DOI 10.37882/2223-2966.2025.12.12

# ИНФОРМАЦИОННАЯ СИСТЕМА КОНТРОЛЯ УРОВНЯ ЗНАНИЙ ОБУЧАЮЩИХСЯ, ПРЕДОСТАВЛЯЮЩАЯ ВОЗМОЖНОСТИ ПРОВЕДЕНИЯ КЛАССИЧЕСКОГО ТЕСТИРОВАНИЯ ЗНАНИЙ И ПРОВЕРКУ ЗНАНИЯ SQL

## AN INFORMATION SYSTEM FOR MONITORING STUDENTS' KNOWLEDGE LEVELS THAT PROVIDES CAPABILITIES FOR CONDUCTING CLASSICAL KNOWLEDGE TESTING AND ASSESSING SQL PROFICIENCY

***S. Zavozkin***

*Summary*. In the context of digital education, the automation of learning efficiency assessment has become increasingly significant. An effective control system at all stages of education forms an essential component of the learning environment, improving the quality of professional training. Automation not only optimizes performance evaluation but also creates a productive feedback mechanism.

For the Institute of Digital Technologies, one core discipline is 'Databases.' Traditional testing tools fail to provide automated assessment of SQL programming skills. To address this limitation, a proprietary information testing system was developed, enabling both conventional knowledge testing and SQL proficiency evaluation across various relational databases. This paper presents the process of developing the proposed information system. As part of the study, a comparative analysis of eight existing testing systems was conducted, system design was carried out using UML notation, implementation tools were selected, and a proprietary information system was developed and prepared for pilot deployment.

*Keywords*: testing of students, automated knowledge assessment, SQL, information system.

***Завозкин Сергей Юрьевич***
*кандидат технических наук, доцент,*
*Федеральное государственное бюджетное образовательное учреждение высшего образования*
*Кемеровский государственный университет*
*shade_s@mail.ru*

*Аннотация*. В современных условиях цифрового обучения вопрос автоматизации контроля эффективности учебного процесса становится наиболее актуальным. Эффективная система контроля на всех этапах обучения становится неотъемлемой частью образовательной среды, обеспечивая повышение качества подготовки специалистов. Автоматизация этого процесса не только оптимизирует оценку достижений, но и способствует созданию эффективной системы обратной связи.

Для института цифры одной из базовых является дисциплина с названием «Базы данных». Классические системы тестирования не предоставляют возможности выполнить автоматизированное тестирование на знание языка программирования SQL. Для решения проблемы принято решение о разработке собственной информационной системы тестирования знаний обучающихся, предоставляющей возможности как проведения классического тестирования знаний, так и проверку знания языка SQL для различных реляционных баз данных.

В данной работе рассмотрен процесс создания разработанной информационной системы. В ходе исследования был проведен сравнительный анализ восьми существующих систем тестирования, выполнено проектирование с использованием диаграмм в нотации UML, определены инструменты для реализации, а также создана и подготовлена к тестовой эксплуатации собственная информационная система.

*Ключевые слова*: тестирование обучающихся, автоматизированная оценка знаний, SQL, информационная система.

Оценка качества усвоения учебного материала является важной частью образовательного процесса. Эффективно выстроенная система контроля знаний способствует повышению результативности учебной деятельности, формирует обратную связь со студентом и напрямую влияет на уровень профессиональной компетенции будущих специалистов. Такой подход дает возможность своевременно определять недочеты в подготовке обучающихся, оценивать эффективность педагогических методов и выстраивать приоритеты в организации учебного процесса. Существуют различные способы проверки знаний: устный опрос, индивидуальное задание, письменная работа, тестирование, проектная и исследовательская работа, дифференцированная проверочная работа и многое другое. Каждый из указанных способов имеет собственные преимущества и применяется в зависимости от целей и содержания учебного курса.

В условиях цифровизации образования контроль знаний все чаще реализуется посредством использования автоматизированных систем тестирования. Подобные платформы обеспечивают проведение проверки знаний с использованием компьютерных технологий, позволяя комбинировать различные типы заданий: множественный выбор, открытые ответы, сопоставление элементов и другие. Автоматизация процесса тестирования ускоряет обработку результатов, повышает объ-

ективность оценивания и создает условия для детального анализа уровня подготовки обучающихся, что делает контроль знаний более точным и эффективным инструментом образовательной деятельности.

В Кемеровском государственном университете (КемГУ) создана электронная информационно-образовательная среда (ЭИОС), представляющая собой совокупность цифровых ресурсов, программных инструментов, информационных и телекоммуникационных технологий и сервисов, предназначенных для обеспечения образовательного процесса. В состав ЭИОС входит система компьютерного адаптивного тестирования (СКАТ), осуществляющая автоматизацию процесса тестирования [2].

Система СКАТ поддерживает классические типы вопросов, включая: выбор одного или нескольких правильных ответов, заполнение пропусков словами, расположение элементов в заданной последовательности и другие подобные варианты. Однако функциональные возможности данной системы ограничены: она не предусматривает проверку навыков программирования и написания кода на различных языках. В результате студенты вынуждены обращаться к сторонним образовательным платформам и самостоятельно развивать практические умения программирования. У СКАТ есть ряд недостатков помимо этого:

— СКАТ устарел визуально, к тому же он 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.

При изучении учебной дисциплины «Базы данных» проверка уровня владения языком программирования SQL требует применения автоматизированной тестирующей системы, оснащенной функционалом для выполнения программного кода и анализа возможных ошибок при его выполнении. Такой подход способствует ускорению процесса обучения и повышению его эффективности. Однако, действующая система СКАТ не предусматривает реализацию данной функциональности.

Исходя из сказанного, цель данной работы — разработка информационной системы тестирования (далее ИС) знаний обучающихся, предоставляющей возможности как проведения классического тестирования знаний, так и проверку знания языка SQL для различных реляционных баз данных [3].

На стартовом этапе исследования был выполнен обзор и сравнительная оценка доступных информационных систем, ориентированных на проведение многоформатного тестирования обучающихся: «Система компьютерного адаптивного тестирования (СКАТ) КемГУ», «Яндекс.Контест» [4], «Codewars» [5], «HackerRank» [6], «Codecademy» [7], «Socrative» [8], «Визуальная студия тестирования» [9], «Moodle» [10].

По итогам рассмотрения возможностей и ограничений имеющихся информационных систем обоснована необходимость создания собственного решения в формате веб-приложения, реализующего взаимодействие с базой данных через специализированные веб-сервисы, что повышает удобство эксплуатации и обеспечивает масштабируемость архитектуры.

Далее были определены основные пользовательские и системные требования к создаваемому программному продукту [3]. Разрабатываемая информационная система тестирования знаний обучающихся предназначена для проведения классического тестирования и включает четыре типа заданий: с единственным выбором ответа, с множественным выбором, с открытым ответом и на проверку знаний языка SQL. Для всех видов пользователя доступны функции авторизации, редактирования личных данных и восстановления доступа к учетной записи посредством электронной почты.

Ключевые функции пользователя с ролью «Студент»:

1. получение информации о доступных тестах;
2. выполнение тестовых заданий по проверке знаний языка SQL;
3. выполнение классических тестовых заданий;
4. просмотр полученных результатов после завершения теста.

Ключевые функции пользователя с ролью «Преподаватель»:

1. формирование тестов в модуле конструктора SQL-заданий;
2. создание тестов в модуле стандартного тестирования, с возможностью: выбора типа задания; добавления формулировки вопроса; внесения вариантов ответов при необходимости; указания корректных ответов;
3. редактирование и удаление ранее созданных тестов;
4. назначение тестов студентам или группам студентов;
5. настройка параметров проведения тестирования;
6. анализ и просмотр полученных результатов тестирования.

Ключевые функции пользователя с ролью «Администратор»:

1. управление ролями пользователей;
2. управление учетными записями пользователей: создание, удаление, блокировка.

Системные требования:

1. использование хорошо масштабируемой базы данных;
2. использование реляционной базы данных для проверки знания языка SQL;
3. интеграция с ИС «Рейтинг обучающихся» КемГУ;
4. наличие сервера приложений, предназначенного для размещения и обработки веб-сервисов;
5. использование «тонкого» клиента;
6. применение архитектурного подхода REST API при организации обмена данными.

В разработанной ИС реализована архитектура, основанная на сервисно-ориентированном подходе. Система включает веб-приложение, обмен данными которого с базой данных осуществляется посредством разработанных REST-сервисов. Структура программного решения представлена на рисунке 1 в форме диаграммы развертывания, выполненной в нотации UML.

Разрабатываемая ИС для хранения данных использует документно-ориентированную систему управления базами данных. Основные коллекции: Users, replyStud, Tests. 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.

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,
— token_pass — токен пароля,
— about_test — 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,
— problemPreview — массив, хранящий дополнительную информацию к самой задаче,
— problemSolution — массив, хранящий пути к скриптам с правильным ответом,
— scriptTable — массив объектов. Параметр path хранит путь к скрипту, который генерирует таблицу. Параметр name хранит имя скрипта. Параметр table_name хранит имя таблицы.
— scriptTableData — массив объектов. Параметр path хранит путь к скрипту, который генериру-

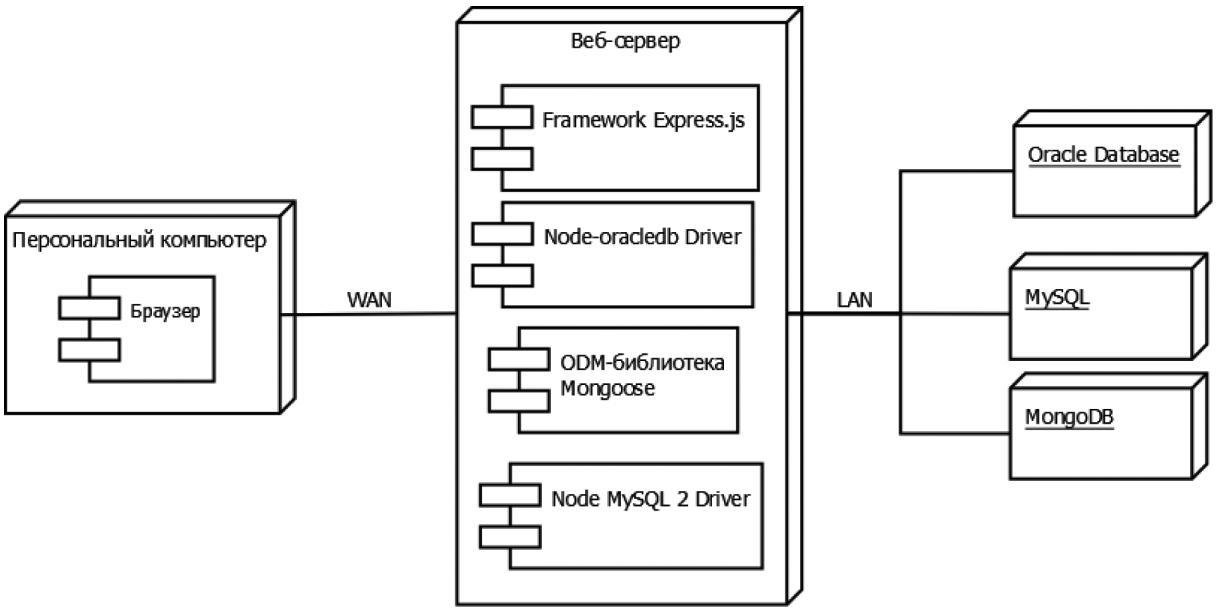


Рис. 1. Диаграмма развёртывания

ет данные в таблице. Параметр name хранит имя скрипта.
- image — массив, хранящий пути к картинкам таблиц,
- token_test — токен теста.

Коллекция replyStud хранит историю с информацией о результатах тестирования для конкретного студента:
- _id — идентификатор тестирования, первичный ключ,
- stud_id — идентификатор студента
- 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.

Для ИС был построен комплекс моделей в нотации UML, включая диаграмму вариантов использования, диаграммы последовательности действий, диаграмму классов, диаграмму развертывания. На рисунке 2 представлена диаграмма вариантов использования для сценария «авторизация студента».

Программные средства реализации информационной системы:
- JavaScript — язык программирования.
- Node.js — кроссплатформенная среда выполнения JavaScript с открытым исходным кодом.
- Express.js –минималистичный фреймворк для Node.js, упрощающий создание веб-приложений и REST API.
- Node-oracledb — официальный драйвер для подключения Node.js-приложений к базам данных Oracle.
- MongoDB — документо-ориентированная NoSQL СУБД, хранящая данные в виде BSON-документов.
- Oracle Database — промышленная реляционная СУБД.
- 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

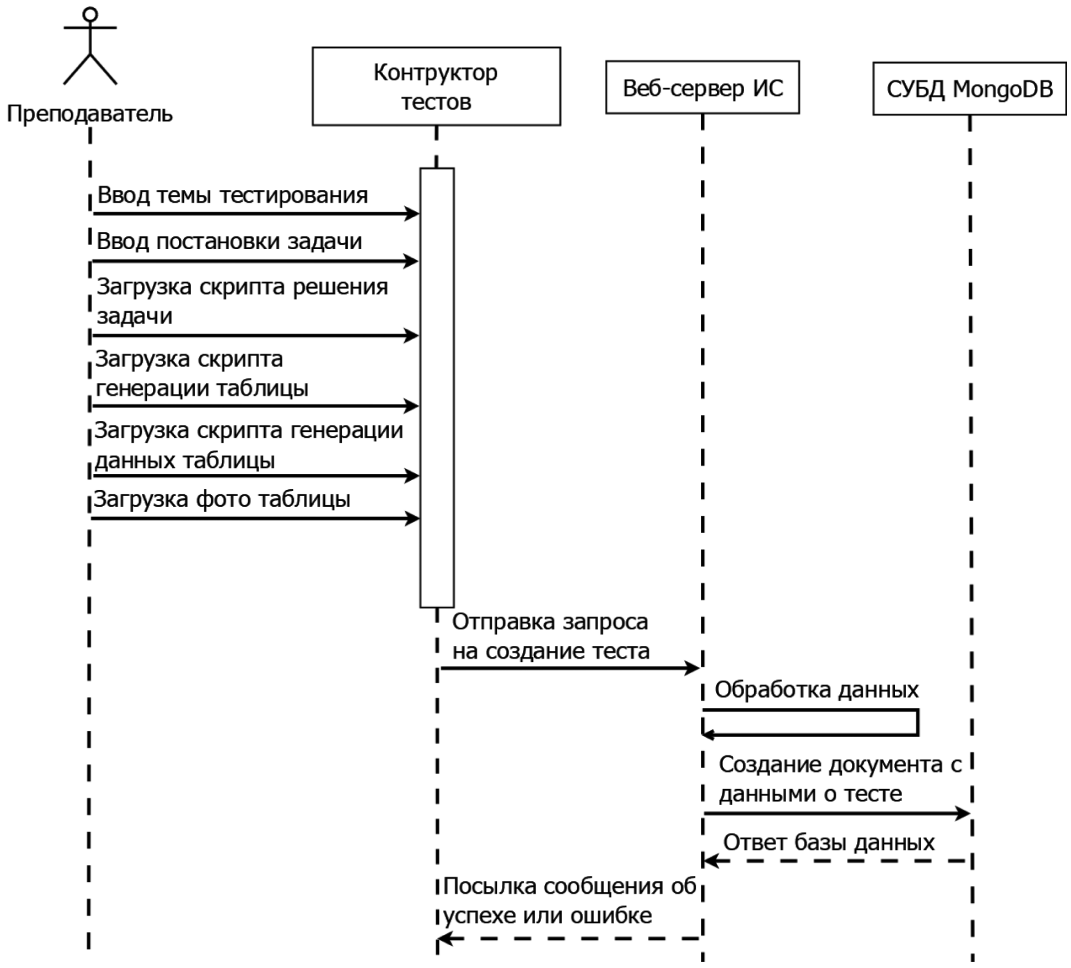


Рис. 2. Диаграмма последовательности для сценария «процесс создания задачи преподавателем»

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.

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.

Информационная система состоит из двух основных компонентов: Front-end и Back-end. В Front-end используется шаблонизатор EJS 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), который направлен на создание гибких и легко поддерживаемых программных систем.

Система разделена на несколько архитектурных слоев [11]:

— 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.

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.

На рисунке 3 представлен пример интерфейса для создания задачи на знание SQL для теста.

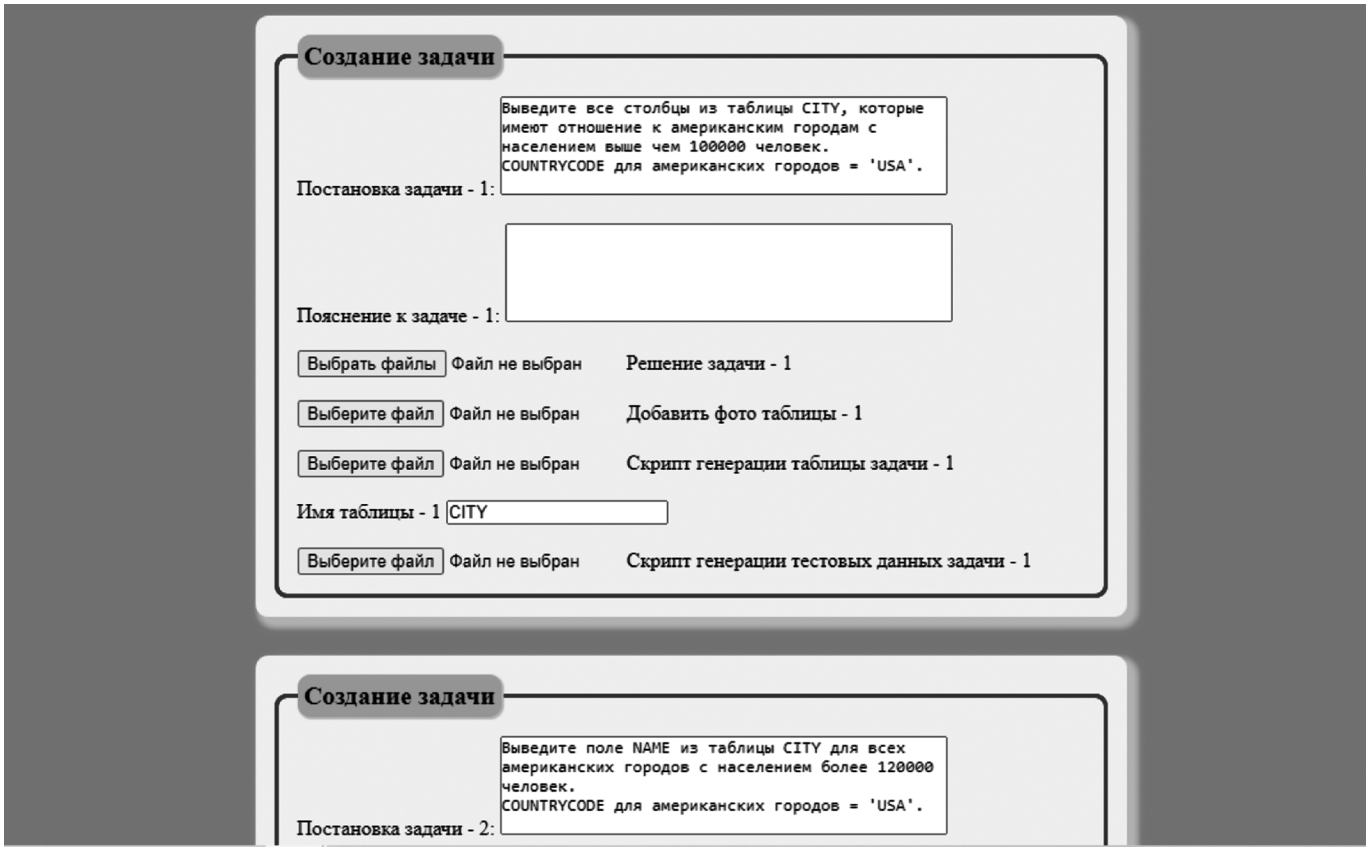


Рис. 3. Создание задач на знание SQL для теста

На данный момент система реализована и планируется к дальнейшему внедрению в тестовую эксплуатацию в КемГУ. В перспективе предусмотрена интеграция разработанной ИС с ИС КемГУ «Рейтинг обучающегося», что позволит автоматически передавать результаты тестирования в ИнфОУПро для их последующей обработки и учета.

ЛИТЕРАТУРА

1. Горшенева И.А. Функции и формы контроля как важнейшего компонента учебно-воспитательного процесса / И.А. Горшенева, А.К. Буравлева, Е.А. Буравлев // Вестник экономической безопасности. — 2018. — №4. — С. 301–304.
2. Система компьютерного адаптивного тестирования (СКАТ) [Электронный ресурс] // URL: https://xiais.kemsu.ru/tests/?backToNewEios=https://eios.kemsu.ru/a/eios/personal-area (Дата обращения 30.10.2025).
3. Завозкин С.Ю. (науч. рук), 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. — С.129–131.
4. Яндекс Контест. Тесты и решения [Электронный ресурс] // URL: https://yandex.ru/support/contest-management/problem/tests.html (дата обращения: 30.10.2025).
5. Codewars / The Codewars Docs [Электронный ресурс] // URL: https://docs.codewars.com/ (дата обращения: 30.10.2025).
6. HackerRank — Online Coding Tests and Technical Interviews [Электронный ресурс] // URL: https://www.hackerrank.com/ (дата обращения: 30.10.2025).
7. Using Codecademy — Codecademy Help Center [Электронный ресурс] // URL: https://help.codecademy.com/hc/en-us/categories/202886527-Using-Codecademy (дата обращения: 30.10.2025).
8. Socrative [Электронный ресурс] // URL: https://b.socrative.com/teacher/#launch (Дата обращения 30.10.2025).
9. Визуальная студия тестирования [Электронный ресурс] URL: https://www.mmis.ru/programs/vts (Дата обращения: 30.10.2025).
10. Moodle (СКАТ) [Электронный ресурс] URL: https://moodle.org (Дата обращения: 30.10.2025).
11. Мартин Р.С. Чистая архитектура. Искусство разработки программного обеспечения / Р.С. Мартин // Питер. — 2022. — 352 с.


Журнал «Современная наука: актуальные проблемы теории и практики»