УДК 004.9

РАЗРАБОТКА МОДУЛЯ РЕШЕНИЯ АЛГОРИТМИЧЕСКИХ ЗАДАЧ ДЛЯ СИСТЕМЫ КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ СТУДЕНТОВ LMSDOT

Клочков Даниил Сергеевич[1], Дедович Татьяна Григорьевна[2], Смирнов Даниил Павлович[3]

[1]Студент;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: danil.klochkov.92@bk.ru.

[2]Старший научный сотрудник;
Объединенный институт ядерных исследований;
Россия, 141980, Московская обл., г. Дубна, ул. Жолио-Кюри, 6;
Кандидат физико-математических наук, доцент;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: tdedovich@yandex.ru.

[3]Аспирант;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: smirnov@lmsdot.com

В работе проанализированы популярные платформы, позволяющие решать алгоритмические задачи. Выбраны технологии реализации модуля и изучена структура приложения «Lmsdot». Созданы диаграмма последовательности и диаграмма классов модуля. Спроектирована база данных. Модуль решения алгоритмических задач реализован с помощью фреймворков Elysia.js и Next.js. Произведена интеграция модуля с уже существующими серверами и базой данных в приложении «Lmsdot». Проведено тестирование модуля и представлены рекомендации для дальнейшего масштабирования и улучшения работы модуля.

Ключевые слова: электронное обучение, *LMS* системы, образование, тестирование алгоритмических задач.

Для цитирования:

Клочков Д. С., Дедович Т. Г., Смирнов Д. П. Разработка модуля решения алгоритмических задач для системы контроля и оценки знаний студентов Lmsdot // Системный анализ в науке и образовании: сетевое научное издание. 2024. № 4. С. 57-67. EDN: IMMCUU. URL: https://sanse.ru/index.php/sanse/article/view/636.

DEVELOPMENT OF A MODULE FOR SOLVING ALGORITHMIC PROBLEMS FOR MONITORING AND ASSESSING STUDENTS' KNOWLEDGE LMSDOT SYSTEMS

Klochkov Daniil S.[1], Dedovich Tatyana G.[2], Smirnov Daniil P.[3]

[1]Student;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: danil.klochkov.92@bk.ru.

[2]Senior Researcher;
Joint Institute for Nuclear Research;
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;
PhD in Physical and Mathematical Sciences, associate professor;
Dubna State University;

19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: tdedovich@yandex.ru.

[3]PhD student;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: smirnov@lmsdot.com

The presented work describes the creation of an algorithmic problem-solving module that will be integrated into the «Lmsdot» application. In the work, popular platforms for solving algorithmic problems are analyzed. Technologies for implementing the module have been selected and the structure of the «Lmsdot» application has been studied. A sequence diagram and a class diagram of the module have been created. The database structure has been designed. The algorithmic problem-solving module is implemented using the Elysia.js and Next.js frameworks. Integration with existing servers and databases in the «Lmsdot» application has been carried out. The module has been tested and recommendations for further scalability and improvement of the module's performance have been made.

Keywords: e-learning, LMS systems, education, platforms for solving algorithmic problems.

For citation:
Klochkov D. S., Dedovich T. G., Smirnov D. P. Development of a module for solving algorithmic problems for monitoring and assessing students' knowledge Lmsdot systems, *System analysis in science and education*, 2024;(4):57-67 (in Russ). EDN: IMMCUU. Available from: https://sanse.ru/index.php/sanse/article/view/636.

Введение

Системы управления обучением *LMS* (от англ. *Learning Management System*) позволяют администрировать и автоматизировать процесс обучения, хранить материалы курсов и собирать статистику успеваемости студентов. В наше время такой формат обучения получил огромную популярность. Системы управления обучением делают работу преподавателя более структурированной и удобной, а отслеживание прогресса студента более прозрачным.

Современные *LMS* системы должны удовлетворять требованиям, которые возникают в образовательном процессе, в том числе, предоставлять удобный интерфейс для решения алгоритмических задач. Студенты, обучающиеся по специализации, связанной с информационными технологиями (ИТ), изучают курсы, на которых необходимо решать алгоритмические задачи. Данная тема является настолько важной, что при приеме на работу ИТ компании используют такого рода задачи на собеседованиях. Существует огромное количество интернет-ресурсов, предоставляющих возможность решения алгоритмических задач с последующим тестированием. Все они имеют свои системы для проверки, а также уникальные пользовательские интерфейсы и разнообразное представление задач. Однако, рядовому студенту при решении схожих университетских заданий приходится компилировать решение на локальной машине, прежде чем отсылать задания на проверку.

В Университете «Дубна» преподавание курсов «Дискретная математика», «Математическая логика и теория алгоритмов», «Теория автоматов и формальных языков» и «Теория языков программирования и методов трансляции» включает изучение алгоритмов для решение алгоритмических задач. Алгоритмы реализуются в виде программного кода с помощью языков программирования. Для преподавания этих курсов используется система «*Lmsdot*» [1], которая не имеет модуля решения алгоритмических задач. Вместо него используется сторонний ресурс «Информатикс» [2]. Разрабатываемый модуль должен позволять преподавателям создавать алгоритмические задачи, тесты для автоматической проверки, просматривать результаты тестов, решение студента, представленного в виде программы на выбранном языке программирования и выставить оценку. Для студентов модуль должен предоставлять возможность написания кода программы в удобном интерфейсе и просматривать оценки, выставленные преподавателем.

1. Анализ платформ для решения алгоритмических задач

Существует множество платформ для решения алгоритмических задач. Для сравнительного анализа были выбраны наиболее популярные системы в мире «*LeetCode*» [3] и «*Codeforces*» [4], а также в России «Информатикс» [2] и «Яндекс.Контест» [5]. Информация пользовательских предпочтениях получена на основе данных с ресурса «*SimilarWeb*» [6]. «*LeetCode*» используется для подготовки и проведения собеседований по программированию, а «*Codeforces*» для проведений соревнований по программированию. «Информатикс» – отечественная платформа для дистанционной подготовки по информатике, которая предлагает разнообразные курсы, задачи и тесты для учеников. «Яндекс.Контест» используется для решения задач по программированию в виде индивидуальных и командных соревнований.

Данные системы исследовались по следующим характеристикам: возможность создания и проверки задач преподавателем, установка сложности и тега задачи, наличие редактора для написания программ, возможность добавления материала для обучения и удобство навигации. Баллы выставлялись по следующей шкале оценок: «1» – соответствует критерию, «0.5» – частично соответствует критерию и «0» – не соответствует критерию. Результаты анализа представлены в табл. 1.

Табл. 1. Сравнительный анализ платформ

	«*LeetCode*»	«*Codeforces*»	«Информатикс»	«Яндекс.Контест»
Создание задач	1	0	1	0
Проверка задач	0	0	1	0
Сложность задач	1	1	0	0
Теги задач	1	0	0	0
Редактор	1	0.5	0	0.5
Материалы	0.5	0	1	0
Удобство	1	0.5	0.5	0.5

Создание задач преподавателем на платформах «*Codeforces*» и «Яндекс.Контест» невозможно, так как задачи создает определенная группа лиц, готовящая соревнование. В «*Codeforces*» во время соревнования задачи обозначаются буквами латинского алфавита (*A, B, C, D, E, F, G, H, I*) и сложность задач возрастает от *A* до *I*. В «*LeetCode*» для этого существует отдельный параметр. Описание типа алгоритма в «*LeetCode*» указывается отдельным тегом, а в остальных системах отсутствует. Редактор для написания программ в «*LeetCode*» имеет подсветку синтаксиса, в «Яндекс.Контест» имеет форму текстового поля, в «*Codeforces*» используется плагин для *IntelliJ IDEA*, а в «Информатикс» решение загружается в виде файла с кодом. Материалы для обучения на платформе «*LeetCode*» представлены в виде курсов, разбитых по главам, но доступ к ним можно получить только на платной основе. На платформе «Информатикс» материалы представлены для каждого раздела на бесплатной основе.

Из табл. 1 видно, что платформа «*LeetCode*» имеет преимущество по большинству характеристик. Но использовать ее для обучения затруднительно, так как преподаватель не видит решения студентов и не может оценить их работы. Эта платформа подходит больше для индивидуального обучения. Система «Информатикс» предоставляет преподавателю возможность создания и проверки работ студентов, но имеет ряд недостатков. К ним относятся невозможность установки рейтинга задач по сложности и тегов. Кроме этого, если преподаватель работает в системе «*Lmsdot*», то рейтинг приходится переносить вручную. В связи с этим, было принято решение разработать свою систему с более развитым пользовательским интерфейсом и включить ее в систему «*Lmsdot*».

2. Требования к модулю

Опишем функциональные и требования для преподавателей и студентов.

В проектируемой системе преподаватель должен иметь возможность:

1. Создавать, просматривать, редактировать и удалять задачи и тестовые сценарии к ним;
2. Устанавливать сложность задачи;

Студент должен иметь возможность:

1. Просматривать условие задачи, примеры входных и выходных данных, а также ограничения;
2. Писать код с помощью редактора;
3. Просматривать результаты тестирования решения;

Для определения аппаратных и программных компонентов, которые необходимы для установки и работы модуля, были установлены следующие нефункциональные требования:

1. Операционная система: Windows, Linux и macOS;

2. Веб-браузер: Google Chrome, Яндекс Браузер, Mozilla Firefox, Microsoft Edge и Safari;

3. Поддержка языков программирования, таких как JavaScript [7] (компиляторы Node.js [8] версия 18.11.0) и Python [9] (версия 3.9), с возможностью запуска кода на сервере;

4. Масштабируемости серверной инфраструктуры.

Интерфейс приложения должен быть удобным, понятным и простым в восприятии для пользователей. Все кнопки должны иметь названия, соответствующие содержимому при их нажатии. Должен быть достигнут функциональный минимализм, при это информация должна быть актуальная и исчерпывающая. Пользователь должно легко взаимодействовать с пунктами меню, иконками и правилами приложения. Интерфейс приложения должен быть выполнен в единой стилистике и соответствовать современные тенденциям.

3. Архитектура системы

Для описания архитектуры решения использовался подход *C4* [10], который объединяет четыре иерархических уровня: *context* (с англ. – контекст), *container* (с англ. – контейнер), *Component* (с англ. – компонент) и *Code* (с англ. – код). Контекст показывает взаимодействия системы с внешними системами и пользователями, а контейнер – высокоуровневые блоки, которые являются отдельно запускаемые и развертываемые объекты. Компонент показывает устройство контейнера в виде функций, объединенных общим интерфейсом. Код является дополнительной формой детализации, отражающей программное решение для отдельно взятого компонента. Этот подход является наиболее подходящим для описания архитектуры. Он обеспечивает структурированный и понятный способ описания архитектуры системы на разных уровнях абстракции. Позволяет членам команды эффективно общаться и понимать архитектурные решения, так как использует наглядные и простые диаграммы. Благодаря модульному подходу и выделению компонентов, облегчает поддержку и изменение системы, позволяя более гибко вносить изменения в отдельные компоненты без нарушения общей архитектуры.

На рис. 1 изображена диаграмма контейнеров для разрабатываемого модуля

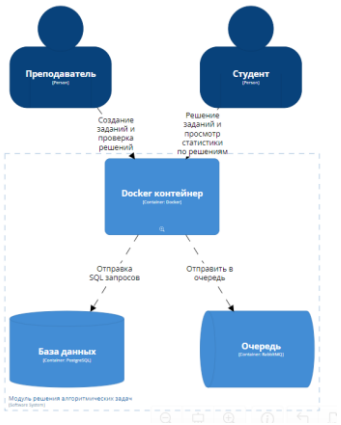

Рис. 1. Модель C4. Представление контейнеров

Она иллюстрирует взаимодействие «*Docker* контейнера» с очередью через отправку сообщений и с базой данных через отправку *SQL*-запросов. А также взаимодействие преподавателя через создание и проверку заданий и студента через решение и просмотр заданий.

На рис. 2 изображена диаграмма компонентов. Она иллюстрирует взаимодействие пользователей с остальными сервисами через пользовательский интерфейс. Пользовательский интерфейс взаимодействует с сервисом управления задачами и решениями через запросы на создание и проверку задач, а также просмотр и создание решений. Сервис управления задачами и решениями отправляет *SQL*-запросы к базе данных, а также отправляет и получает сообщения из очереди. С очередью взаимодействует сервис тестирования, который отправляет и получает из нее решения пользователей.

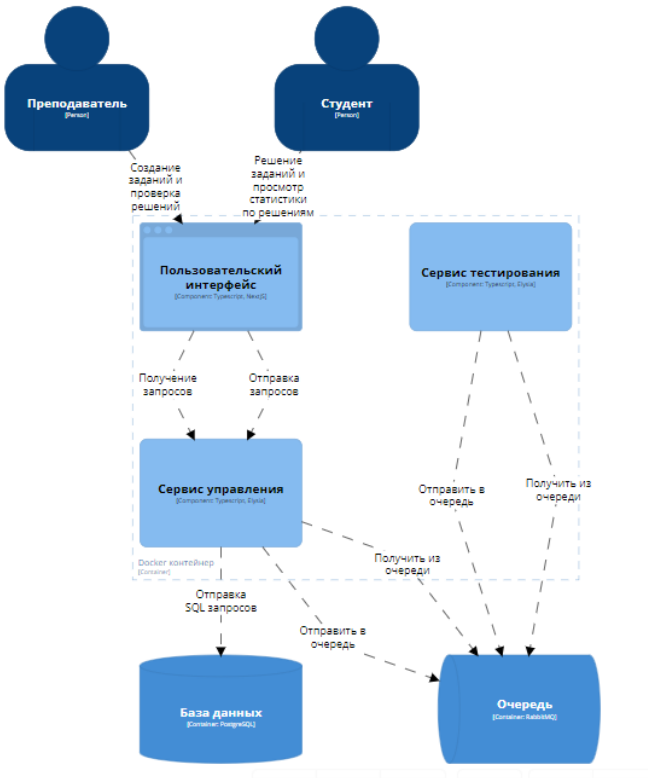

Рис. 2. Модель C4. Представление компонентов

4. Функциональная модель системы

Для моделирования взаимодействия объектов в модуле и описания последовательности обмена сообщениями между ними была создана диаграмма последовательности *UML* (от англ. *Unified*

Modeling Language – унифицированный язык моделирования). На рис. 3 изображена диаграмма последовательности данного модуля. Она показывает процесс решения задачи студентом. Студент отправляет запрос на получение доступных задач в *UI* (от англ. *User Interface* – пользовательский интерфейс), который рисует список из данных, полученных из сервиса управления *API* (от англ. *Application Programming Interface* – программный интерфейс приложения). Далее после перехода на страницу с задачей, пользовательский интерфейс рисует пользователю страницу с данными о задаче, полученными из «Сервиса управления». После написания студентом кода в редакторе, пользовательский интерфейс отправляет запрос к «Сервису управления». «Сервис управления» отправляет данные в очередь, из которой их получает «Сервис тестирования». Далее из данных генерируются тестирующая программа, которая после исполнения в «Контейнере для тестирования» возвращает результат работы обратно в «Сервис тестирования». Данные попадают обратно в очередь, из которой их достает «Сервис управления». После этого они возвращаются и отображаются в пользовательский интерфейс для пользователя.

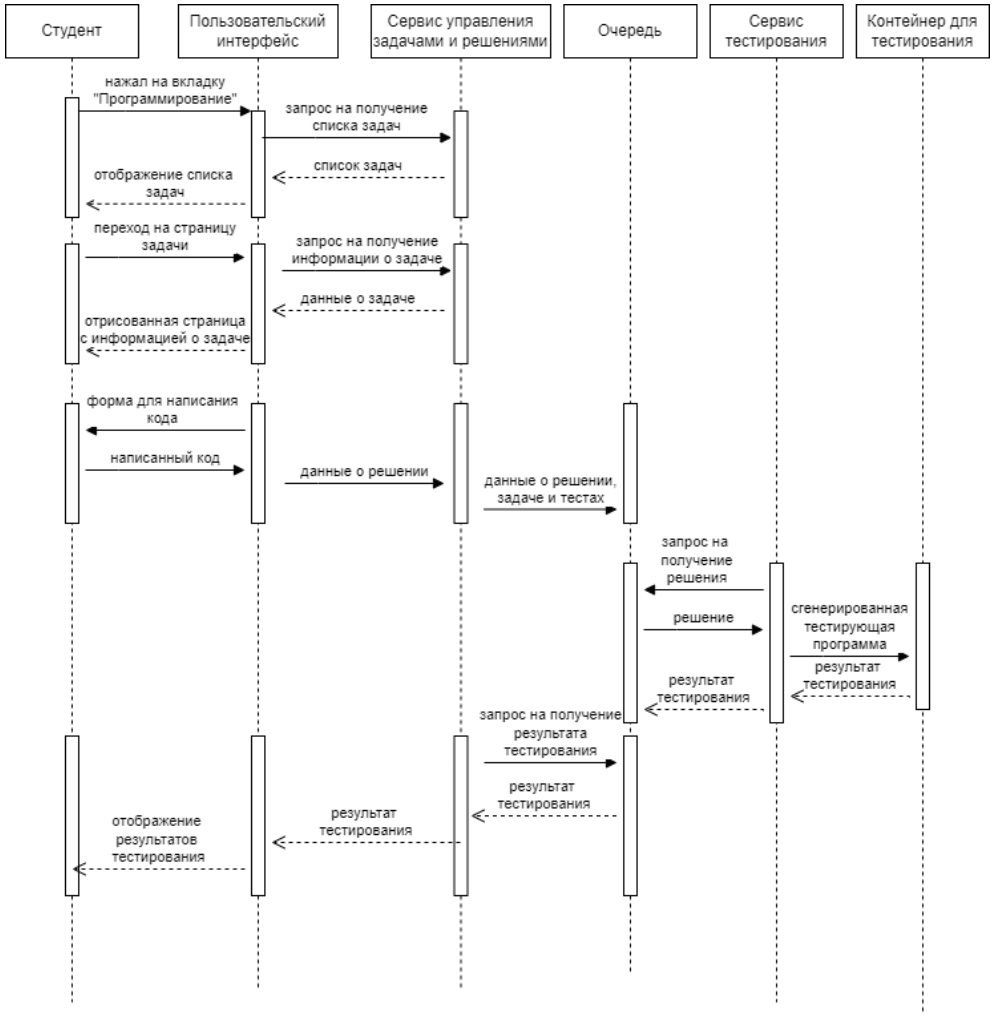

Рис. 3. Диаграмма последовательности. Решение задачи

Для моделирования структуры и планирования дизайна модуля, а также для документирования, были созданы диаграммы классов. На рис. 4 изображена диаграмма классов «Сервиса тестирования». Для генерации тестов был создан интерфейс *ITestStrategy*, который имплементируют конкретные реализации: *JSTestStrategy* (для генерации тестов программ, написанных на *JavaScript*), *PythonTestStrategy* (для генерации тестов программ, написанных на *Python*). Для инкапсуляции выбора реализации генерации тестов необходим класс *TestStrategyContext*. *TaskService* представляет собой класс, управляющий и запускающий тестирующие программы.

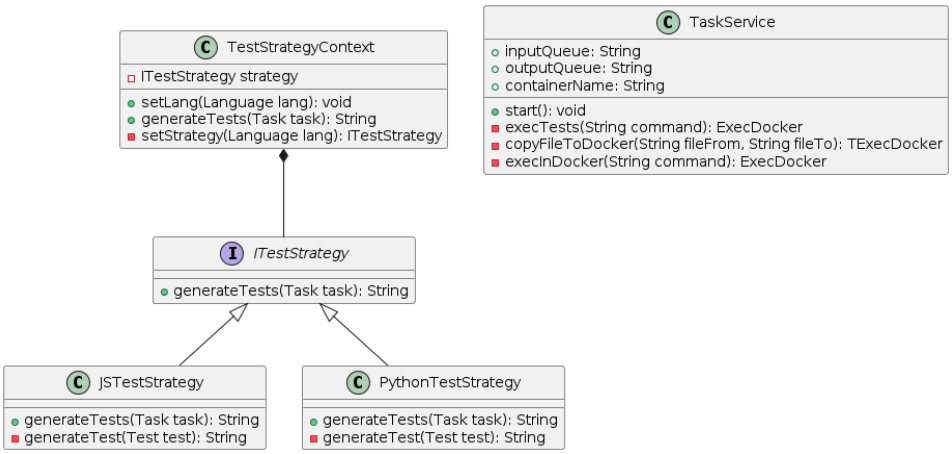

Рис. 4. Диаграмма классов. Сервис тестирования

На рис. 5 изображена диаграмма классов сервиса управления задачами и решениями. Класс *AmqpUtil* – это утилита для взаимодействия с очередью сообщений. *PrismaClient* – это класс, управляющий подключением к объектно-реляционной модели *Prisma*. Класс *SolutionService* является сервисом, который управляет решениями студентов, он агрегирует классы *AmqpUtil* для отправки решений в очередь сообщений и *PrismaClient* для отправки запросов к базе данных. Класс *TaskService* является сервисом, который управляет задачами, он агрегирует *PrismaClient*. Контроллеры *SolutionController* и *TaskController* управляют решениями и задачами соответственно. *App* – главный класс, запускающий приложение.

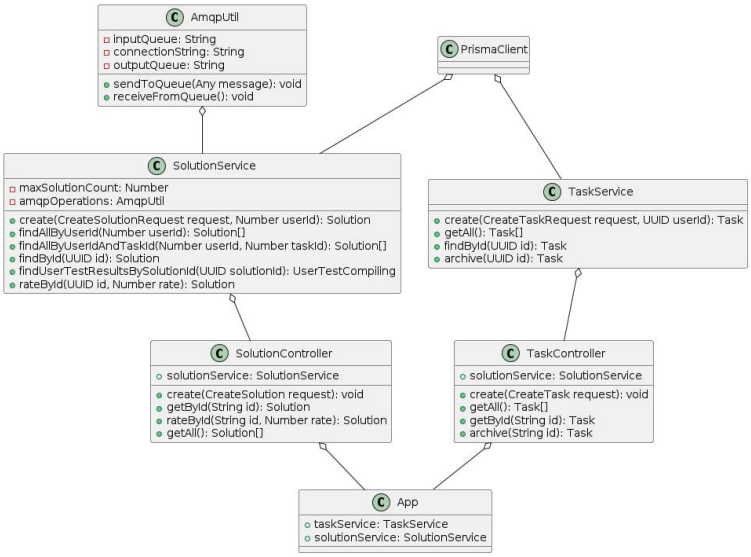

Рис. 4. Диаграмма классов. Сервис управления задачами и решениями

Для модуля была спроектирована схема базы данных (БД), представленная на рис. 6. В базе данных представлены таблицы: «Пользователь» – данные пользователя, «Сложность задачи» – уровень (критичная, высокая, средняя, низкая), «Тема» – тематика задачи, «Язык программирования» – доступные для всех задач языки программирования, «Языки программирования доступные в задаче», «Тег» – метки, доступные для всех задач, «Теги задачи и «Задача» – характеристики.

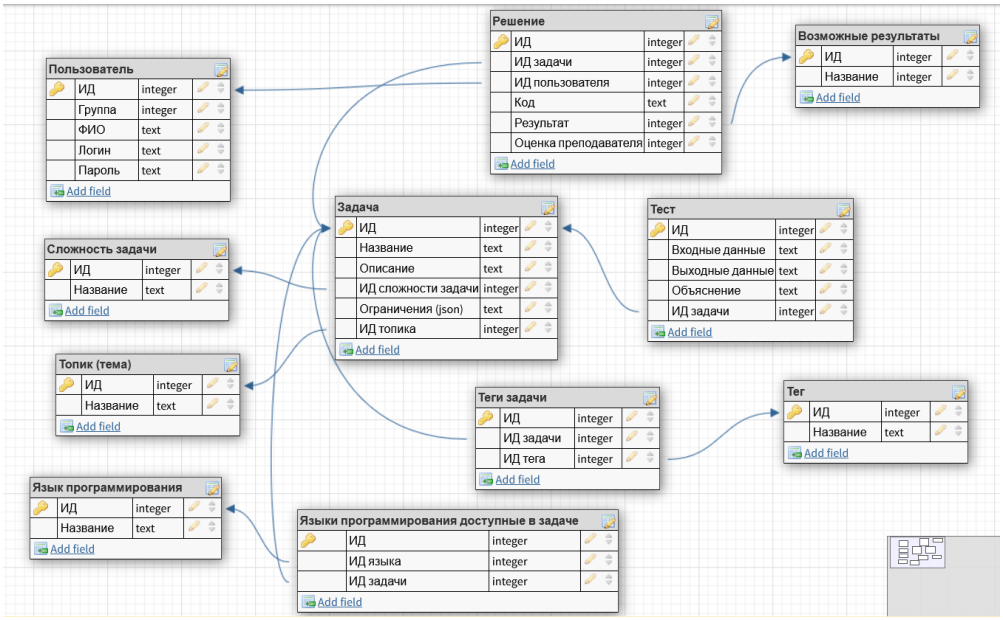

Рис. 5. Схема базы данных

5. Проектирование интерфейса

В работе были спроектированы экран студента, отображающий список задач и страница задачи на экране студента (см. рис. 7). Сверху на панели изображен заголовок задачи «Сумма чисел». Слева на экране показаны описание задачи и возможные примеры («#1 Сценарий»). Справа – редактор кода (код программы) с выбором языка программирования (JavaScript). Ниже, расположена панель с результатом выполнения и кнопкой «Запустить проверку» для тестирования программы. Над редактором изображена кнопка «Отправить» для отправки решения на проверку преподавателю.

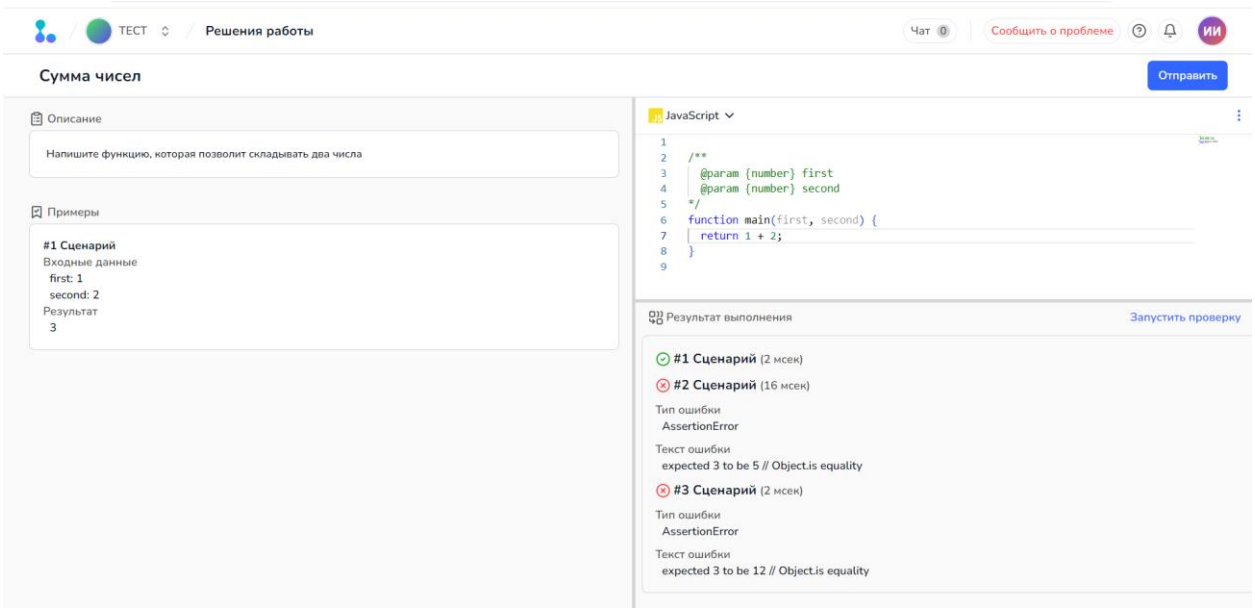

Рис. 7. Интерфейс задачи на экране студента

На рис. 8 изображен экран преподавателя для создания задачи.

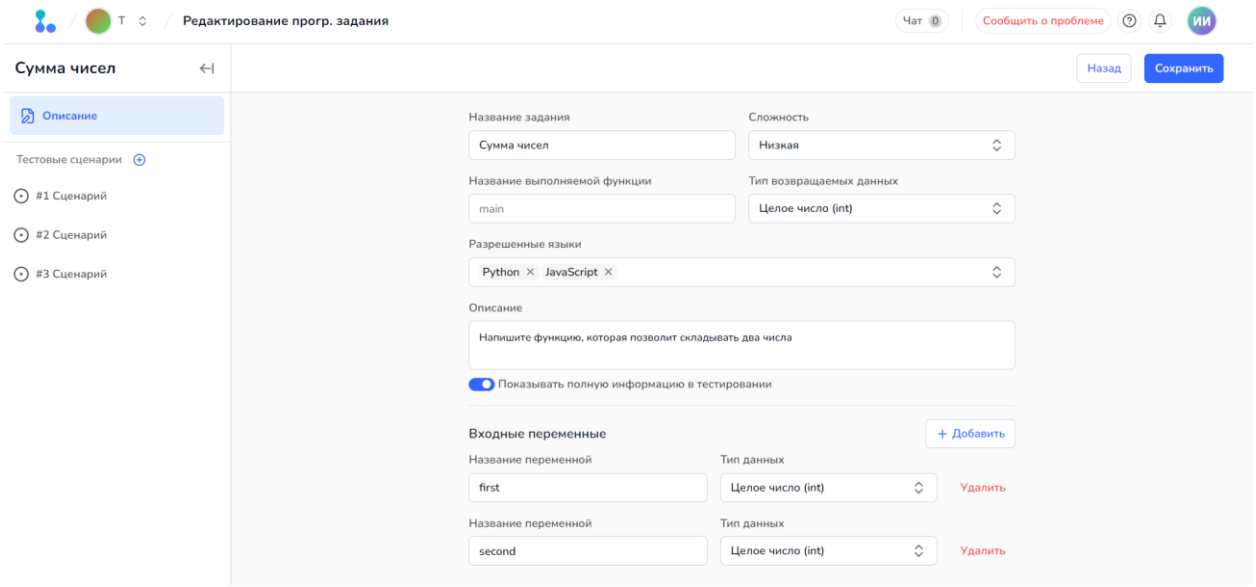

Рис. 8. Создание задачи на экране преподавателя

Слева на панели два пункта меню «Описание» (текущая вкладка) и «Тестовые сценарии». Справа представлена форма для создания задачи, в которой содержатся следующие поля: название задачи, название выполняемой функции, описание – текстовое поле; сложность, тип возвращаемого значения, разрешенные языки программирования выбираются из списка; скрывать данные сценариев (тестов) – переключатель; входные параметры – список с возможностью добавления; редактирования и удаления элементов. Элемент списка: название переменной – текстовое поле; тип данных — выбор из списка.

На рис. 9 изображен экран преподавателя для заполнения тестовых сценариев. Сверху присутствует выпадающий список с выбором сценария, который можно использовать как пример. Для этого справа изображен переключатель «Использовать как пример». На экране с конкретным сценарием пользователь может заполнить следующие поля: входные переменные, результат — список полей в зависимости от типа, выбранном на предыдущем экране; объяснение — текстовая область.

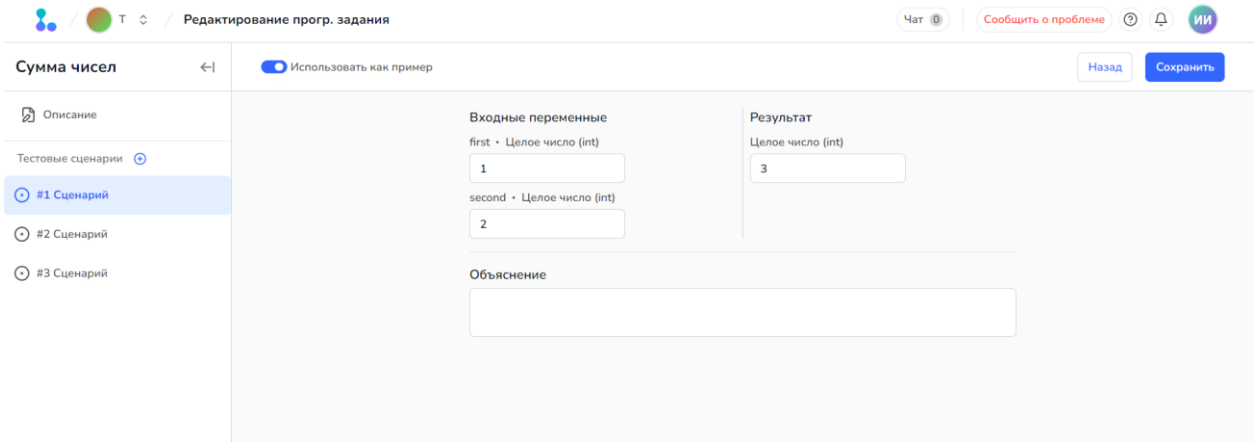

Рис. 9. Заполнение тестовых сценариев на экране преподавателя

6. Технологии реализации

Для реализации описанного решения были использованы следующие программные средства:

1. *TypeScript* — высокоуровневый язык программирования, расширяющий *JavaScript* [7], добавляя строгую типизацию [10].
2. *Node.js* — среда выполнения *JavaScript* от компании *Google* [8].

3. *Bun.js* — среда выполнения *JavaScript*, созданный с помощью языка программирования *Zig* [11].
4. *HTML* (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов для веб-страниц в браузере [12].
5. *CSS* (от англ. *Cascading Style Sheets* — «каскадные таблицы стилей») — язык таблиц стилей, используемый для описания представления документа, написанного на HTML или XML [13].
6. *PostgreSQL* — объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом [14].
7. *Prisma* — объектно-реляционное отображение (*ORM*) с открытым исходным кодом [15].
8. *Next.js* — JavaScript фреймворк с открытым исходным кодом, построенный поверх библиотеки *React.js* [16].
9. *Elysia.js* — современный *TypeScript* фреймворк со сквозной (*E2E*) безопасностью типов, который создан для работы на среде *Bun.js* в 2022 году [17].
10. *RabbitMQ* — брокер обмена сообщениями и потоковой передачи [18].
11. *Docker* — платформа с открытым исходным кодом для разработки, доставки и запуска приложений [19].
12. *Git* — распределенная система контроля версий с открытым исходным кодом, предназначенная для обработки проектов [20].
13. *Intellij IDEA* — интегрированная среда разработки (*IDE*) для создания программных продуктов на основе большого количества языков программирования [21].

7. Внедрение и тестирование

Внедрение модуля алгоритмических задач в систему «*Lmsdot*» происходило по следующим этапам. Были добавлены таблицы в существующую базу данных, а также загружены необходимые справочные данные. С помощью системы версионирования *Git*, код модуля был добавлен в репозиторий системы на *Github* [22], а также была воспроизведена интеграция и адаптация с уже имеющимися сервисами. Подготовка клиентских машин заключалась в добавлении *Docker*-файлов для установки модуля и контейнера запуска тестирования кода. После инсталляции модуля, был произведен процесс тестирования.

Заключение

В работе проведен анализ популярных платформ, предназначенных для решений алгоритмических задач. Детально сформулированы функциональные и нефункциональные требования к разрабатываемому модулю. Изучена структура приложения «*Lmsdot*». Выбраны технологии реализации.

Для моделирования взаимодействия объектов в разрабатываемом модуле и описания последовательности обмена сообщениями между ними была создана диаграмма последовательности. Разработана диаграмма классов для представления внутренней структуры программы, планирования дизайна модуля и составления документации. Спроектирована схема базы данных и приведено ее описание.

В работе реализованы «Сервис тестирования» программ студентов и «Сервис управления» задачами и решениями. Разработаны таблицы базы данных для модуля.

С помощью системы версионирования *Git*, код модуля был добавлен в репозиторий системы на *Github*. После инсталляции модуля, был произведен процесс тестирования.

Список источников

1. Кочешков А. Д., Смирнов Д. П., Дедович Т. Г. Разработка информационной системы контроля и оценки знаний студентов по математическим дисциплинам в университете «Дубна»// Системный анализ в науке и образовании. 2021. № 2. С. 140–150. URL: http://sanse.ru/download/442.
2. Информатикс : [система дистанционного обучения]. – URL: https://informatics.msk.ru/ (дата обращения: 05.03.2024).
3. LeetCode - The World's Leading Online Programming Learning Platform. –LeetCode, 2024. – URL: https://leetcode.com/ (дата обращения: 05.03.2024).
4. Codeforces : [платформа для соревнований по программированию]. – Михаил Мирзаянов, 2010-2024. – URL: https://codeforces.com/ (дата обращения: 05.03.2024).
5. Яндекс.Контест : [Онлайн-платформа для решения задач по программированию]. – Яндекс, 2013–2024. – URL: https://contest.yandex.ru/ (дата обращения: 05.03.2024).
6. Трафик сайта: проверяйте и анализируйте : [веб-сайт]. – Similarweb LTD, 2024. – URL: https://www.similarweb.com/ru/ (дата обращения: 05.03.2024).
7. JavaScript | MDN. – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 04.04.2024).
8. Node.js – Run JavaScript Everywhere / OpenJS Foundation. – URL: https://nodejs.org/en (дата обращения: 06.04.2024).
9. Welcome to Python.org. – Python Software Foundation, 2001-2024. – URL: https://www.python.org/ (дата обращения: 04.04.2024).
10. TypeScript: JavaScript With Syntax For Types – Microsoft, 2012-2024. – URL: https://www.typescriptlang.org/ (дата обращения: 06.04.2024).
11. Bun – A fast all-in-one JavaScript runtime. – URL: https://bun.sh (дата обращения: 06.04.2024).
12. HTML Standard / WHATWG (Apple, Google, Mozilla, Microsoft). – URL: https://html.spec.whatwg.org/ (дата обращения: 06.04.2024).
13. CSS: Cascading Style Sheets - MDN Web Docs. – URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата обращения: 06.04.2024).
14. PostgreSQL: The world's most advanced open source database. –The PostgreSQL Global Development Group, 1996-2024. – URL: https://www.postgresql.org (дата обращения: 06.04.2024).
15. Prisma | Simplify working and interacting with databases – Prisma Data, Inc., 2024. – URL: https://www.prisma.io (дата обращения: 09.04.2024).
16. Next.js by Vercel - The React Framework. –Vercel, Inc., 2024. – URL: https://nextjs.org (дата обращения: 09.04.2024).
17. ElysiaJS: Elysia - Ergonomic Framework for Humans. – URL: https://elysiajs.com (дата обращения: 09.04.2024).
18. RabbitMQ: One broker to queue them all | RabbitMQ. – Broadcom, 2005-2024. – URL: https://www.rabbitmq.com (дата обращения: 09.04.2024).
19. Docker: Accelerated Container Application Development – Docker Inc., 2024. – URL: https://www.docker.com (дата обращения: 09.04.2024).
20. Git : [website] / Software Freedom Conservancy. – URL: https://git-scm.com/ (дата обращения: 09.04.2024).
21. IntelliJ IDEA – ведущая IDE для разработки на Java и Kotlin. – JetBrains s.r.o., 2000-2024. – URL: https://www.jetbrains.com/ru-ru/idea (дата обращения: 09.04.2024).
22. GitHub: Let's build from here // GitHub : [web platform]. – GitHub, Inc., 2025. – URL: https://github.com/ (дата обращения: 09.04.2024).