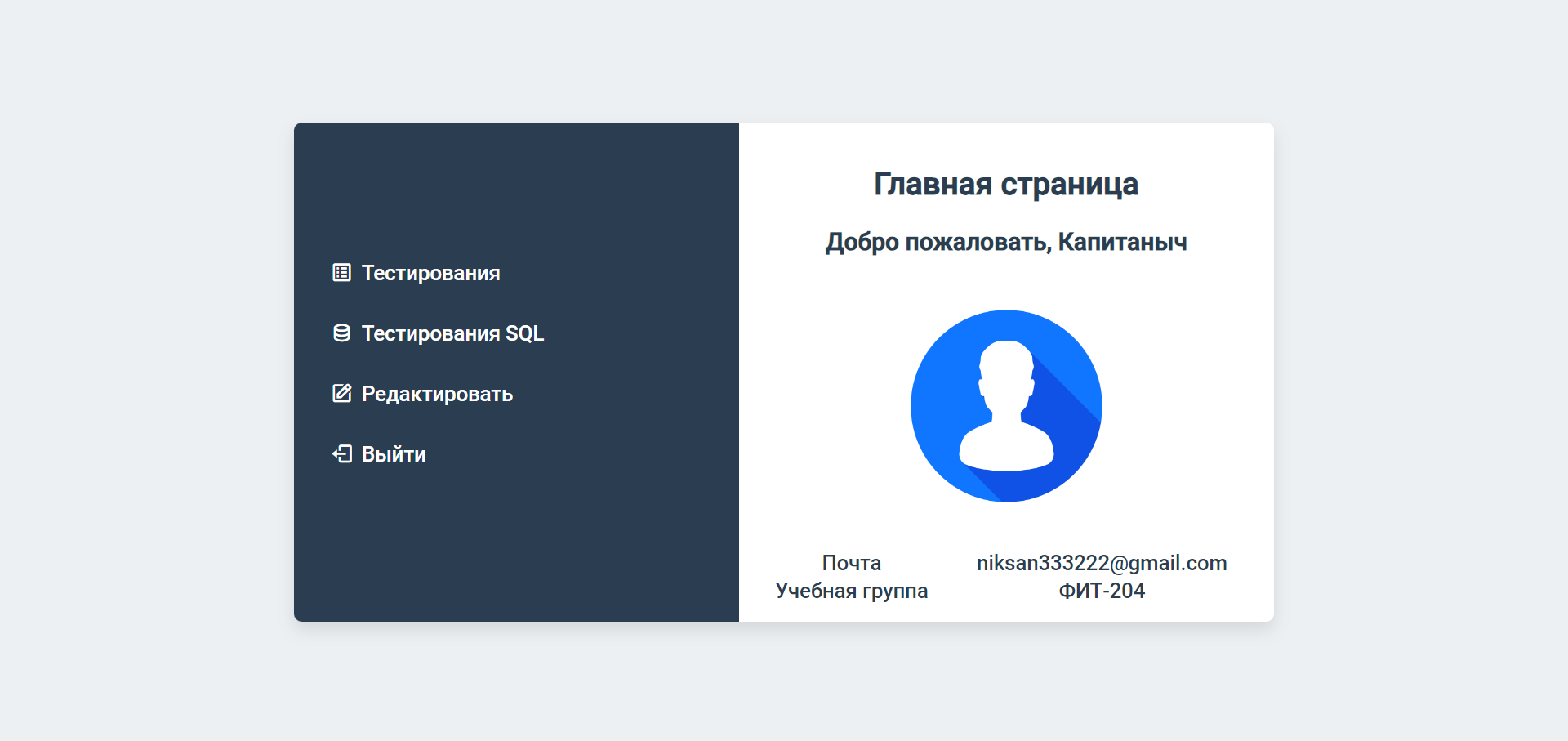
****

Рис. 16 - Меню для роли «Студент»

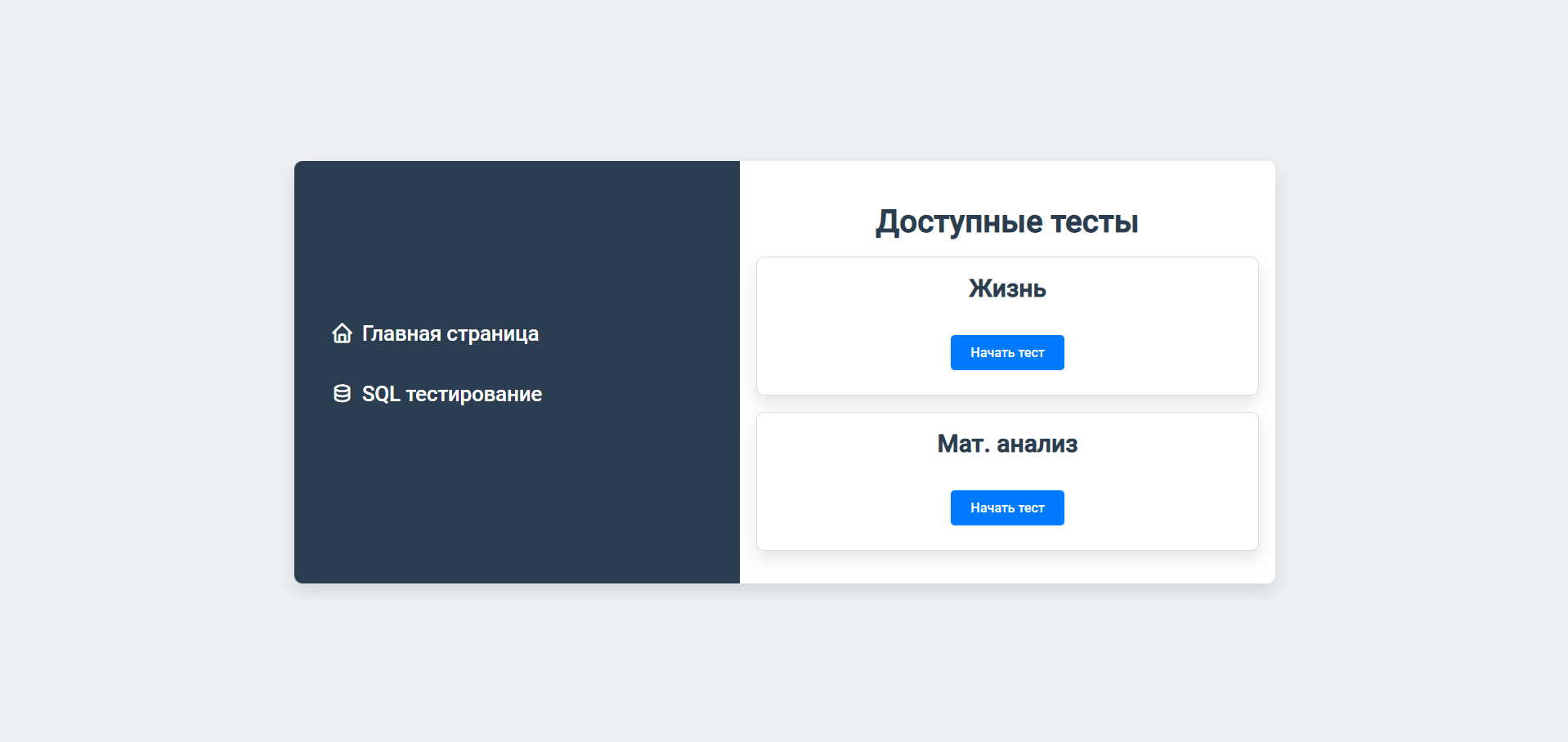
****

Рис. 17 - Выбор теста для начала тестирования «Студента»

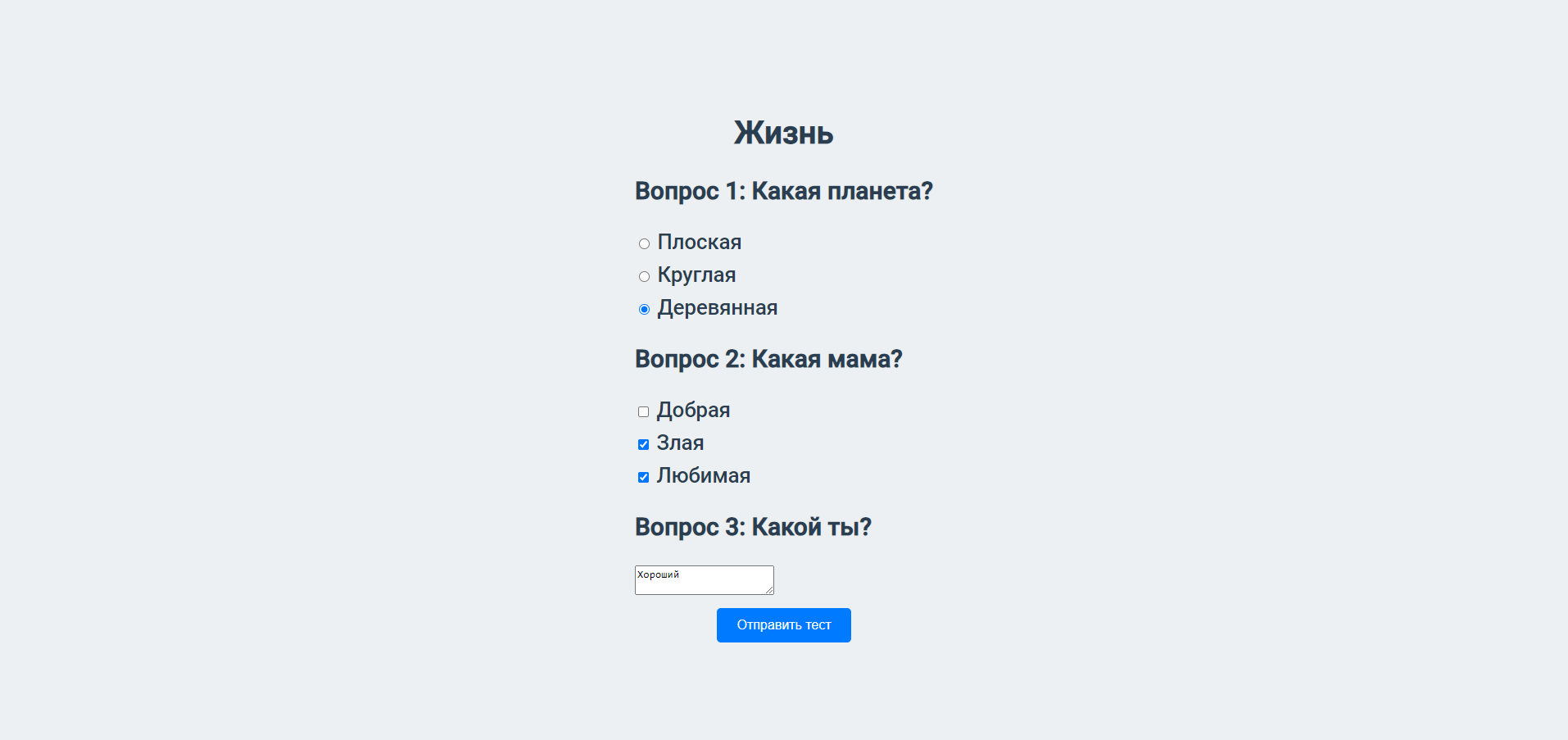
****

Рис. 18 - Прохождения тестирования, 3 типа вопроса

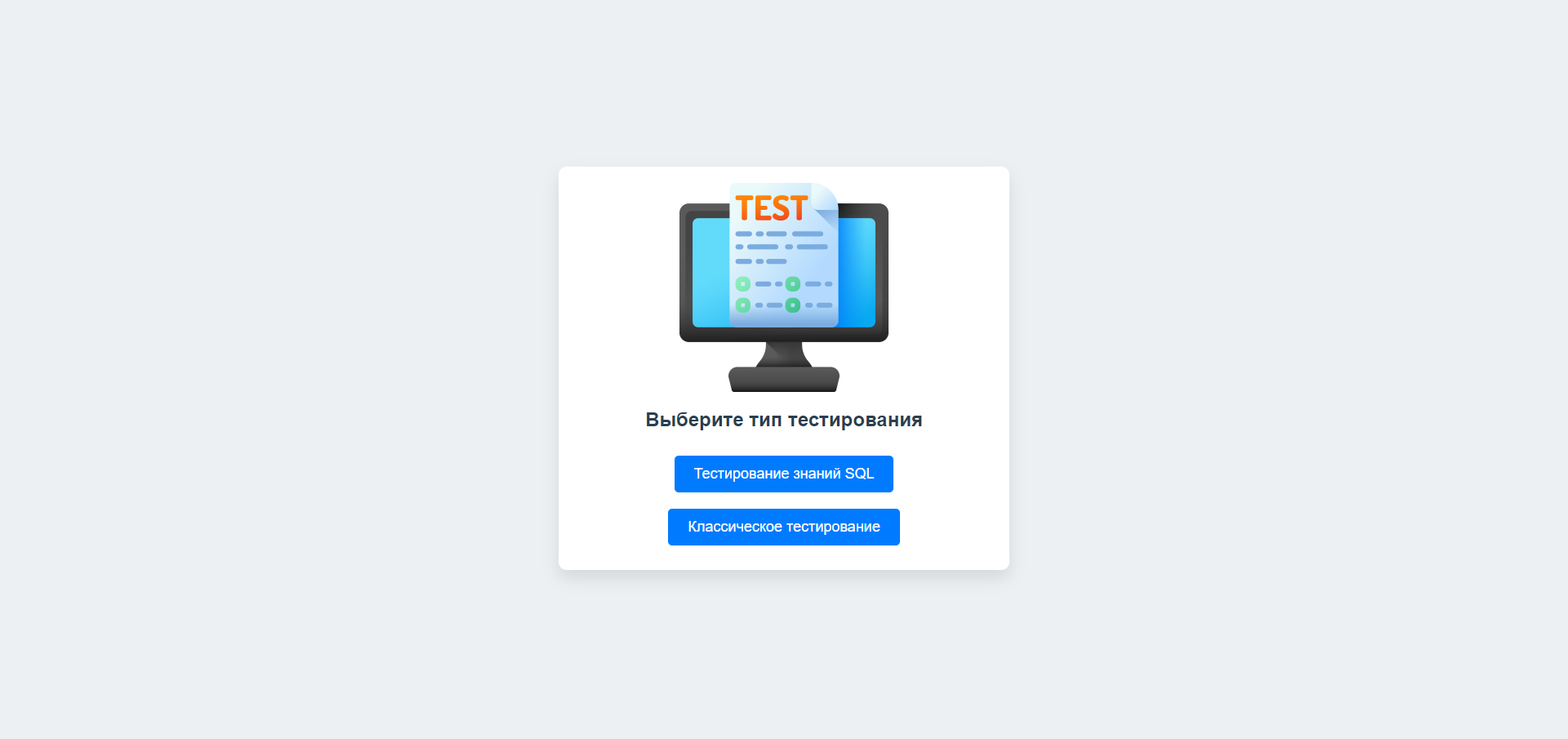
****

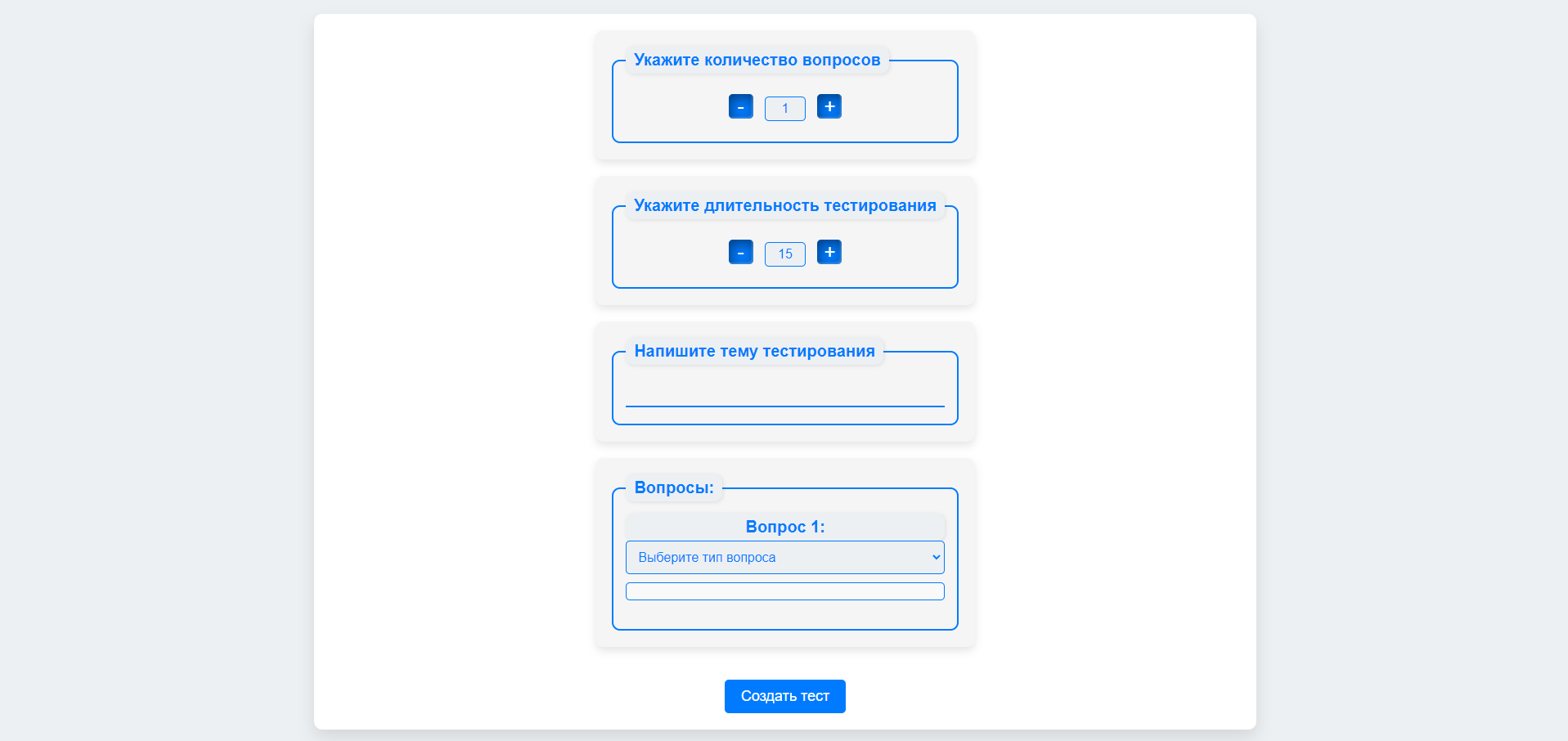
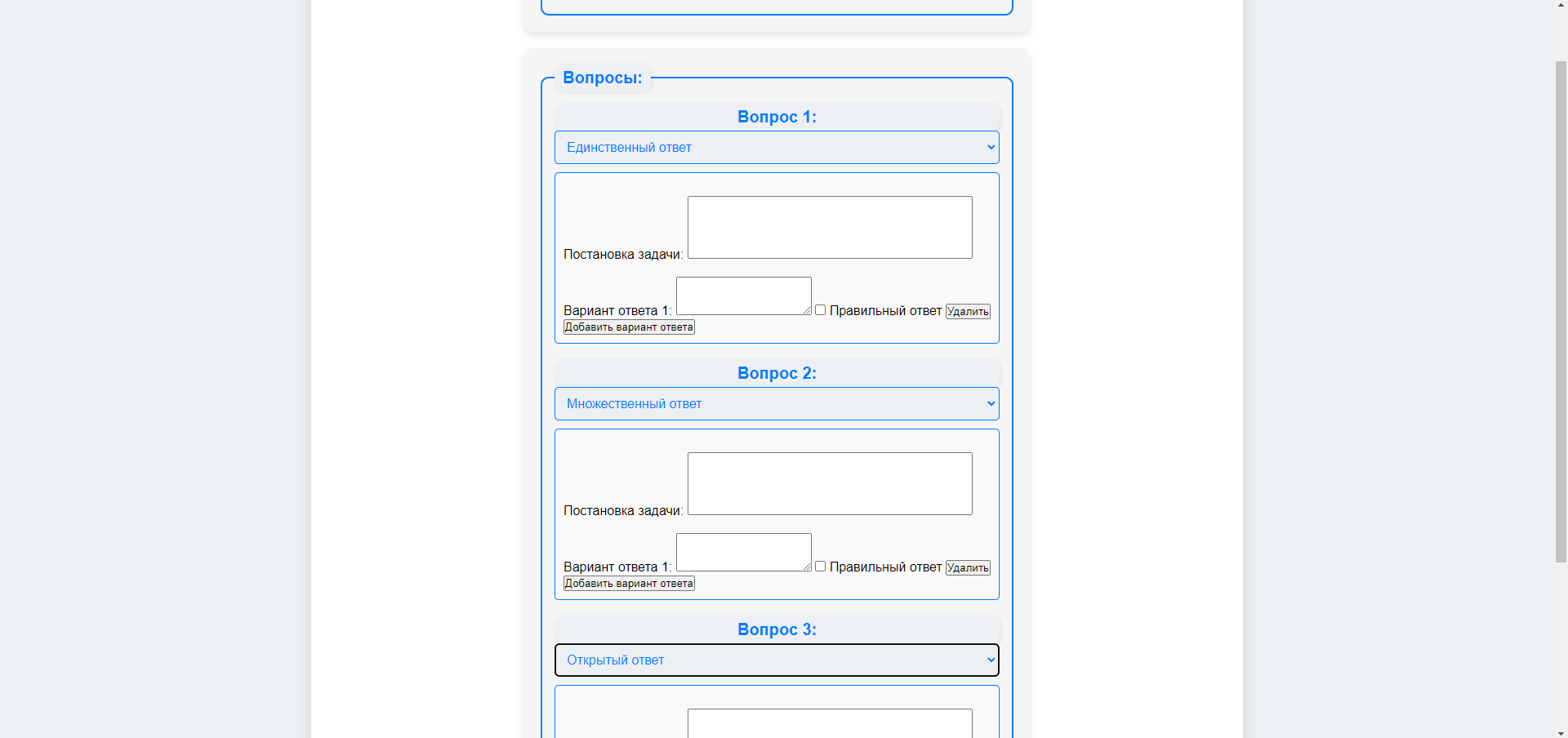
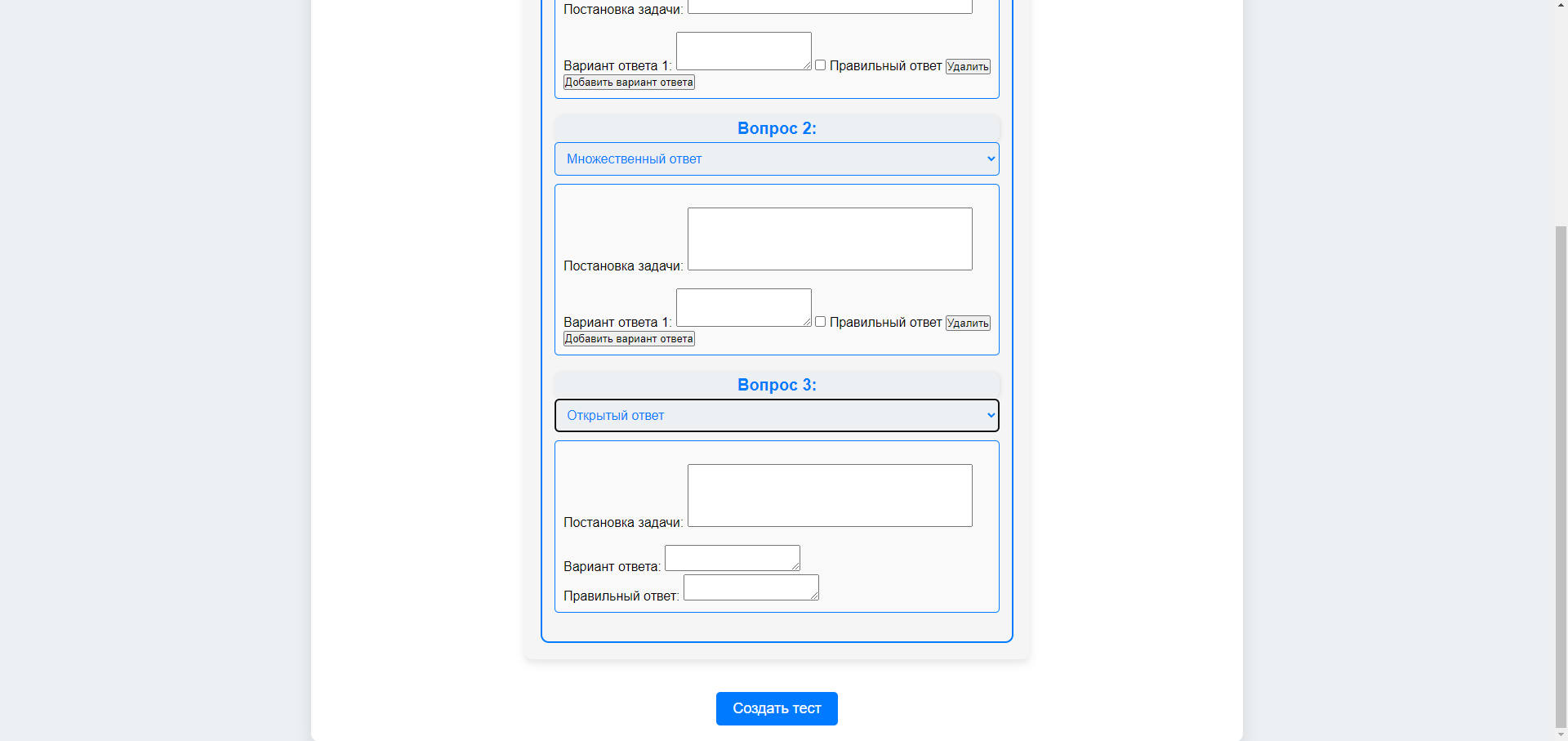
Рис. 19 - Выбор типа тестирования при создании для роли «Преподаватель»****

Рис. 20 - Конструктор классического тестирования для роли «Преподаватель»

****Рис. 21 - Вопрос «Единственный ответ» и «Множественный ответ»****Рис. 22 - Вопрос «Открытый ответ»

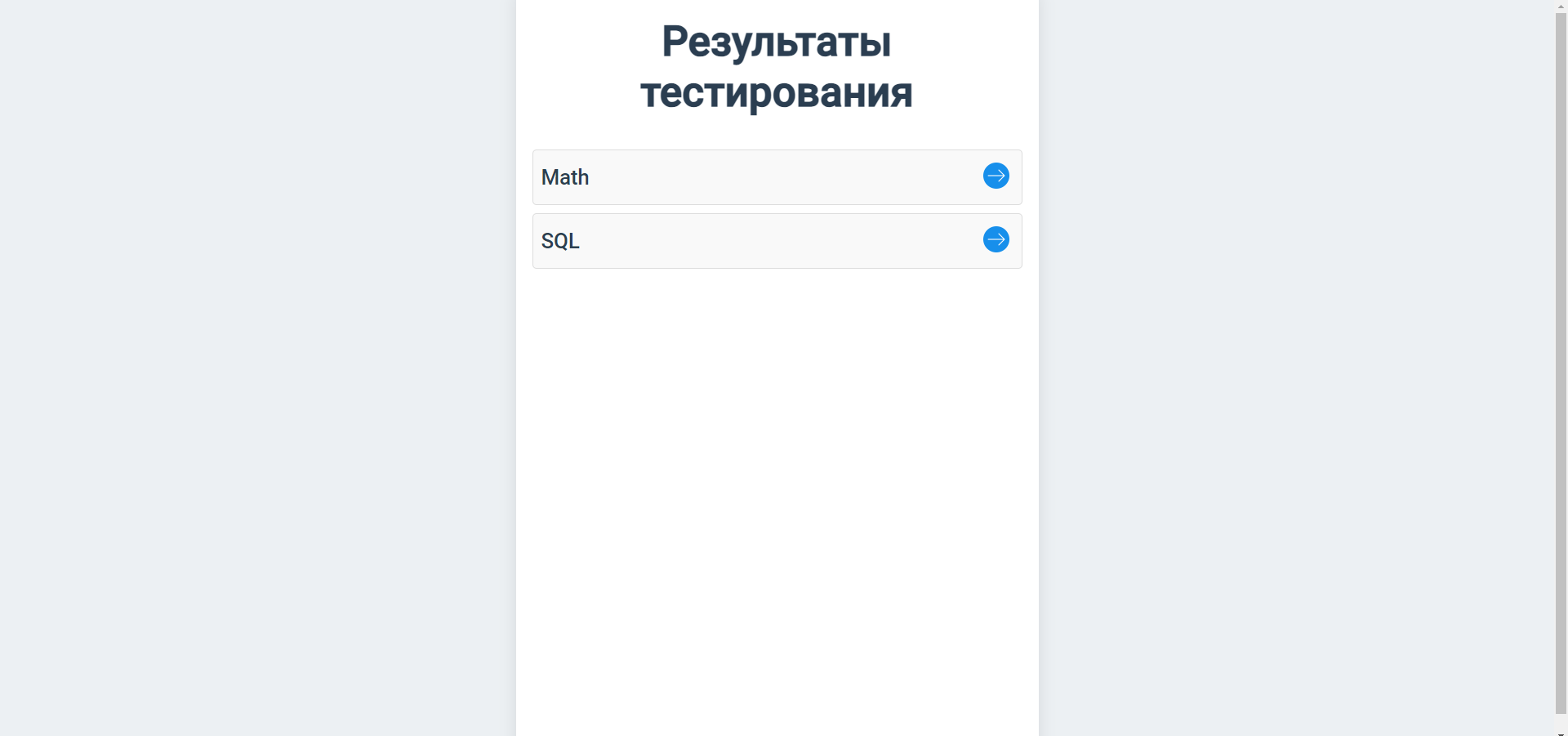
****

Рис. 23 - Просмотр тем результатов тестирования, которые принадлежат преподавателю.

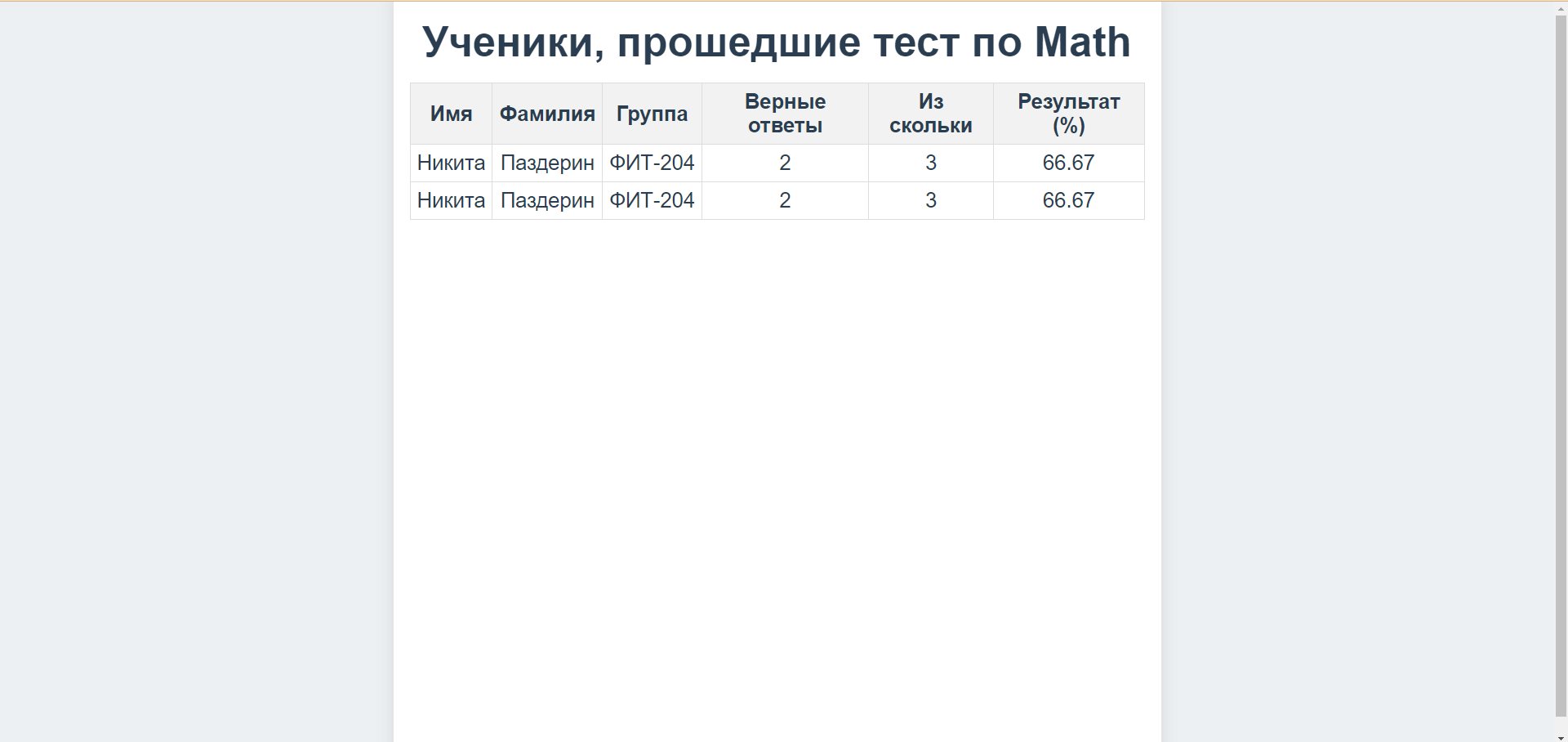
****

Рис. 24 - Просмотр результатов тестирования выбранной темы.

## 

## 3.4 Тестирование веб-сервисов

В рамках данного исследования было осуществлено ручное тестирование[28] веб-сервисов. Для этого был разработан и применен набор из 16 тестовых сценариев, описанных в форме чек-листа[28].

| Номер тестового сценария | Название сценария | Результат |
| --- | --- | --- |
| 1 | Получение результатов студентов преподавателем по выбранной дисциплине | Без ошибок |
| 2 | Получение всех дисциплин преподавателя, для проверки результата | Без ошибок |
| 3 | Получение преподавателем всех созданных им тестов | Без ошибок |
| 4 | Получение преподавателем выбранного теста | Без ошибок |
| 5 | Корректное отображение формы создания нового теста | Без ошибок |
| 6 | Корректное отображение классического конструктора тестов | Без ошибок |
| 7 | Создание нового теста преподавателем | Без ошибок |
| 8 | Изменение теста преподавателем | Без ошибок |
| 9 | Удаление теста преподавателем | Без ошибок |
| 10 | Перенаправление на страницу авторизации при попытке неавторизованного доступа | Без ошибок |
| 11 | Перенаправление на главную страницу студента при попытке обращения к маршрутам преподавателя | Без ошибок |
| 12 | Прохождение теста студентом | Без ошибок |
| 13 | Сохранение результата тестирования студентом | Без ошибок |
| 14 | Корректная проверка результатов | Без ошибок |
| 15 | Прохождение теста студентом не более 1 раза | Без ошибок |
| 16 | Перенаправление на главную страницу преподавателя, при попытке обращения к маршрутам преподавателя | Без ошибок |

Таблица 1. Тестирование веб-сервисов.

## 3.5 Тестирование пользовательского интерфейса

В рамках данного исследования было осуществлено ручное тестирование[28] веб-сервисов. Для этого был разработан и применен набор из 16 тестовых сценариев, описанных в форме чек-листа[29].

| Номер тестового сценария | Название сценария | Результат |
| --- | --- | --- |
| 1 | Корректное отображение дисциплин преподавателя на странице результатов | Без ошибок |
| 2 | Корректное отображение всех студентов с их результатами по выбранной дисциплине | Без ошибок |
| 3 | Корректное отображение созданных преподавателем тестов | Без ошибок |
| 4 | Корректное отображение страницы редактирования теста | Без ошибок |
| 5 | Возможность добавить вопрос | Без ошибок |
| 6 | Возможность удалить вопрос | Без ошибок |
| 7 | Возможность изменить вопрос | Без ошибок |
| 8 | Возможность изменить количество вариантов ответа | Без ошибок |
| 9 | Возможность изменить правильный ответ | Без ошибок |
| 10 | Возможность изменить тип вопроса | Без ошибок |
| 11 | Возможность удалить тест | Без ошибок |
| 12 | Корректное отображение страницы выбора конструктора теста | Без ошибок |
| 13 | Корректное отображение конструктора теста | Без ошибок |
| 14 | Возможность добавить вопрос | Без ошибок |
| 15 | Возможность удалить вопрос | Без ошибок |
| 16 | Возможность изменить вопрос | Без ошибок |
| 17 | Возможность изменить количество вариантов ответа | Без ошибок |
| 18 | Возможность изменить правильный ответ | Без ошибок |
| 19 | Возможность изменить тип вопроса | Без ошибок |
| 20 | Корректное отображение всех доступных тестов студенту | Без ошибок |
| 21 | Корректное отображение страницы прохождения теста | Без ошибок |
| 22 | Возможность завершить тест | Без ошибок |
| 23 | Возможность автоматического завершения теста после окончания времени | Без ошибок |

Таблица 2. Тестирование пользовательского интерфейса.

# ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы (далее - ВКР) реализована ИС, включающая в себя веб-сервисы для ролей «Студент», «Преподаватель» и веб-сервисы общие для этих ролей. Реализована клиентская часть.

Веб-приложение предоставляет следующие функции:

* авторизацию,
* регистрацию студента,
* функции по работе с тестами,
* функции по работе с конструктором тестов,
* функции по работе с профилями пользователей,
* функции по управлению данными профиля.

В ходе ВКР Паздерин Н.И. выполнил следующие задачи:

* выполнен обзор систем, осуществляющих автоматическое тестирование. Сформирован вывод об особенностях этих систем;
* построены диаграммы в нотации UML: диаграммы последовательности и диаграмма развёртывания;
* доработаны три коллекции для хранения данных в документоориентированной СУБД MongoDB;
* реализованы общие веб-сервисы для пользователей;
* реализованы веб-сервисы для студента;
* реализованы веб-сервисы для преподавателя;

В ходе ВКР Тимошенко Н.Р. выполнил следующие задачи:

* выполнен обзор систем, осуществляющих автоматическое тестирование. Сформирован вывод об особенностях этих систем;
* построены диаграммы в нотации UML: диаграммы вариантов использования;
* построена диаграмма данных ;
* произведена вёрстка визуальной составляющей ИС;
* реализованы JS-скрипты для визуальных эффектов на веб-странице.

# 

# СПИСОК ЛИТЕРАТУРЫ

1. Чистов, Д. В. Проектирование информационных систем: учебник и практикум для вузов / под общей редакцией Д. В. Чистова. — Москва: Издательство Юрайт, 2022. — 258 с. — (Высшее образование). — ISBN 978-5-534-00492-2. [Электронный ресурс]. Режим доступа: https://urait.ru/bcode/489307 (дата обращения: 26.12.2023).
2. Система компьютерного адаптивного тестирования (СКАТ) [Электронный ресурс]. Режим доступа: https://xiais.kemsu.ru/tests/?backToNewEios=https://eios.kemsu.ru/a/eios/personal-area (Дата обращения 12.11.2023).
3. Информационная система контроля уровня знания SQL [Локальный ресурс]. (дата обращения 01.10.2023)
4. Socrative [Электронный ресурс]. Режим доступа: https://b.socrative.com/teacher/#launch (Дата обращения 12.11.2023).
5. Визуальная студия тестирования [Локальный ресурс]. Режим доступа для развертывания: https://www.mmis.ru/programs/vts (Дата обращения 12.11.2023).
6. Moodle (СКАТ) [Электронный ресурс]. Режим доступа: https://moodle.org/ (Дата обращения 12.11.2023).

1. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0, 3-е изд.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2008. — 816 с.: ил. — Парал. тит. англ.

1. Тесты и решения [Электронный ресурс]. Режим доступа: https://yandex.ru/support/contest-management/problem/tests.html (дата обращения: 01.01.2024). – Текст: электронный.

1. Грекул, В.И. Проектирование информационных систем. Практикум: Учебное пособие / В.И. Грекул, Н.Л. Коровкина, Ю.В. Куприянов — М.: Национальный Открытый Университет «ИНТУИТ» 2012. — 187 с., ил. — (Серия «Основы информационных технологий»).

1. Чистов, Д. В. Проектирование информационных систем: учебник и практикум для вузов / под общей редакцией Д. В. Чистова. — Москва: Издательство Юрайт, 2022. — 258 с. — (Высшее образование). — ISBN 978-5-534-00492-2. [Электронный ресурс]. Режим доступа: https://urait.ru/bcode/489307 (дата обращения: 26.12.2022).

1. Соммервилл, Иан. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 624 с.: ил. – Парал. тит. англ.

1. Буч, Г. Язык UML. Руководство пользователя. / Г. Буч, Д. Рамбо, И. Якобсон, 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.: ил.

1. Онлайн-руководство по MongoDB [Электронный ресурс]. Режим доступа: https://metanit.com/nosql/mongodb/ (Дата обращения 19.12.2024).

1. Паттерн MVC [Электронный ресурс]. Режим доступа: https://metanit.com/web/nodejs/7.1.php (дата обращения: 04.01.2024).

1. Документация - Node.js [Электронный ресурс]. Режим доступа: https://nodejs.org/ru/docs (Дата обращения 05.01.2024).

1. Документация Node.js - Node.js с примерами кода [Электронный ресурс]. Режим доступа: https://nodejsdev.ru/guides/webdraftt/ (Дата обращения 05.01.2024).

1. Express 4.x - API Reference [Электронный ресурс]. Режим доступа: https://nodejsdev.ru/guides/webdraftt/ (Дата обращения 02.12.2024).

1. Documentation for the Oracle Database Node.js Add-on [Электронный ресурс]. Режим доступа: https://node-oracledb.readthedocs.io/en/latest/ (Дата обращения 04.11.2023).

1. Multer - API Reference [Электронный ресурс]. Режим доступа: https://github.com/expressjs/multer#api (Дата обращения 04.01.2024).

1. Модуль Nodemailer [Электронный ресурс]. Режим доступа: https://nodejsdev.ru/guides/webdraftt/email/ (Дата обращения 04.01.2024).
2. Express session [Электронный ресурс]. Режим доступа: https://expressjs.com/en/resources/middleware/session.html (Дата обращения 04.01.2024).

1. jQuery API Documentation [Электронный ресурс]. Режим доступа: https://api.jquery.com/category/selectors/ (Дата обращения 04.01.2024).

1. Что такое jQuery и как его использовать [Электронный ресурс]. Режим доступа: https://www.mousedc.ru/learning/122-chto-kak-jquery/ (Дата обращения 04.01.2024).
2. bcrypt for Node.js [Электронный ресурс]. Режим доступа: https://github.com/kelektiv/node.bcrypt.js (Дата обращения 05.01.2024).
3. Harihara Subramanian, Pethuru Raj, Hands-On RESTful API Design Patterns and Best Practices — Бирмингем, Великобритания: Packt Publishing, 2019. — 378 с. — ISBN 978-1788992664.
4. EJS [Электронный ресурс]. Режим доступа: https://nodejsdev.ru/guides/metanit/express-ejs/
5. Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.
6. Куликов, С. C. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — 3-е изд. — Минск: Четыре четверти, 2020. — 312 с. ISBN 978-985-581-362-1.
7. Корбин Е.К., Оспищев А.А. Информационная система контроля уровня знания SQL / Материалы XVIII (L) Международной научной конференции студентов, аспирантов и молодых ученых, приуроченной к 50-летию КемГУ – г. Кемерово: Кемеровский государственный университет, 2023. – 129-131 с.