**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc200886639)

[1 Анализ предметной области 8](#_Toc200886640)

[1.1 Анализ пользовательских сценариев взаимодействия 8](#_Toc200886641)

[1.1.1 Сценарий взаимодействия со стороны ученика 8](#_Toc200886642)

[1.1.2 Сценарий взаимодействия со стороны преподавателя 9](#_Toc200886643)

[1.1.2 Сценарий автоматизированной проверки решения 11](#_Toc200886644)

[1.2 Описание предметной области и сущностей системы 13](#_Toc200886645)

[1.2.1 Сущность Пользователь 13](#_Toc200886646)

[1.2.2 Сущность Задача 14](#_Toc200886647)

[1.2.3 Сущность Тег 15](#_Toc200886648)

[1.2.4 Сущность Язык 15](#_Toc200886649)

[1.2.5 Сущность Решение 16](#_Toc200886650)

[1.2.6 Сущность Результат проверки решения 16](#_Toc200886651)

[1.2.7 Сущность Комната 17](#_Toc200886652)

[2 Проектирование системы 18](#_Toc200886653)

[2.1 Проектирование базы данных 18](#_Toc200886654)

[2.2 Проектирование функциональных слоев 20](#_Toc200886655)

[3 Проектирование презентационного слоя 22](#_Toc200886656)

[3.1 Карта сайта 22](#_Toc200886657)

[3.2 Макеты страниц сайта 22](#_Toc200886658)

[ЗАКЛЮЧЕНИЕ 28](#_Toc200886659)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc200886660)

ВВЕДЕНИЕ

В современном мире информационные технологии играют ведущую роль в развитии различных отраслей экономики и науки. Следовательно, спрос на IT-специалистов, включая разработчиков программного обеспечения, постоянно растет. В связи с этим, возникает необходимость в подготовке новых специалистов в области программирования. Образовательные информационные системы с алгоритмическими задачами по программированию становятся неотъемлемой частью этого процесса подготовки. Эти системы предоставляют студентам и начинающим программистам возможность развивать свои навыки, решая разнообразные задачи с различным уровнем сложности. Благодаря таким системам студенты могут улучшить свои навыки алгоритмизации, научиться эффективно решать задачи и практиковаться в программировании на различных языках. Более того, образовательные информационные системы обеспечивают доступ к обучающим материалам, а также предоставляют возможность соревноваться с другими пользователями, что мотивирует студентов к развитию и позволяет им оценить свой прогресс в сравнении с другими. Такие системы также часто предлагают системы автоматической проверки решений, что позволяет студентам получать обратную связь и исправлять ошибки, улучшая свои навыки программирования.

# **1 Анализ предметной области**

Для формирования требований к информационной системе обучения программированию был проведён анализ существующих решений и рассмотрен типовой сценарий использования платформы со стороны обучающегося.

## **1.1 Анализ пользовательских сценариев взаимодействия**

Начнем анализ предметной области с анализа пользовательских сценариев. Это позволяет выделить основные сущности предметной области, их ограничения и функциональность.

## **1.1.1 Сценарий взаимодействия со стороны ученика**

Пользователь, обладающий базовыми навыками программирования, стремится улучшить свои компетенции в данной области. Его мотивация может быть связана с подготовкой к собеседованию, изучением нового языка программирования или закреплением ранее изученных концепций. В поисках подходящей платформы пользователь находит онлайн-ресурс (например, Codewars [1]) и приступает к работе. На платформе пользователь проходит следующие этапы.

Регистрация и вход в систему. Пользователь авторизуется с использованием электронной почты и пароля, получая доступ к личному кабинету.

Выбор задачи для решения. На странице платформы представлен список задач, доступных для решения. Каждая задача сопровождается описанием, тегами, а также указанием поддерживаемых языков программирования. Пользователь выбирает интересующую задачу и переходит к её решению.

Решение задачи и проверка ответа. Пользователь пишет программный код в интегрированном редакторе и отправляет его на автоматическую проверку. Результат проверки формируется и возвращается в течение примерно 1–2 секунд.

Пример полученного отчёта:

* время выполнения: 757 мс;
* тесты пройдены: 0;
* тесты не пройдены: 41;
* выходной код: 41.

После внесения правок и повторной отправки решения результаты проверки обновляются:

* время выполнения: 852 мс;
* тесты пройдены: 41;
* тесты не пройдены: 0.

Таким образом, пользователь получает обратную связь по качеству решения и может переходить к следующей задаче.

Совместная работа над задачей. Платформа поддерживает возможность совместного решения задач несколькими пользователями в онлайн-режиме. Пользователь может пригласить другого участника, отправив ссылку на совместную сессию. Подключившийся участник получает доступ к редактору и может вносить изменения в код. Проверку решения осуществляет только пользователь, создавший сессию (владелец комнаты).

## **1.1.2 Сценарий взаимодействия со стороны преподавателя**

Рассмотрим функциональные возможности платформы с точки зрения преподавателя — пользователя, обладающего достаточным уровнем знаний для формирования заданий по программированию.

Преподаватель, ранее сам прошедший путь обучения, имеет возможность создавать новые задачи для практики обучающихся. Для этого он переходит на страницу создания задачи, где выполняет следующие действия.

Формирование задачи. В процессе создания задания преподаватель заполняет основные поля:

* название задачи — краткий и ёмкий заголовок, отражающий суть задания;
* описание задачи — подробное текстовое описание, поддерживающее форматирование для оформления (включая заголовки, списки, выделение кода и др.);
* список тегов — ключевые слова, позволяющие классифицировать задачу по темам и уровням сложности.

Настройка параметров задачи по языкам программирования. Преподаватель указывает поддерживаемые языки программирования для задачи. Для каждого выбранного языка настраиваются следующие файлы:

* файл с кодом полного решения — эталонное решение, необходимое для проверки корректности тестов;
* файл с кодом шаблона для решения — минимальная заготовка кода, предоставляемая обучающемуся для начала работы;
* файл с бойлерплейт-кодом (необязательный) — дополнительный файл, содержащий вспомогательные структуры, которые могут быть использованы при решении задачи;
* файл с кодом тестов — скрипт, автоматически выполняющий проверку корректности решения обучающегося.

Управление задачами. После создания задачи преподаватель имеет следующие возможности:

* сохранить задачу как черновик;
* опубликовать задачу для общего доступа обучающимся;
* обновить существующую задачу и повторно опубликовать её в виде новой версии;
* снять задачу с публикации, сделав её недоступной для решения;
* удалить задачу полностью из системы.

Каждое редактирование задачи сопровождается созданием новой ревизии, для которой фиксируются следующие данные:

* номер ревизии;
* дата создания;
* автор изменений.

При повторной публикации задачи используется последняя ревизия, что обеспечивает контроль версионности и воспроизводимость материалов.

## **1.1.2 Сценарий автоматизированной проверки решения**

Ключевым компонентом системы является модуль автоматизированной проверки решений, обеспечивающий обратную связь пользователям в режиме реального времени. Процесс проверки решения задачи включает следующие этапы.

Отправка задачи на проверку. При отправке решения на проверку система получает от клиента следующие данные:

* идентификатор пользователя (userId);
* идентификатор задачи (problemId);
* идентификатор выбранного языка программирования (languageId);
* исходный код решения (code).

Подготовка к выполнению проверки. Система выполняет следующие действия:

* находит задачу в базе данных по problemId;
* извлекает файлы с кодом задачи, включая тесты и шаблоны, соответствующие выбранному языку программирования (languageId);
* получает конфигурацию запуска контейнера (ПОДа) для выполнения решения на указанном языке.

Запуск проверки в изолированной среде. Для обеспечения безопасности и изоляции выполнения кода создаётся контейнер (ПОД), настроенный для запуска выбранного языка программирования. В контейнер передаются:

* код задачи (тесты и вспомогательные файлы);
* код решения, написанный пользователем.

Выполнение проверки и анализ результатов. Внутри ПОДа запускается тестовый фреймворк, который выполняет проверку решения пользователя по заранее подготовленным тестам. По завершении выполнения система получает результаты выполнения тестов, и анализирует их на предмет успешного прохождения всех тестов.

Формирование и возврат результата. Система возвращает пользователю информацию о результате проверки, включая количество успешно пройденных и не пройденных тестов, а также возможные сообщения об ошибках.

Дополнительной возможностью платформы является предоставление пользователям среды для запуска кода вне контекста конкретной задачи (Code Playground). Данная функциональность аналогична инструменту, доступному на платформе LeetCode, и предоставляет следующие возможности.

Выбор языка и написание кода. Пользователь выбирает язык программирования из поддерживаемых системой и вводит исходный код в редакторе.

Запуск кода. Система находит конфигурацию запуска ПОДа для выбранного языка; создаёт изолированный контейнер, запускает его и передаёт в него введённый пользователем код.

Получение результатов выполнения. После выполнения кода в ПОДе система возвращает пользователю консольный вывод программы, что позволяет отлаживать и тестировать код в интерактивном режиме.

## **1.2 Описание предметной области и сущностей системы**

Система интерактивного обучения программированию представляет собой совокупность взаимосвязанных сущностей, каждая из которых выполняет строго определённую роль в процессе обучения, создания и проверки задач. Ниже приведены ключевые сущности, их описания и пояснения к полям.

## **1.2.1 Сущность Пользователь**

Пользователь — это зарегистрированный участник системы, который может выполнять действия в зависимости от своих ролей и прав доступа. Атрибуты пользователя:

* email — строка; адрес электронной почты пользователя, используется для входа в систему и связи;
* username — строка; отображаемое имя пользователя;
* password — строка; хеш пароля для аутентификации;
* image — строка; путь или URL к изображению профиля пользователя;
* roles — массив строк; роли пользователя в системе; возможные значения: "user" (обычный пользователь) или "admin" (администратор);
* permissions — массив строк; конкретные права пользователя (например, право на создание задач, модерацию и т.д.).

Основной функционал:

* регистрация и авторизация пользователя;
* удаление пользователя из системы (доступно администратору);
* смена пароля и изображения профиля.

## **1.2.2 Сущность Задача**

Задача — основная единица контента системы, предназначенная для обучения и проверки навыков программирования. Атрибуты Задачи:

* author — ссылка на сущность User; автор задачи;
* name — строка; название задачи;
* description — строка; описание задачи в формате Markdown;
* difficulty — строка; уровень сложности задачи (например, "легкая", "средняя", "сложная");
* tags — массив сущностей Tag; тематические теги задачи;
* revision — число; номер текущей ревизии задачи;
* isPublished — логический тип; признак публикации задачи (true — задача опубликована, false — находится в черновике);
* languages — массив объектов, описывающих реализацию задачи для разных языков программирования;
  + completeSolution — строка; файл с полным, правильным решением задачи;
  + initialSolution — строка; файл с шаблоном для начала решения задачи;
  + preloaded — строка; дополнительный (необязательный) файл с предзагруженным кодом;
  + tests — строка; файл с тестами для проверки решения.

Основной функционал:

* создание, редактирование, удаление и публикация задач (при наличии соответствующих прав доступа);
* просмотр списка задач;
* получение детальной информации о задаче.

## **1.2.3 Сущность Тег**

Тег — метаданные, помогающие классифицировать задачи по тематикам. Атрибуты Тега:

* name — строка; название тега (например, "массивы", "рекурсия", "строки").

Основной функционал:

* создание, редактирование и удаление тегов (для администратора);
* получение списка тегов (для всех пользователей).

## **1.2.4 Сущность Язык**

Язык программирования описывает поддерживаемый язык, на котором пользователь может писать решения задач. Атрибуты Языка:

* name — строка; название языка программирования (например, "Python", "JavaScript");
* image — строка; Docker-образ, содержащий рантайм языка и настроенный тестовый фреймворк для запуска кода;
* podManifest — строка; Kubernetes-манифест для создания изолированного контейнера (ПОДа) для запуска решений;
* testRunner — строка; Java-класс, который выполняет запуск тестов для проверки решений;
* taskRunner — строка; Java-класс, выполняющий произвольный код в режиме Code Playground.

Основной функционал:

* получение списка поддерживаемых языков;
* интеграция новых языков в систему через механизм плагинов.

## **1.2.5 Сущность Решение**

Решение — код, отправляемый пользователем для проверки задачи. Атрибуты Решения:

* user — ссылка на сущность User; пользователь, отправивший решение;
* problem — ссылка на сущность Problem; задача, для которой предоставлено решение;
* language — ссылка на сущность Language; язык программирования решения;
* code — строка; исходный код решения.

Основной функционал:

* отправка решения на автоматизированную проверку;
* просмотр списка отправленных решений и их результатов.

## **1.2.6 Сущность Результат проверки решения**

Результат проверки решения фиксирует исход выполнения проверки кода, отправленного пользователем. Атрибуты Результата проверки решения:

* solution — ссылка на сущность Solution; связанное решение;
* exitCode — число; код завершения процесса выполнения решения;
* stderr — строка; стандартный поток ошибок;
* stdout — строка; стандартный поток вывода;
* timedOut — логический тип; признак того, что выполнение превысило лимит времени;
* time — число; время выполнения решения (в миллисекундах);
* completed — логический тип; признак завершения проверки;
* tests — массив объектов, каждый из которых содержит:
* name — строка; название теста;
* passed — логический тип; результат прохождения теста (успешно/неуспешно);

Основной функционал:

* создание и хранение результатов проверки (выполняется системой автоматически).

## **1.2.7 Сущность Комната**

Комната — виртуальное пространство для совместной работы пользователей над решением задачи.

Атрибуты:

* problem — ссылка на сущность Problem; задача, которая редактируется в комнате;
* owner — ссылка на сущность User; владелец комнаты;
* link — строка; уникальная ссылка для подключения к комнате;
* members — массив сущностей User; участники комнаты;
* code — строка; текущий код решения в комнате.

Основной функционал:

* создание комнаты для совместной работы;
* подключение участников к комнате по уникальной ссылке;
* совместное редактирование кода.

# **2 Проектирование системы**

На основе проведённого анализа предметной области разработано проектное решение системы.

## **2.1 Проектирование базы данных**

Основные сущности и их взаимосвязи выявлены, описаны и формализованы в виде схемы базы данных. Для наглядности и удобства дальнейшей реализации создана ER-диаграмма, отражающая структуру данных системы. Она представлена на рисунке 2.1.

ER-диаграмма включает все основные сущности:

* пользователи (User);
* задачи (Problem);
* теги (Tag);
* языки программирования (Language);
* решения (Solution);
* результаты проверки решений (SolutionResult).

Схема базы данных, представленная в виде ER-диаграммы, позволяет визуализировать и структурировать данные проекта, а также служит основой для построения реляционной модели данных в выбранной системе управления базами данных.

Изображение выглядит как текст, диаграмма, План, Параллельный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.1 – ER диаграмма базы данных платформы

## **2.2 Проектирование функциональных слоев**

При разработке программной системы особое внимание было уделено архитектуре, обеспечивающей модульность, масштабируемость и упрощённую поддержку кода. Для достижения этих целей применён подход многоуровневой (слоистой) архитектуры, разделяющей систему на функциональные слои, каждый из которых решает строго ограниченный круг задач.

Функциональные слои представляют собой логические уровни архитектуры, отделяющие друг от друга различные аспекты системы: отображение, бизнес-логику, работу с данными и т. д. Каждый слой взаимодействует только с соседними слоями, что обеспечивает слабую связность компонентов и способствует принципу единой ответственности. На рисунке 2.1 представлена схема функциональных слоев системы.

Слой представления (Presentation Layer). Представлен в виде веб-клиента, разработанного с использованием React [2]. Этот слой отвечает за взаимодействие с пользователем: отображение данных, отправку форм, визуализацию результатов. Он обращается к доменному слою через API.

Доменный слой (Domain Layer). Содержит основные модели предметной области (User, Problem, Solution, и др.) и сервисы бизнес-логики (UserService, ProblemService, и др.). Вся логика обработки данных и правил валидации сосредоточена здесь. Этот слой не зависит от конкретной реализации хранения данных и может использоваться в различных контекстах.

Слой доступа к данным (Data Access Layer). Содержит репозитории (UserRepository, ProblemRepository, и др.), отвечающие за сохранение и извлечение данных. Этот слой взаимодействует с реальными хранилищами, инкапсулируя детали SQL-запросов, структуры таблиц и API внешних хранилищ.

Инфраструктурный слой. На нижнем уровне расположены СУБД PostgreSQL [3], используемая для хранения структурированных данных, и MinIO [4] — объектное хранилище, применяемое для хранения файлов (например, изображений пользователей или исходного кода решений). Доступ к этим хранилищам осуществляется строго через data-слой.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.1 – Функциональные слои приложения

# **3 Проектирование презентационного слоя**

Глава 3 содержит информацию о проектировании презентационного слоя. Приведена карта сайта и макеты страниц будущего сайта.

## **3.1 Карта сайта**

Карта сайта позволяет оценить глобальную структуру сайта. Она изображена на рисунке 3.1. Розовым цветом помечены страницы, которые доступны неавторизованным пользователям (общедоступные), а помеченные голубым страницы требую авторизацию.

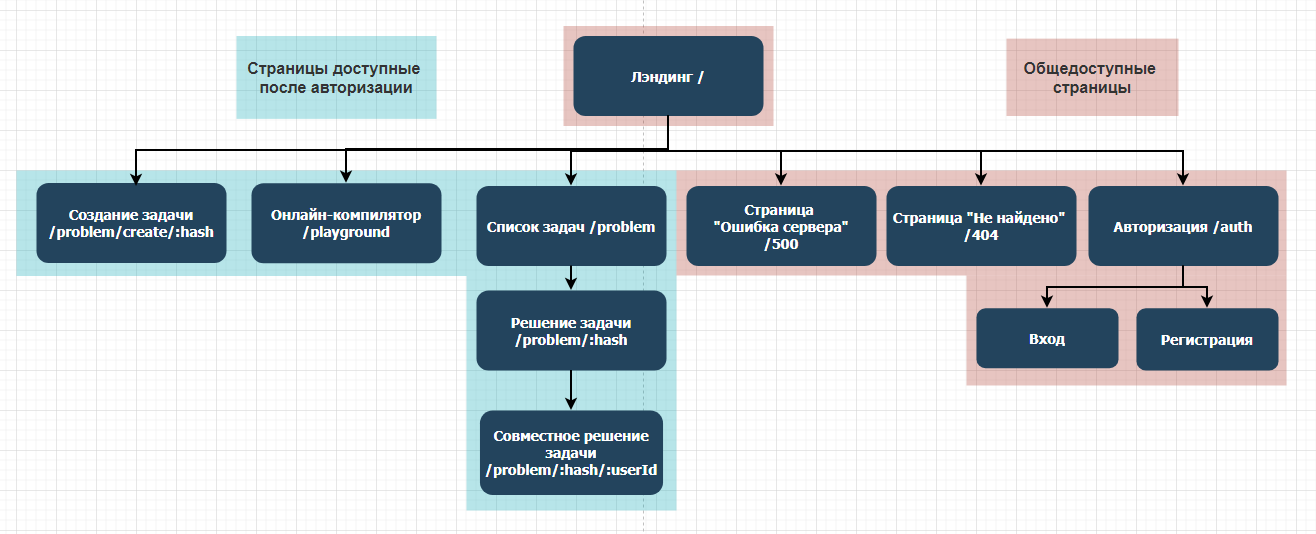


Рисунок 3.1 – Карта сайта

## **3.2 Макеты страниц сайта**

На рисунках 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 представлены спроектированные макеты страниц сайта.

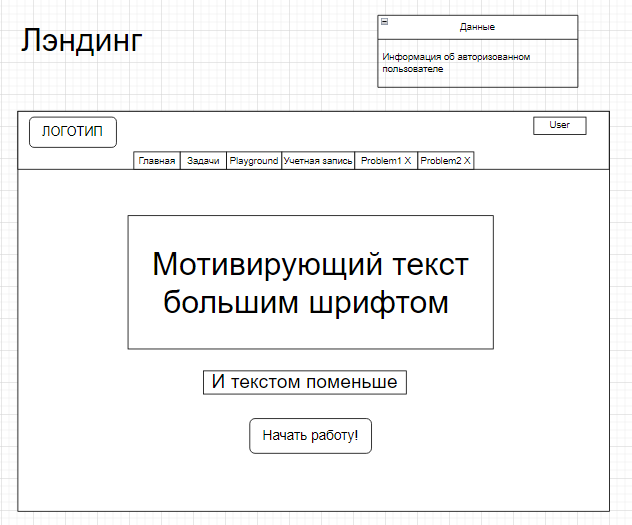


Рисунок 3.2 – Макет главной страницы сайта

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.3 – Макет страницы авторизации

Изображение выглядит как текст, снимок экрана, Параллельный, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.4 – Макет страницы авторизации

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.5 – Макет страницы авторизации

Изображение выглядит как текст, снимок экрана, диаграмма, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.6 – Макет страницы авторизации

Изображение выглядит как текст, диаграмма, снимок экрана, План

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.7 – Макет страницы авторизации

ЗАКЛЮЧЕНИЕ

В рамках данной работы проведён комплексный анализ предметной области, связанной с автоматизированным обучением программированию и проверкой решений. Были определены сценарии работы с системой, сущности предметной области, ограничения и их функциональность.

Разработана функциональная модель системы, включающая описание предметной области, основных сущностей и их взаимосвязей. Построена ER-диаграмма, отражающая структуру базы данных, а также представлена архитектура системы по функциональным слоям, обеспечивающая модульность и расширяемость. Разделение на уровни презентации, доменной логики и доступа к данным позволило выделить ответственность каждого слоя и сформировать основу для масштабируемой архитектуры.

Дополнительно были подготовлены макеты пользовательских интерфейсов и составлена карта сайта, что позволяет обеспечить удобство пользовательского взаимодействия и целостность пользовательского опыта.

Полученные результаты служат основой для реализации онлайн-платформы «SupremeCode», предназначенной для обучения программированию с возможностью отправки решений и их автоматической проверки в изолированной среде. Платформа ориентирована как на учеников, так и на преподавателей, обеспечивая гибкость, безопасность и расширяемость функционала.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Codewars Обучение программированию [Электронный ресурс],  
   URL: https://www.codewars.com/dashboard (дата обращения: 01.05.2025).
2. React JavaScript библиотека для создания пользовательских интерфейсов [Электронный ресурс], URL: https://react.dev/ (дата обращения: 01.05.2025).
3. PostgreSQL Объектно-реляционная система управления базами данных [Электронный ресурс], URL: https://www.postgresql.org/ (дата обращения: 01.05.2025).
4. MinIO S3-совместимое хранилище файлов [Электронный ресурс], URL: https://min.io/ (дата обращения: 01.05.2025).