Оглавление

[1. Постановка и анализ задачи 2](#_Toc153819198)

[1.1 Анализ предметной области 2](#_Toc153819199)

[1.2 Обоснование актуальности работы 5](#_Toc153819200)

[1.2.1 Существующие решения 7](#_Toc153819201)

[1.2.2 Сравнительная характеристика 11](#_Toc153819202)

[1.3 Постановка задачи 12](#_Toc153819203)

[1.4 Выбор языков программирования и технологий 14](#_Toc153819204)

[2. Проектирование платформы 17](#_Toc153819205)

[2.1 Проектирование процесса проверки решения на серверной части 17](#_Toc153819206)

[2.2 Инфологическое проектирование Базы Данных. 19](#_Toc153819207)

[2.3 Даталогическое проектирование Базы Данных. 23](#_Toc153819208)

# 1. Постановка и анализ задачи

## 1.1 Анализ предметной области

В наше время информационных технологий отмечается внушительный рост интереса к сфере IT со стороны трудовых ресурсов. Большое количество желающих начать карьеру в этой сфере создает конкурентное окружение, в котором отличие и выделение требуют особых усилий и навыков от начинающих специалистов. Это обусловлено не только привлекательностью высоких заработных плат, но и осознанием растущего спроса на квалифицированных IT-профессионалов.

С ростом числа работников в IT-сфере увеличиваются и требования, предъявляемые к начинающим специалистам. Компании ожидают не только теоретических знаний, но и умения применять их на практике. Это обусловлено динамикой индустрии, постоянными технологическими изменениями и потребностью в сотрудниках, способных быстро адаптироваться к новым вызовам.

Одним из важных аспектов в арсенале начинающего IT-специалиста является умение эффективно решать алгоритмические задачи. В современной практике собеседований в технологических компаниях, классические алгоритмические задачи стали непременным этапом отбора. Подобные задания не только оценивают знание базовых алгоритмов и структур данных, но и позволяют судить о широте мышления и готовности к решению реальных проблем.

Решение алгоритмических задач представляет собой не просто формальное испытание, а возможность показать свою способность анализа, творческого мышления и глубины понимания основных принципов программирования. В этом контексте, алгоритмическое программирование становится неотъемлемой частью профессионального роста и успешного вступления в индустрию.

Сложность и разнообразие алгоритмических задач позволяют оценить не только техническую подготовку, но и гибкость мышления, стремление к самосовершенствованию и умение быстро принимать решения в условиях ограниченного времени. Эти качества являются неотъемлемой частью успешной карьеры в области информационных технологий.

Постоянная тренировка и развитие в области алгоритмического программирования становятся непременной частью жизни IT-специалиста. Систематическое решение алгоритмических задач не только поддерживает высокий уровень технических компетенций, но и обеспечивает гибкость в принятии решений в условиях быстро меняющейся индустрии.

Для упрощения и более объективного отбора специалистов в сфере информационных технологий, многие компании предпочитают использовать различные форматы конкурсов и контестов, в рамках которых участники решают сложные алгоритмические задачи. Этот метод позволяет выделить наиболее перспективных кандидатов, предоставляя компаниям возможность более тщательно оценить не только технические навыки, но и креативность, способность к коллаборации и быстрое принятие решений.

Контесты по алгоритмическому программированию становятся своеобразным полигоном, где участники могут продемонстрировать себя и выделиться на фоне конкуренции. Сложность задач, предлагаемых в таких мероприятиях, поднимается настолько высоко, что только настоящие мастера своего дела могут успешно справиться.

Проведение контестов также подчеркивает важность не только знания алгоритмов, но и умение применять их в условиях ограниченного времени и неопределенности. Соревновательный характер задач позволяет выявить не только теоретическую готовность, но и способность к принятию быстрых и обоснованных решений в динамичной среде.

Такие мероприятия не только служат эффективным инструментом отбора талантливых специалистов, но и мотивируют кандидатов на саморазвитие и постоянное совершенствование. Успех в контесте становится наглядным доказательством высокого уровня компетенций и готовности к решению сложных задач.

## 1.2 Обоснование актуальности работы

В условиях стремительного развития информационных технологий и увеличивающейся конкуренции на рынке труда в сфере IT, актуальность разработки платформы для решения алгоритмических задач и автоматизированного тестирования кандидатов становится неоспоримой. С постоянным увеличением числа желающих присоединиться к этой перспективной области профессиональной деятельности, компании сталкиваются с необходимостью эффективного и объективного отбора кандидатов среди большого числа желающих.

В наше время успешный старт в карьере IT-специалиста требует не только теоретических знаний, но и демонстрации практических навыков, способности эффективно решать сложные алгоритмические задачи. Существующая практика собеседований в технологических компаниях подчеркивает важность алгоритмических задач в процессе отбора кандидатов. Это отражает реальные потребности индустрии и подчеркивает необходимость создания инструмента, который поможет компаниям более эффективно оценивать навыки алгоритмического программирования у потенциальных сотрудников.

Кроме того, важным аспектом в обосновании актуальности является удобство тренировки. Платформа, предназначенная для решения алгоритмических задач, может предложить пользователям самые актуальные задачи, доступные для решения из любой точки мира с использованием WEB-браузера с выполнением кода и проверки решения на стороне сервера.

Наконец, следует подчеркнуть, что постоянная практика в решении алгоритмических задач не только развивает умение быстро и эффективно находить оптимальные решения, но также формирует аналитическое мышление и готовность к промышленной разработке. Практическое владение алгоритмами и структурами данных позволяет использовать языки программирования и структуры более эффективно; библиотеки и фреймворки не как зависимости, задающие весь процесс разработки, а как эффективные инструменты для решения сложных задач в реальных проектах.

### 1.2.1 Существующие решения

Существующие решения в области онлайн-платформ для решения алгоритмических задач и автоматизированного тестирования кандидатов представляют собой важный контекст для разработки новой платформы.

В данном разделе будет проведен детальный анализ ряда популярных онлайн-платформ, специализирующихся на решении алгоритмических задач. Этот обзор позволит выявить основные характеристики, функциональные возможности и особенности существующих решений, а также проанализировать их применимость и основные недостатки.

1. LeetCode

Основная функциональность LeetCode:

LeetCode — платформа, акцентирующая внимание на подготовке к техническим собеседованиям и предоставляющая обширную коллекцию алгоритмических задач. Ее функциональные возможности включают:

* Обширный набор задач: платформа предлагает разнообразные задачи, охватывающие различные аспекты программирования и алгоритмов.
* Система обсуждения: каждая задача снабжена системой комментариев, что позволяет пользователям обсуждать решения, давать советы и обмениваться опытом.
* Решения от компаний: LeetCode предлагает задачи, разработанные или используемые компаниями при технических собеседованиях.
* Различные курсы для обучения решению алгоритмических задач.

Почему LeetCode не подходит для наших целей:

* Иностранная платформа: LeetCode преимущественно ориентирована на англоязычную аудиторию и базируется на английском языке. Это может создать языковые и культурные барьеры для ряда пользователей, особенно в государственных предприятиях, где использование русскоязычных ресурсов предпочтительно.
* Ограниченный доступ для государственных предприятий: в связи с ограничениями в использовании внешних иностранных платформ государственными предприятиями, LeetCode может оказаться недоступной для значительной части целевой аудитории.
* Отсутствие возможности создавать компаниями закрытые соревнования по решению задач.

1. Codewars

Основная функциональность Codewars

Codewars – платформа, направленная на развитие и совершенствование

навыков программирования через решение задач. Ее функциональные возможности включают:

* обширный набор задач различной сложности на множестве языков программирования.
* платформа акцентирует внимание на соревновательном процессе и позволяет участникам соревноваться между собой в решении задач.
* каждая задача снабжена системой комментариев, что создает пространство для обсуждения различных подходов к решению. Пользователи могут делиться своим опытом, предлагать советы и взаимодействовать с сообществом.
* помимо самих задач, Codewars предоставляет возможность написания тестового кода на популярном фреймворке для каждого языка, что делает систему тестирования более прозрачной.

Однако, Codewars имеет схожие с Leetcode недостатки, связанные с доступом в государственных и не только предприятиях. Так же данная платформа обладает следующими недостатками:

* Перегруженный интерфейс, непонятный человеку, только что зашедшему на платформу.
* Отсутствие возможности создавать компаниями закрытые соревнования по решению задач.

1. Codeforces:

Codeforces - популярная онлайн-платформа по программированию и соревнованиям в области информационных технологий. Ее функциональные возможности включают:

* Большая коллекция задач: Codeforces предлагает обширный набор алгоритмических задач, от простых до сложных, включая различные категории, такие как графы, динамическое программирование, жадные алгоритмы и т. д.
* Система соревнований: платформа позволяет пользователям организовывать и участвовать в соревнованиях по программированию. Участники могут соревноваться друг с другом, решая задачи в заданное время, и сражаться за рейтинговые позиции.
* Система рейтинга: платформа имеет систему рейтинга, которая формируется при решении задач и участии в сореванованиях.
* Создание соревнований: Codeforces предоставляет возможность организации соревнований, однако это недоступно рядовому пользователю.
* Система оценивания: Codeforces имеет продуманную систему оценки решения по занимаемой памяти и времени выполнения программы.

Платформа имеет следующие недостатки:

* В задачах необходимо реализовывать ввод и вывод данных, что может отвлекать от процесса решения.
* Устаревший и неудобный интерфейс.

4) Яндекс Контест

Яндекс Контест - платформа для онлайн-проверки заданий, позволяющая проводить состязания любого уровня сложности, от школьных олимпиад — и до соревнований международного класса. Позволяет устраивать как командные, так и личные соревнования. На ее базе также проходят тренировки спортивных программистов и ежегодный чемпионат «Яндекса» по разработке решений. Поддерживает больше двадцати языков программирования. Ее главные преимущества:

* Встроенный редактор кода.
* Возможность самому создавать соревнования через специальную страницу.
* Документация по созданию соревнований и задач.
* Известность.

Несмотря на свою популярность, данная платформа не совсем подходит для наших целей в связи с следующими недостатками:

* Усложненный процесс создания задач.
* В задачах необходимо реализовывать чтение из консоли и вывод в консоль, что не относится к самому алгоритму решения задачи.

### 1.2.2 Сравнительная характеристика

В таблице 1 представлена сравнительная характеристика существующих решений, а также желаемые характеристики платформы, предназначенной для тренировки в решении алгоритмических задач и автоматизированного тестирования кандидатов.

Таблица 1 – Сравнительная характеристика существующих решений в рамках задачи тренировки в решении алгоритмических задач и автоматизированного тестирования кандидатов.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Платформа** | **Русский язык** | **Для решения только реализовать метод** | **Удобный интерфейс** | **Быстрая обратная связь** | **Быстрое создание соревнований** | **Быстрое создание задач** |
| LeetCode | Нет | Да | Да | Да | Нет | Нет |
| CodeWars | Нет | Да | Нет | Да | Да | Нет |
| Codeforces | Да | Нет | Нет | Нет | Нет | Нет |
| Яндекс Контест | Да | Нет | Да | Нет | Частично | Частично |
| Желаемое | Да | Да | Да | Да | Да | Да |

## 1.3 Постановка задачи

Исходя из сравнительной характеристики существующих решений, становится ясной задача создания платформы, спроектированной для улучшения процесса тренировки в решении алгоритмических задач и тестирования кандидатов.

Важная особенность будущей платформы — возможность компаниям легко формировать задачи и контесты, но основная цель заключается в предоставлении пользователям простого и эффективного инструмента для самостоятельного совершенствования навыков путем решения задач. Платформа призвана быть не только средством отбора кандидатов, но и пространством для обучения и тренировки. Главные задачи платформы включают:

* Упрощенный процесс решения задач: участники контестов и просто желающие потренироваться в решении задач должны иметь возможность решать задачи, сводящиеся к реализации одной функции, без необходимости использования консольного ввода/вывода и приведения типов.
* Автоматическая проверка правильности решения: процесс проверки решения пользователя должен быть полностью автоматическим и простым для ускорения обратной связи с пользователем.
* Прозрачность и простота автоматического тестирования: процесс тестирования должен быть максимально понятен для пользователей, обеспечивая им возможность легко понимать причины возможных ошибок. Так же это упростит процесс добавления данных для тестирования к задаче.
* Возможность создания задач и контестов: платформа должна предоставлять компаниям и другим заинтересованным лицам удобный интерфейс для создания контестов и задач.
* Оценка решения пользователей для рекомендации компаниям: поскольку процесс проверки решения не направлен на оценку решения по времени выполнения, а проверяется только правильность решения, платформа не должна предлагать конкретные оценки и рекомендации компаниям по рассмотрению работ конкретных пользователей. Объективными оценками, которые может предоставить платформа являются количество решенных задач и время, которое потратил пользователь на решение всего набора задач.
* Различные форматы контестов: платформа должна поддерживать как открытые, так и закрытые форматы проведения контестов.
* Возможность просмотра решений: пользователи должны иметь возможность просматривать решения других пользователей по решаемым задачам вне констеста.
* Организация удобного поиска задач: платформа должна облегчить пользователю процесс выбора задачи рекомендациями, подборками, сортировкой по сложности.

## 1.4 Выбор языков программирования и технологий

При принятии решений относительно технологий для разработки платформы в первую очередь было решено, что взаимодействие с платформой будет осуществляться через веб-приложение. Это решение обусловлено не только огромной популярностью, но и доступностью с любого устройства, что является важным фактором.

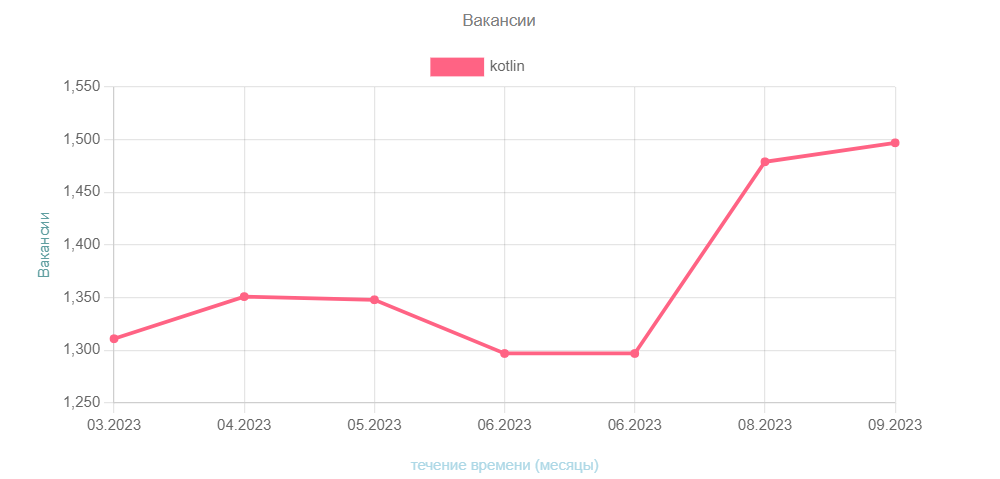
В качестве языка программирования для написания серверной части платформы, было решено использовать Kotlin. Этот выбор обусловлен стремительным ростом популярности этого языка в сфере разработки, превращая его не только в основной инструмент для создания приложений под Android, но и в востребованную замену Java в промышленных приложениях. Kotlin предоставляет удобство и выразительность, а также отлично интегрируется с существующим Java-кодом, что является значимым преимуществом. График востребованности Kotlin представлен на рисунке 1.

Рисунок 1 – Востребованность Kotlin на 09.2023

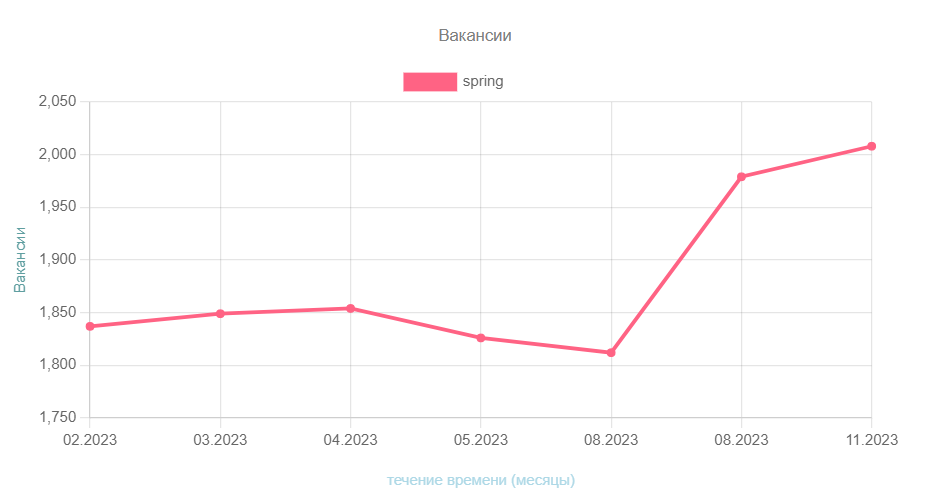
Для построения серверной части платформы был выбран фреймворк Spring. Он обладает обширной экосистемой, охватывающей все аспекты приложения, от внедрения зависимостей до реализации сложных микросервисных архитектур. Его гибкость и многолетний опыт использования в промышленных проектах сделали его одним из наиболее популярных фреймворков для разработки приложений на языках JVM. График востребованности Spring представлен на рисунке 2.

Рисунок 2 – Востребованность Spring на 11.2023

При разработке клиентской части веб-платформы было решено использовать Vue.js. Этот современный JavaScript-фреймворк известен своей простотой и гибкостью, что делает его идеальным выбором для создания динамичных пользовательских интерфейсов. Vue.js пользуется широкой популярностью в сообществе разработчиков благодаря своей легкости в освоении и прекрасной документации. График востребованности Vue представлен на рисунке 3.

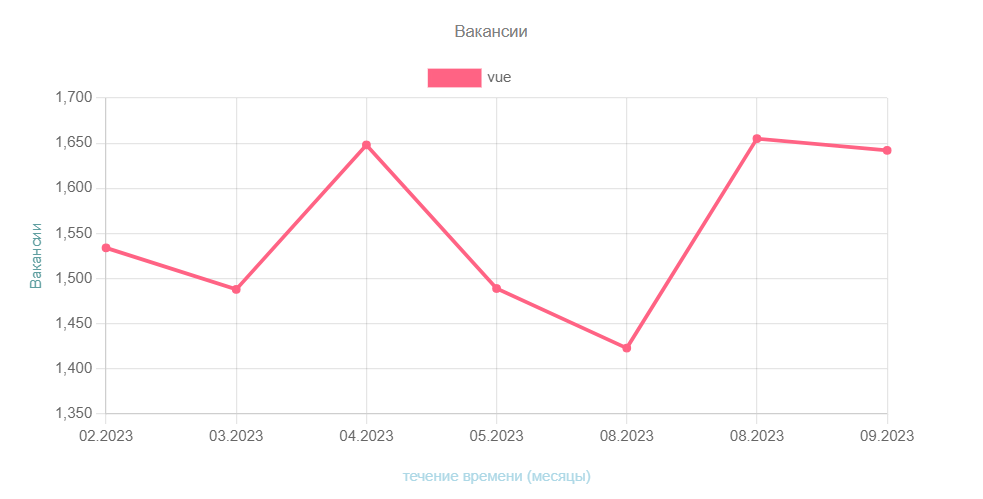


Рисунок 3 – Востребованность Vue на 09.2023

Выбор СУБД и технологий для размещения платформы для использования отложен до момента проектирования архитектуры.

# 2. Проектирование платформы

## 2.1 Проектирование процесса проверки решения на серверной части

Проверка решения задачи, отправленного пользователем, является ключевой функциональностью будущей платформы. Рассмотрим этот процесс подробнее:

1. Пользователь, взаимодействуя с клиентским веб-приложением пишет код на выбранном языке, после чего отправляет решение.
2. На серверное приложение приходит код пользователя, язык, на котором написан код и идентификатор задачи, по которому можно получить задачу из хранилища.
3. Приложение получает техническое описание задачи из базы данных. Техническое описание может состоять из тестовых данных, при помощи которых проверяется решение, название задачи, из которого формируется название реализуемого метода, входные и выходные типы, ограничение по времени выполнения. Такая усложненная структура задачи связана с тем, что мы берем на себя ответственность преобразования типов, получаемых из консоли, вызов метода, который реализовывает пользователь.
4. Из полученного технического описания задачи, кода, полученного от пользователя и специально заготовленного кода для каждого языка, формируется код, который будет выполняться на сервере. Специально заготовленный код, названный драйвером задачи, предназначен для того, чтобы подготовить программу к получению тестовых данных, считывать с консоли тестовые данные, преобразовывать их в необходимые типы, вызывать решение пользователя, выводить решение в консоль. Все это необходимо для того, чтобы минимизировать действия пользователя, не относящиеся к алгоритму.
5. В случае компилируемых языков, полученный код сохраняется в временный файл и компилируется компилятором языка.
6. Приложение создает процесс, в котором выполняется полученный код.
7. Серверное приложение, взаимодействуя с процессом через консоль, отправляет тестовые данные через консоль и считывает результаты.
8. Полученные результаты сравниваются с ожидаемыми и формируется результат проверки, который сохраняется в базу данных.
9. Результат возвращается для отображения пользователю.

В дальнейшем процесс может быть дополнен техническими деталями, связанными с архитектурой, такими как сохранение попытки перед запуском процесса и так далее.

## 2.2 Инфологическое проектирование Базы Данных.

В предметной области можно выделить следующие сущности:

1) Пользователь

2) Задача

3) Решение

4) Тест

5) Соревнование

6) Компания

В предметной области можно выделить следующие связи между сущностями:

1. Пользователь создает Решение
2. Решение решает Задачу
3. Задача содержит Тест
4. Задача включена в Соревнование
5. Компания организует Соревнование
6. Пользователь участвует в Соревновании
7. Пользователь администрирует Компанию

Построение ER-Диаграмм

1. Связь Пользователь создает Решение

Для степени связи:

* Пользователь создает множество решений;
* Решение создано только одним пользователю;

Для класса принадлежности сущности к связи:

* Пользователь необязательно создает Решение;
* Решение обязательно создано пользователем;



Рисунок 2.1 – ER-Диаграмма связи Пользователь создает Решение

1. Связь Решение решает Задачу

Для степени связи:

* Задача имеет множество решений;
* Решение решает только одну задачу;

Для класса принадлежности сущности к связи:

* Задача необязательно имеет Решение;
* Решение обязательно решает Задачу;



Рисунок 2.2 – ER-Диаграмма связи Решение решает Задачу

1. Связь Задача содержит Тест

Для степени связи:

* Задача содержит множество Тестов;
* Тест привязан к одной Задаче;

Для класса принадлежности сущности к связи:

* Задача необязательно содержит Тест;
* Тест обязательно привязан к Задаче;



Рисунок 2.3 – ER-Диаграмма связи Задача содержит Тест

1. Связь Задача включена в Соревнование

Для степени связи:

* Задача включена в множество Соревнований;
* Соревнование содержит множество Задач;

Для класса принадлежности сущности к связи:

* Задача необязательно включена Соревнование;
* Соревнование необязательно содержит Задачи;



Рисунок 2.4 – ER-Диаграмма связи Задача включена в Соревнование

1. Связь Компания организует Соревнование

Для степени связи:

* Компания организует множество Соревнований;
* Соревнование организуется одной Компанией;

Для класса принадлежности сущности к связи:

* Компания необязательно организует Соревнование;
* Соревнование обязательно организуется Компанией;



Рисунок 2.5 – ER-Диаграмма связи Компания организует Соревнование

1. Связь Пользователь участвует в Соревновании

Для степени связи:

* Пользователь участвует в множестве Соревнований;
* В Соревновании участвуют множество Пользователей;

Для класса принадлежности сущности к связи:

* Пользователь необязательно участвует в Соревновании;
* В Соревновании необязательно участвуют Пользователи;



Рисунок 2.6 – ER-Диаграмма связи Пользователь участвует в Соревновании

1. Связь Пользователь администрирует Компанию

Для степени связи:

* Пользователь администрирует множество Компаний;
* Компанию администрирует множество Пользователей;

Для класса принадлежности сущности к связи:

* Пользователь необязательно администрирует Компанию;
* Компанию обязательно администрирует Пользователь;



Рисунок 2.7 – ER-Диаграмма связи Пользователь администрирует Компанию

Общая диаграмма представлена на рисунке 2.8.

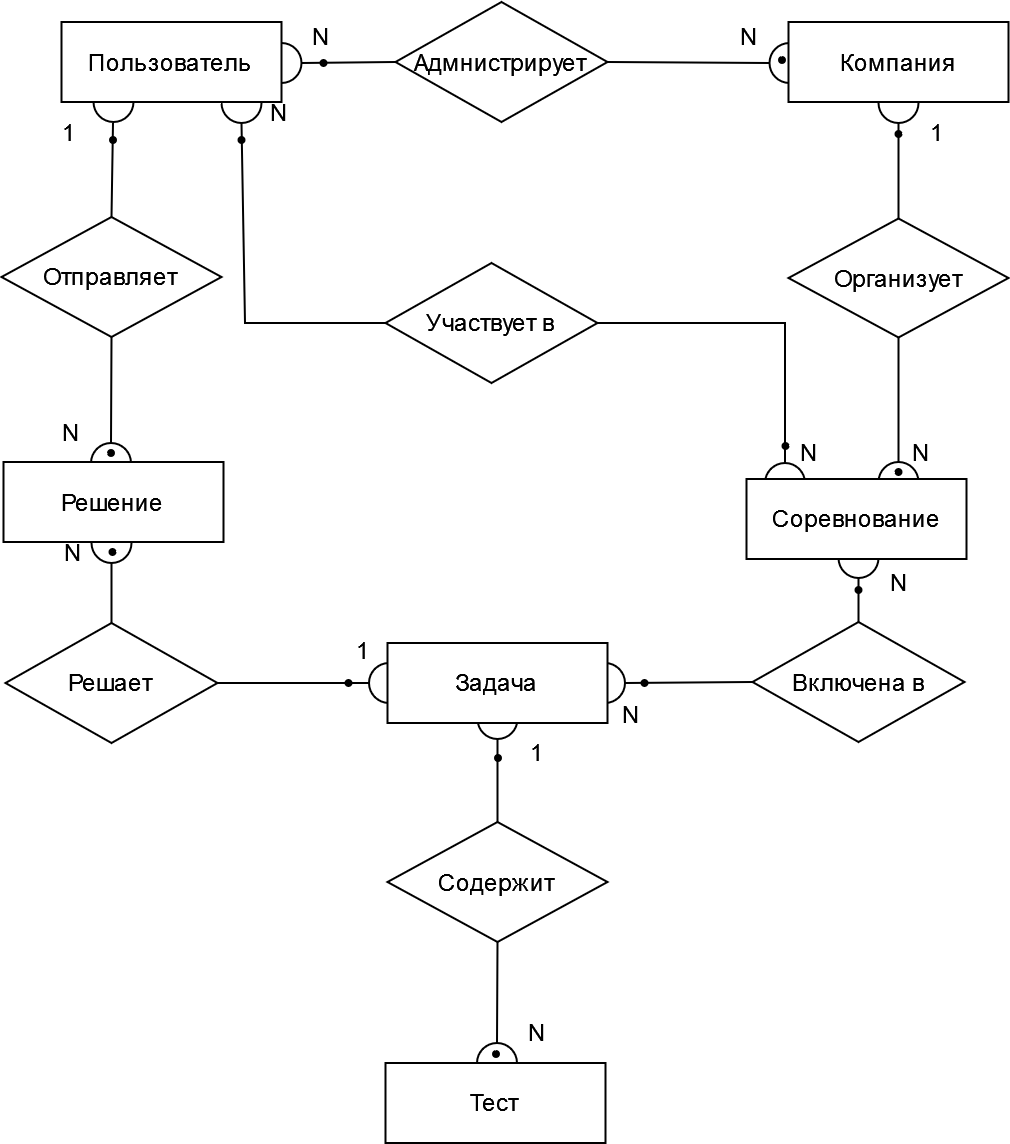


Рисунок 2.8 – Общая ER-Диаграмма

## 2.3 Даталогическое проектирование Базы Данных.

2.3.1 Формирование предварительных отношений по ER-диаграмме

1) Пользователь создает Решение:

* Пользователь (user\_id)
* Решение (attempt\_id, user\_id)

2) Решение решает Задачу:

* Задача (task\_id)
* Решение (attempt\_id, task\_id)

3) Задача содержит Тест

* Задача (task\_id)
* Тест (test\_id, task\_id)

4) Задача включена в Соревнование

* Задача (task\_id)
* Соревнование (competition\_id)
* Задача\_Соревнование (task\_id, competition\_id)

5) Компания организует Соревнование

* Компания (company\_id)
* Соревнование (competition\_id, company\_id)

6) Пользователь участвует Соревнование

* Пользователь (user\_id)
* Соревнование (competition\_id)
* Пользователь\_Соревнование (user\_id, competition\_id)

7) Пользователь администрирует Компанию

* Пользователь (user\_id)
* Компания (company\_id)
* Пользователь\_Компания (user\_id, company\_id)

2.3.2 Подготовка списка атрибутов. Распределение их по отношениям

1) Пользователь (user) – user\_id, username, email, password, created\_at, edited\_at

2) Задача (task) - task\_id, title, description, input\_types, output\_type, languages, is\_enabled, method\_name, is\_private, level, tags, time\_limit, created\_at

3) Тест (test) - test\_id, task\_id, input\_data, output\_data

4) Решение (attempt) – attempt\_id, user\_id, task\_id, status, code, language, execution\_time, error\_message, actual\_result, test\_id, created\_at

5) Компания (company) – company\_id, title, description, created\_at

6) Пользователь\_Компания (user\_id, company\_id)

7) Соревнование (competition) – competition\_id, title, description, start\_at, created\_at, is\_private, company\_id

8) Задача\_Соревнование (task\_competition) - task\_id, competition\_id, points

9) Пользователь\_Соревнование (user\_competition) – user\_id, competition\_id, result\_points

## 2.4 Проектирование архитектуры платформы.