**Содержание**

Введение 5

1 Теоретический раздел 7

1.1 Обзор предметной области 7

1.1.1 Интеллектуальная кристаллическая решетка «смарт-бруска» 9

1.2 Постановка задачи 10

2 Проектирование информационной системы. Заголовок, который состоит из двух и более строк 11

2.1 Инфологическая модель информационной системы 11

2.2 Даталогическая модель 11

2.2.1 Описание таблиц базы данных 11

Заключение 13

Перечень использованных информационных ресурсов 14

Приложение А Техническое задание 15

Приложение Б Листинг программы 19

Введение

Транспорт является одной из важных составляющих экономики любого государства, поскольку обеспечивает нормальное функционирование практически всех ее отраслей. Существование транспорта дает людям возможность перемещаться по городу или путешествовать по миру.

Увеличение темпа жизни и развитие современных технологий привели к росту количества автомобилей за последние годы. Иногда обстоятельства складываются таким образом, что люди не могут воспользоваться автомобилем. Например, ремонт автомобиля или командировка в другую страну. Решением в такой ситуации могут стать услуги аренды транспортных средств на определенный период.

В каждый город ежедневно прибывает множество гостей, которым требуется на время какое-либо транспортное средство, поэтому особенно востребован и пользуется популярностью прокат легковых автомобилей. Крупные города часто принимают людей, ограниченных во времени, и арендованный автомобиль позволяет им успеть сделать все необходимые дела и посетить запланированные места. Среднестатистическая компания по прокату автомобилей предоставляет напрокат автомобили среднего или бизнес-класса. Машины оборудованы необходимыми и дополнительными опциями по запросу клиента: навигатор, видеорегистратор, детское кресло и др. Обычно такие компании предлагают постоянным клиентам скидки и льготные условия аренды. Стоимость аренды автомобилей эконом-класса может варьироваться в пределах 2000-3000 рублей за сутки. Аренда автомобилей бизнес-класса (Honda Accord, Volkswagen Passat и т.п.) стоит от 3000 до 5000 рублей за сутки. За прокат автомобиля представительского класса арендодатель попросит примерно 5500-7000 рублей.

В современных условиях для успешной работы предприятиям по прокату автомобилей необходимо использовать информационные технологии и эффективную информационную систему, обеспечивающую быстрый поиск и обработку нужной информации.

Актуальным вопросом для таких предприятий является привлечение клиентов и автоматизация процессов обслуживания клиентов, например, управление бронированием автомобилей. Важной задачей, решение которой способно обеспечить существенное конкурентное преимущество предприятию, является возможность коммуникации с клиентом в режиме онлайн.

Таким образом, целью выпускной квалификационной работы является проектирование и разработка информационной системы для компании, предоставляющей услуги аренды автомобилей, внедрение которой позволит:

* добавлять информацию о новых автомобилях в автопарке;
* предоставить клиенту информацию о компании и услугах;
* помочь клиенту предварительно ознакомиться с транспортными средствами компании;
* обеспечить клиентам поиск автомобиля по указанным критериям;
* оформить заказ выбранного автомобиля;
* просмотреть заявки клиентов на аренду автомобилей;
* влиять на качество получаемых услуг за счет отзывов от клиентов;
* повысить уровень информационного взаимодействия между клиентами и компанией, предоставляющей услуги аренды автомобилей.

*и так далее*

# 1 Теоретический раздел

В данном разделе приведены сведения о решаемой задаче. Описываются функции онлайн-платформы. Производится обзор существующих платформ для решения алгоритмических задач.

## 1.1 Обзор предметной области

Онлайн-платформы для решения алгоритмических задач стали важным инструментом для разработчиков программного обеспечения, желающих улучшить свои навыки кодирования и алгоритмического мышления. Эти платформы позволяют пользователям решать задачи различной сложности, охватывающие различные аспекты программирования, включая сортировку, поиск, динамическое программирование, структуры данных и математические алгоритмы. Платформы предоставляют доступ к обширной библиотеке задач, часто разделенных по категориям и уровню сложности. Пользователи могут решать задачи на различных языках программирования, таких как Python, Java, C++, и многие другие, что позволяет им тренироваться и развивать навыки на выбранном языке.

Решение задач сопровождается мгновенной оценкой решений: результаты могут включать правильность ответа, эффективность решения и рекомендации по улучшению кода. Платформы также предлагают объяснения решений, советы по оптимизации, и обсуждения, где пользователи могут обмениваться идеями и лучшими практиками. Это создает сообщество разработчиков, поддерживающее рост и обучение. Некоторые платформы предлагают систему очков, рейтингов и достижений, стимулирующих пользователей к участию в решении все более сложных задач.

Таким образом, онлайн-платформы для решения алгоритмических задач представляют собой ценные ресурсы для тех, кто стремится улучшить свои навыки программирования, подготовиться к собеседованиям в крупных компаниях и оставаться в курсе новейших тенденций и методов в области разработки программного обеспечения.

### 1.2 Аналитический обзор механизма автоматизированной проверки решений

Автоматизированная проверка решений на онлайн-платформах для решения алгоритмических задач заключается в прогоне каждого отправленного решения по заранее установленному набору автоматизированных тестов. Эти тесты охватывают различные аспекты задачи, включая простые и сложные входные данные, чтобы убедиться, что решение работает корректно и эффективно во всех случаях. Авто-тесты также проверяют, чтобы решение не только выдавал правильный ответ, но и соблюдало ограничения по времени и памяти, заданные в условиях задачи.

Такой подход к проверке позволяет мгновенно и объективно оценить код пользователя, обеспечивая его точность и производительность. Прогон по авто-тестам также выявляет граничные случаи, которые могут быть неочевидными для разработчика, и помогает избежать ошибок, которые могли бы возникнуть в условиях реальной среды. Если решение пользователя не проходит один или несколько тестов, платформа возвращает соответствующее сообщение о сбое и, в некоторых случаях, предоставляет подсказки или указания о том, что именно пошло не так.

Автоматизированная проверка с прогоном по набору авто-тестов также дает возможность платформам обеспечивать конкурентную и справедливую среду для всех пользователей, гарантируя, что все решения проверяются по одинаковым критериям и стандартам. Это способствует развитию навыков программирования, улучшению качества кода и повышению уровня компетенции пользователей.

### 1.3 Аналитический обзор механизма автоматизированной проверки решений

Безопасность платформы является важной темой, поскольку запуск пользовательского кода небезопасен на хост-системе. Это создает потенциальные риски, включая возможность выполнения вредоносного кода и несанкционированного доступа к данным или ресурсам системы. Решением данной проблемы является применение контейнеров приложений. Docker — это платформа для контейнеризации, которая позволяет создавать, развертывать и управлять приложениями в изолированных средах, называемых контейнерами. Контейнер — это легковесный, автономный пакет, содержащий все необходимое для запуска приложения, включая код, библиотеки, зависимости и конфигурации.

Контейнеры предоставляют возможность развертывать приложения в согласованной и предсказуемой среде, независимо от того, где они запускаются — на локальной машине, в облаке или на других серверах. Возможности контейнеров включают изоляцию процессов, файловой системы и сетевых интерфейсов, что позволяет им работать в среде, отделенной от хост-системы и других контейнеров.

Использование контейнеров для изоляции автоматизированной проверки на онлайн-платформах алгоритмических задач может существенно улучшить безопасность и стабильность платформы. Контейнеры позволяют запускать каждое решение пользователя в отдельной изолированной среде, тем самым ограничивая возможные взаимодействия между кодом пользователя и основной инфраструктурой платформы. Это предотвращает потенциальные атаки, такие как доступ к системным файлам хост-системы или запуск вредоносного кода.

Контейнеры также позволяют контролировать ресурсы, такие как процессорное время и память, которые доступны для каждого контейнера. Это обеспечивает справедливые условия для всех пользователей и предотвращает злоупотребление ресурсами. Изолированная среда контейнера также позволяет платформе применять ограничения безопасности, такие как запрет на определенные системные вызовы, что дополнительно снижает риск вредоносных действий.

В целом, использование контейнеров для изоляции автоматизированной проверки на онлайн-платформах алгоритмических задач повышает безопасность и надежность платформы, обеспечивая при этом справедливую и предсказуемую среду для всех пользователей. Это способствует защите данных и ресурсов платформы, а также улучшает опыт работы пользователей.

### 1.4 Обзор существующих решений в области обучающих веб-сервисов с возможностью автоматической проверки решений задач

На сегодняшний день существует множество онлайн-платформ для решения алгоритмических задач и проверки знаний в области программирования. Одной из самых известных и популярных платформ на международном уровне является «Leetcode», а в российском сегменте — «Информатикс».

Эти две платформы значительно отличаются по своим подходам и целевым аудиториям. «Информатикс» сосредоточена на дистанционном обучении в области информатики и предлагает разнообразные курсы, задачи и тесты для школьников и студентов, а также для учителей. Эта платформа ориентирована на обучение и развитие студентов, позволяя преподавателям контролировать и отслеживать результаты учащихся.

В отличие от «Информатикс», «Leetcode» поддерживает большее количество языков программирования, что делает платформу более доступной для различных категорий программистов. В то же время сложность задач является более высокой, что увеличивает порог входа и уменьшает возможности решения слабо подготовленным студентам. Также отмечается формат задач: в частности, на платформе «Информатикс» менее обширное описание, нет статистики, задачи не делятся по тегам, напротив, в пользу данной платформы говорит наличие сопутствующих теоретических материалов в помощь студентам.

Важным преимуществом «Leetcode» является наличие встроенного редактора с подсветкой синтаксиса и возможностью тестирования программы на предоставленных данных, что облегчает процесс отладки и проверки кода. В отличие от этого, «Информатикс» предоставляет лишь кнопку для загрузки файла, что вынуждает пользователя писать и отлаживать код локально.

Также стоит упомянуть платформу «Codewars», которая предлагает задачи для программистов всех уровней, позволяя участникам повышать свои навыки посредством обучения через решение задач, созданных сообществом. Подобно «Leetcode», «Codewars» поддерживает различные языки программирования и позволяет пользователям сравнивать свои решения с решениями других участников, что способствует улучшению своих навыков. Более того, «Codewars» предоставляет возможность создавать свои собственные задачи, что привлекает более опытных программистов и дает возможность сообществу активно участвовать в создании новых задач для платформы.

С точки зрения пользовательского интерфейса, «Информатикс» не дотягивает до современных стандартов, так как его дизайн устарел, а главная страница переполнена информацией, затрудняющей навигацию. Поиск задач также затруднен, поскольку они обозначены номерами, которые часто известны только преподавателям. Интерфейс страницы задачи представлен на рисунке 1.1.

В отличие от него, «Leetcode» предлагает современный дизайн и тщательно продуманный список задач с различными возможностями сортировки, тегирования и поиска по темам. Это упрощает пользователям поиск необходимых задач и улучшает общее взаимодействие с платформой. Интерфейс страницы задачи представлен на рисунке 1.2.



Рисунок 1.1 – Платформа «Информатикс»



Рисунок 1.2 – Платформа «LeetCode»

«Codewars» также отличается удобным пользовательским интерфейсом, ориентированным на задачи для программистов всех уровней. Платформа предоставляет возможность поиска задач по разным уровням сложности, языкам программирования и темам. Кроме того, интерфейс предоставляет возможность взаимодействия с сообществом путем обсуждения задач и решений, что повышает качество обучения и способствует обмену знаниями между участниками. Интерфейс страницы задачи представлен на рисунке 1.3.



Рисунок 1.3 – Платформа «CodeWars»

Значительное преимущество перед другими платформами имеет «LeetCode», благодаря наличию онлайн-компилятора (playground), который позволяет выполнять пользовательские программы вне решения задач. Этот инструмент предоставляет разработчикам гибкость и свободу для экспериментов с кодом в интерактивной среде, что повышает качество обучения и развитие навыков.

Онлайн-компилятор позволяет пользователям проверять, тестировать и оптимизировать свои алгоритмы и решения без давления или ограничений конкретной задачи. Это дает возможность разработчикам работать над собственными проектами или идеями, применять новые концепции и подходы к программированию, а также исследовать различные языки программирования.

Интерактивный характер онлайн-компилятора позволяет пользователям сразу видеть результаты выполнения своего кода, что ускоряет процесс обучения и отладки. Это также позволяет пользователям проверять эффективность и производительность своего кода, сравнивая различные подходы к решению задач.

Кроме того, наличие онлайн-компилятора помогает пользователям развивать навыки программирования в более широком контексте, чем просто выполнение конкретных задач. Это стимулирует творчество и инновации, позволяя пользователям экспериментировать и исследовать новые способы решения проблем.

## 1.5 Постановка задачи

Проведенный анализ существующих образовательных веб-сервисов с автоматизированной проверкой задач выявил, что, несмотря на наличие основного функционала, они также имеют ряд недостатков. Например, отсутствует поддержка русского языка, выбор языков программирования ограничен, или же интерфейс может быть неудобным для пользователя. Нет возможности добавить собственную задачу или запустить программный код вне решения задачи.

Недостатки существующих платформ значительны и подчеркивают необходимость создания собственной онлайн-платформы «SupremeCode».

Цель разрабатываемой платформы – предоставить пользователям возможность учиться программированию и информационным технологиям с возможностью практического решения задач, которые будут автоматически проверяться.

Перечень задач для реализации:

* Исследование платформ, тестирующих решение алгоритмических задач.
* Проектирование архитектуры онлайн-платформы.
* Разработать серверное приложение, обрабатывающее запросы пользователей и взаимодействующее с базой данных.
* Реализовать интерфейс, который позволит пользователю взаимодействовать с системой и решать задачи.

## 1.6 Вывод по главе

В этой главе были изучены случаи, когда подобная онлайн-платформа окажется полезной. Проанализированы ее главные характеристики. Проведено исследование аналогичных программных продуктов. Рассмотрена потребность в платформе «SupremeCode». Сформулирована задача разработки и описаны ключевые функции, которые должны присутствовать в создаваемой информационной системе.

# 2 Проектирование онлайн-платформы по решению алгоритмических задач.

В этой главе описано проектирование программного средства, включающее в себя проектирование структур данных, базы данных, алгоритмов и конечных точек программного интерфейса приложения.

## 2.1 Проектирование структур данных

Данный раздел содержит в себе описание необходимых для реализации программного средства структур данных.

### 2.2.1 Описание структуры «Задача»

Задача является центральной сущностью в онлайн-платформах по решению алгоритмическим задач. Подробное описание данных о задаче представлено в таблице 2.1.

Таблица 2.1 – Описание структуры «Задача»

|  |  |  |
| --- | --- | --- |
| Название | Описание | Примечание |
| 1 | 2 | 3 |
| slug | Уникальный идентификатор задачи | Строка, однозначно определяет конкретную задачу на платформе |
| name | Название задачи | Обычно краткое и описательное |
| description | Описание задачи на языке разметки Markdown, включающее условия задачи, примеры ввода и вывода, а также дополнительные разъяснения и рекомендации | Включает ограничения и требования к эффективности |
| difficulty | Уровень сложности задачи | Уровень сложности один из нескольких: "Easy", "Normal" или "Hard" |
| languages | Список языков программирования, на которых пользователь может решать данную задачу | Например, "Cpp", "Java", "Javascript" и другие |
| tests | Код с автоматизированными тестами для проверки корректности решения | Для каждого языка указывается свой файл с тестами |
| solution | Шаблонный код для программирования решения | Не должен содержать решения задачи |

### 2.2.2 Описание структуры «Решение»

Пользователи отправляют свои варианты решения задачи. Описание структуры данных решения задачи представлено в таблице 2.2.

Таблица 2.2 – Описание структуры «Задача»

|  |  |  |
| --- | --- | --- |
| Название | Описание | Примечание |
| 1 | 2 | 3 |
| problem\_slug | Уникальный идентификатор задачи | Строка, однозначно определяет конкретную задачу на платформе |
| language | Язык, который выбрал пользователь | Например, "Cpp", "Java", "Javascript" и другие |
| user\_id | Уникальный идентификатор пользователя | Число, однозначно определяет конкретного пользователя на платформе |
| code | Решение пользователя | Текст, программный код на выбранном языке |

### 2.2.3 Описание структуры «Результаты решения»

Решение, отправленное пользователь и результат его проверки могут быть сильно разделены по времени создания. Имеет смысл рассмотреть их раздельно.

Таблица 2.3 – Описание структуры «Результаты решения»

|  |  |  |
| --- | --- | --- |
| Название | Описание | Примечание |
| 1 | 2 | 3 |
|  |  |  |
| solution\_id | Уникальный идентификатор решения, связывает результат с конкретным решением | Не может быть нулевым |
| tests | Общее количество тестов, по которым проверялось решение | Не может быть нулевым |
| failures | Количество тестов, которые завершились неудачно | Не может быть нулевым |
| errors | Количество ошибок, возникших во время выполнения решения | Не может быть нулевым |
| status\_code | Код статуса выполнения решения | Может содержать коды, указывающие на успех, неудачу или ошибку |
| time | Время, затраченное на выполнение решения | Значение с плавающей точкой, указывает на время в секундах |
| logs | Лог-файл с деталями выполнения решения, включая возможные ошибки или предупреждения | Может содержать текстовое описание проблем |

Окончание таблицы 2.3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| junit\_xml | XML-отчет, форматированный в стиле JUnit, содержит результаты тестирования | Требует дальнейшей обработки для вычисления значения solved |
| solved | Индикатор того, решена ли задача успешно | Логическое значение |

### 2.2.4 Описание структуры «Пользователь»

Структура пользователя довольно простая, ее описание представлено в таблице 2.4.

Таблица 2.4 – Описание структуры «Пользователь»

|  |  |  |
| --- | --- | --- |
| Название | Описание | Примечание |
| 1 | 2 | 3 |
|  |  |  |
|  |  |  |
| id | Уникальный идентификатор пользователя | Число, однозначно определяет конкретного пользователя на платформе |
| username | Имя пользователя, используемое для входа и отображения в сервисе | Не может быть нулевым, уникальное значение |
| password | Хэшированный пароль пользователя | Не может быть нулевым |
| image | Ссылка на изображение профиля пользователя | Может быть нулевым, длина не должна превышать 255 символов |

### 2.3 Проектирование базы данных

Решено использовать реляционную модель данных для хранения информации в системе. В реляционной модели отдельные структуры данных описываются отношениями.

В этом разделе рассмотрены основные таблицы базы данных, такие как: пользователь, решение и результат решения.

Отношение «users», представляющее пользователей, связано с отношением «solution», представляющим решения пользователя. Связь имеет вид «один ко многим».

Отношение «solution\_result», представляющее результаты проверки решений, связано с отношением «solution» связью вида «один к одному». Каждому решению соответствует единственный результат. Диаграмма, описывающая схему базы данных представлена на рисунке 2.1.



Рисунок 2.1 – Диаграмма структуры базы данных

### 2.4 Проектирование микросервисной архитектуры приложения

Для разрабатываемого средства была выбрана микросервисная архитектура, позволяющая проектировать небольшие, малосвязанные модули-микросервисы.

Решено разделить приложение на три связанных между собой микросервиса: веб-сервер, «task-runner», «test-runner».

## 2.4.1 Проектирования центрального веб-сервера

Веб-сервер является центральным элементом системы и выступает в роли единой точки входа для всех внешних запросов. Он выполняет задачи авторизации и аутентификации пользователей, обеспечивая безопасный доступ к сервису. Веб-сервер обрабатывает запросы от браузера, предоставляя необходимые данные для отображения страниц, включая информацию о задачах, решениях, результатах и других аспектах онлайн-платформы.

Веб-сервер также отвечает за предоставление статистики по онлайн-платформе, что включает в себя метрики и аналитические данные о пользователях, их активности и результатах решения задач. Взаимодействуя с базой данных, веб-сервер обеспечивает доступ к данным, необходимым для работы платформы.

Кроме того, веб-сервер управляет взаимодействием с другими сервисами, такими как сервис тестирования и сервис запуска программ. Веб-сервер отправляет запросы с решениями сервису тестирования и получает обратно результаты. Веб-сервер также отправляет запросы сервису запуска программ, передавая задачи на выполнение и получая результаты их работы. Таким образом, веб-сервер играет ключевую роль в координации и интеграции различных компонентов системы.

## 2.4.2 Проектирования сервиса «test-runner»

Сервис «test-runner» отвечает за обработку запросов на проверку пользовательских решений. Он получает запросы из очереди сообщений Apache Kafka, что обеспечивает асинхронное взаимодействие с веб-сервером и другими сервисами. Благодаря использованию очереди, при отказе одного из сервисов решение пользователя не будет утеряно, так как запросы остаются в очереди для обработки позднее. Это снижает зависимость от времени выполнения задач, позволяя веб-серверу передавать запросы на проверку и сразу же продолжать работу, не ожидая завершения тестирования. Такая архитектура повышает производительность и устойчивость системы, а также обеспечивает более эффективное распределение нагрузки.

После получения запроса сервис «test-runner» запускает тестирование решения в изолированном окружении, таком как Docker контейнер, обеспечивая безопасность и изоляцию выполнения кода пользователя. После выполнения тестирования сервис возвращает полученный результат обратно в Kafka для дальнейшей обработки.

Сервис «test-runner» может быть независимо горизонтально масштабирован при увеличении нагрузки на платформу. Это позволяет динамически добавлять или уменьшать количество экземпляров сервиса в зависимости от потребностей, повышая общую производительность и устойчивость системы. Благодаря этому, система останется эффективной даже при больших объемах запросов или изменениях в нагрузке.

## 2.4.3 Проектирования сервиса «task-runner»

Сервис «task-runner» отвечает за запуск пользовательских программ и возвращение результатов их выполнения. Он взаимодействует с веб-сервером посредством REST API, обеспечивая эффективный и упорядоченный обмен информацией. После получения запроса от веб-сервера сервис запускает выполнение пользовательской программы в изолированном окружении, таком как Docker контейнер, чтобы обеспечить безопасность и изоляцию работы кода пользователя.

По мере выполнения программы сервис «task-runner» возвращает результат стандартного вывода программы обратно веб-серверу. Такая организация взаимодействия позволяет эффективно и последовательно обрабатывать запросы от веб-серверов, поддерживая стабильное и надежное выполнение задач.

Сервис «task-runner» также может быть независимо горизонтально масштабирован при необходимости, позволяя добавлять или удалять экземпляры сервиса в зависимости от объема запросов. Это повышает производительность и устойчивость системы, особенно при увеличении нагрузки или изменении в потребностях пользователей. Изоляция пользовательских программ в Docker контейнерах обеспечивает безопасность и стабильность выполнения, минимизируя риски, связанные с работой неподконтрольного кода.

## 2.5 Проектирования алгоритмов

Данный раздел содержит в себе описание необходимых для реализации программного средства алгоритмов.

## 2.5.1 Алгоритм обработки запроса на проверку решения

Обработка запроса на проверку пользовательского решения включает многоэтапный процесс. Пользователь отправляет запрос через веб-интерфейс, который поступает на веб-сервер. Веб-сервер взаимодействует с базой данных и передает запрос в очередь сообщений через брокера. Это распределяет нагрузку между сервисами и обеспечивает асинхронную обработку.

Микросервис «test-runner» извлекает запрос из очереди, запускает проверку в изолированном контейнере и возвращает результаты через брокера сообщений. Эти результаты поступают на веб-сервер для обработки.

Веб-сервер обновляет базу данных и отправляет пользователю информацию о результатах. Такой подход повышает производительность и надежность системы, обеспечивая стабильный процесс обработки запросов.

Более подробно алгоритм проверки пользовательского решения представлен на рисунке 2.2.

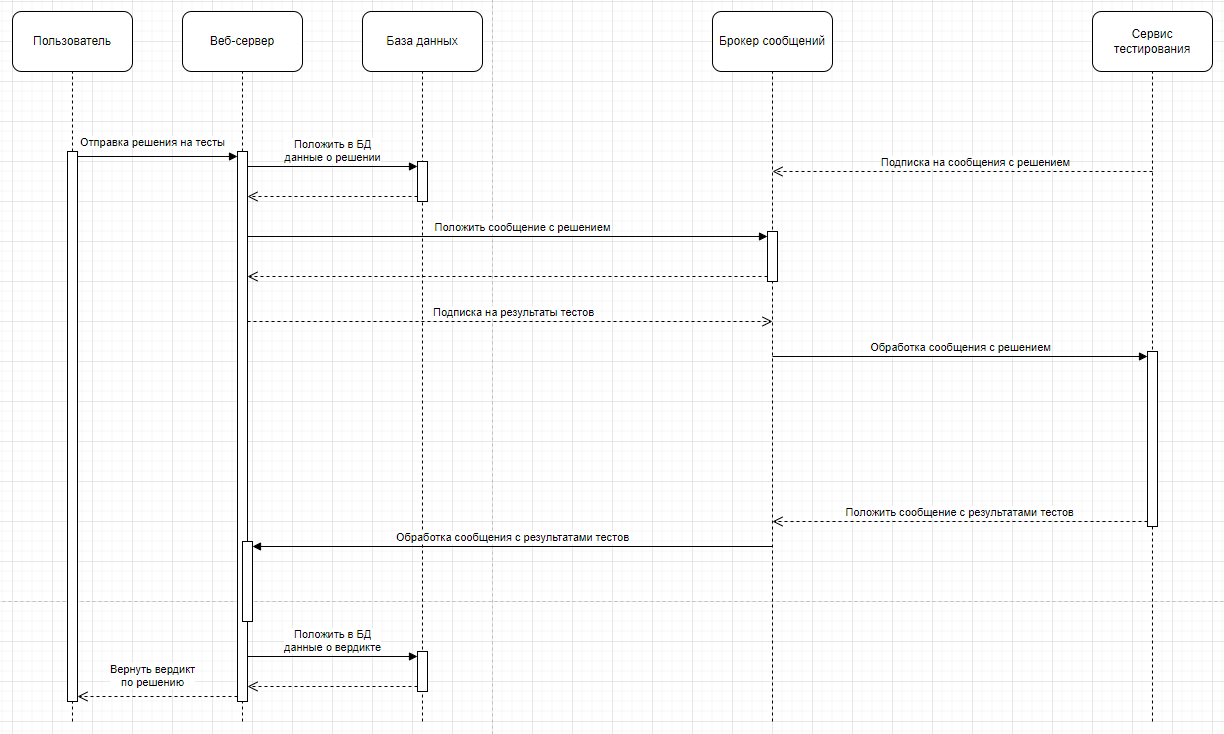


Рисунок 2.2 – Диаграмма последовательностей для алгоритма обработки запроса на проверку решения

## 2.5.2 Алгоритм работы с контейнером при запуске пользовательского кода

Для работы с контейнером с пользовательским кодом требуется свой алгоритм. Такой алгоритм по своей природе асинхронен, поэтому сложен для понимания. Завершить выполнение контейнера может выход времени выполнения за допустимые значения.

Применение алгоритма для ограничения времени выполнения пользовательского кода в контейнере является важным для обеспечения стабильности и безопасности системы. Без ограничений пользовательский код может выполнить долгий или бесконечный цикл, что приведет к избыточному использованию ресурсов, например, процессорного времени. Это может повлиять на производительность системы в целом, замедляя работу других сервисов и вызывая задержки в обработке запросов.

Диаграмма последовательности предоставляет наглядное представление процесса, позволяющее легче понять и проанализировать асинхронную логику, она представлена на рисунке 2.3. Ключевые моменты на диаграмме:

* «Test-runner» – сервис по проверке пользовательских решений;
* «Docker container» – контейнер с пользовательским кодом решения, предоставленного для проверки;
* «Logger» – участник, обрабатывающий сообщения от контейнера;
* «Timer» – участник, ожидающий истечения времени;
* «Поток» – под «потоком» будем понимать объект, который принимает сообщения и ошибки. Его можно преждевременно завершить, например, при появлении ошибки или при выходе времени выполнения за допустимое значение.



Рисунок 2.3 – Диаграмма последовательностей для алгоритма работы с контейнером

# 3 Программное конструирование инструмента сканирования корпоративной сети

В данном разделе приводится описание структуры программного средства, описание классов и методов, реализующие необходимый функционал.

## 3.1 Функциональное конструирование программного средства

Прежде чем приступить к структурному построению архитектуры программного средства, необходимо рассмотреть перечень механизмов, позволяющие решать задачи обеспечения защищенности информационных систем при обработке конфиденциальных данных в корпоративной сети.

…

## 3.2 Объектно-ориентированная модель программного средства

Программное средство содержит объектно-ориентированную модель, состоящую из методов и классов. Функциональная особенность программного продукта разделяется на следующие категории:

* функции, интерпретирующие объекты в базе данных;
* контроллеры, обрабатывающие входящие запросы на сервер;
* функции, обеспечивающие работу алгоритмов, связанных с решением задач.

### 3.2.7 Функция login\_request

Рассматриваемая функция предназначена для проверки введенных авторизационных данных пользователя, в результате которого при успешной валидации отображается веб-страница с предыдущими результатами сканирования.

В качестве параметра функции передается аргумент request. Затем из формы, в которую пользователь вводит собственные учетные данные, в переменные username и password заносятся значения. Метод authenticate позволяет проводить валидацию введенных данных.

Код функции входа в систему содержится в Листинге 1 ниже.

Листинг 1. Функция входа в систему

def login\_request(request):

if request.method == POST:

form = RFPAuthForm(request, data=request.POST)

if form.is\_valid():

username = form.cleaned\_data.get(username)

password = form.cleaned\_data.get(password)

user = authenticate(username=username, password=password)

if user is not None:

login(request, user)

messages.info(request, fYou are now logged in as {username})

return redirect(/home)

else:

messages.error(request, Invalid username and password)

return redirect(/login)

else:

messages.error(request, Invalid username and password)

return redirect(/login)

form = RFPAuthForm()

return render(request=request, template\_name=main/login.html, context={login\_form: form})

…

Заключение

В ходе выполнения выпускной квалификационной работы были решены следующие задачи.

Проведен обзор предметной области и выполнен сравнительный анализ информационных систем для аренды автомобилей.

Выполнено проектирование инфологической и даталогической моделей информационной системы.

Обоснован выбор программного инструментария: серверная часть система реализована на базе PHP, клиентская часть системы реализована посредством JavaScript, для управления базой данных выбран язык MySQL, разметка документов реализована на основе HTML, CSS, фреймворка Bootstrap.

Проведено программное конструирование информационной системы и разработана система, отвечающая всем требованиям технического задания.

Разработана документация на информационную систему, обоснована безопасность и экологичность информационной системы.

*и так далее*

Перечень использованных информационных ресурсов

1. Хайкин, Саймон. Нейронные сети: полный курс, 2-е издание: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.

Приложение А Техническое задание

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Должность руководителя «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ И.О. Фамилия  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | УТВЕРЖДЕНО  Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ В.В. Долгов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

## А.1. Общие сведения

### А.1.1 Имя сайта

«RentCar»

### А.1.2 Полное наименование системы

Информационная система для аренды автомобилей «RentCar»

### А.1.3 Перечень документов, на основании которых создается система

Основанием для разработки является задание к выпускной квалификационной работе (ВКР), согласованное с руководителем ВКР доцентом кафедры «Информационные технологии» Ядровской Мариной Владимировной с одной стороны, студентом гр. ВЗПS-42 Нояновым Данилом Сергеевичем, именуемым в дальнейшем исполнителем с другой стороны, утвержденному заведующим кафедрой «Информационные технологии» Соболем Борисом Владимировичем.

### А.1.4 Порядок оформления и предъявления заказчику результатов работ по созданию системы

Система передается в виде функционирующего комплекса на базе средств вычислительной техники Заказчика и Исполнителя. Приемка системы осуществляется комиссией в составе уполномоченных представителей Заказчика и Исполнителя.

### А.1.5 Перечень нормативно–технических документов, методических материалов, использованных при разработке ТЗ

При разработке автоматизированной системы и создании проектно–эксплуатационной документации Исполнитель должен руководствоваться требованиями следующих нормативных документов:

ГОСТ 19.201–78. Техническое задание. Требования к содержанию и оформлению.

ГОСТ 34.601–90. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

ГОСТ 34.201–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем;

РД 50–34.698–90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

## А.2 Назначение информационной системы

Основным назначением ИС автоматизация бизнес-процессов компании, оказывающей услуги аренды транспортных средств.

## А.3.Требования к информационной системе

### А.3.1 Требования к системе в целом

#### А.3.1.1 Требования к графическому дизайну ИС

Дизайн ИС должен соответствовать дизайну, утвержденному Заказчиком

#### А.3.1.2 Требования к шрифтовому оформлению ИС

Основным шрифтом должен стать Times New Roman*.*

Размер (кегль) шрифтов должен обеспечивать удобство восприятия текста при минимально допустимом размере экрана.

#### А.3.1.3 Требования к средствам просмотра Сайта

Сайт должен обеспечивать корректное отображение данных в следующих браузерах:

Internet Explorer (версия 9 ивыше);

Opera (версия 8.0 и выше);

Google Chrome.

#### А.3.1.4 Требования к контенту и наполнению ИС

Первичная разработка и верстка контента (информационного содержимого) Сайта должна производиться силами Исполнителя при согласовании с Заказчиком. Заказчик предоставляет все необходимые Исполнителю текстовые и графические материалы, а также комментарии, касающиеся их содержания, объема, оформления и размещения.

#### А.3.1.5 Требования к компоновке страниц ИС

Компоновка страниц ИС должна обеспечивать минимальный размер страниц в зависимости от ширины рабочего поля браузера пользователя. Минимальный размер (ширина) рабочего поля браузера, при котором необходимо обеспечить полноценное отображение страниц (без полосы горизонтальной прокрутки), составляет 1024 пиксела.

#### А.3.1.6 Язык сайта

Русский.

### А.3.2 Требования к функциям (возможностям), выполняемых сайтом

#### А.3.2.1 Основные требования

Система должна представлять собой веб-сайт, размещенный в сети Интернет.

## A.4 Требования к программе или программному изделию

### А.4.1 Требования к функциональным характеристикам

Программное средство должно осуществлять следующие функции:

* осуществлять взаимодействие с сервером с целью отправки и анализа данных о найденном питомце;
* регистрировать пользователя в системе и предоставлять ему управление своим профилем;
* регистрировать в системе объявления о пропаже питомца пользователя;
* предоставлять управление пользователю карточками питомцев с их характеристиками в личном списке: добавление, редактирование, удаление;
* просматривать актуальные данные о найденных питомцах с интеллектуальной фильтрацией по характеристикам потерянного.

### А.4.2 Требования к надежности

Надежное функционирование программы должно быть обеспечено выполнением совокупности нижеописанных мероприятий:

* выполнение требований ГОСТ 51188-98;
* защита информации;
* организацией бесперебойного питания технических средств;
* использованием лицензионного программного обеспечения.

### А.4.3 Условия эксплуатации

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации мобильной техники.

Высокая квалификация пользователя программного средства не требуется. Дополнительных требований и ограничений не вводится.

Требования к персоналу, работающему с данным программным продуктом – общие знания вычислительной техники.

### А.4.4 Требования к составу и параметрам технических средств

Состав технических средств на стороне клиента: мобильное устройство с камерой.

### А.4.5 Требования к информационной и программной совместимости

Для функционирования программного средства на стороне клиента необходимо следующее программное обеспечение: ОС Android.

#### А.4.5.1 Язык программирования

Для разработки программного средства должен быть использован язык программирования Kotlin.

#### А.4.5.2 Операционная система

Программное средство должно работать под управлением ОС Android.

### А.4.6 Требования к упаковке и маркировке

Требования к упаковке и маркировке программного средства не предъявляется.

### А.4.7 Требования к транспортировке и хранению

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное программное изделие.

Допустимы все способы транспортирования и хранения, не нарушающие целостность используемого носителя данных. Программное средство может храниться на любом носителе информации, имеющее возможность подключения к персональному компьютеру.

### А.4.8 Специальные требования

Для корректной работы программного средства необходимо разрешение на работу с памятью устройства, чтобы обеспечить корректное открытие и сохранение файлов, и разрешение на работу с камерой, чтобы обеспечить быстрое получение нового изображения.

## А.5 Требования к программной документации

Программная документация должна состоять из следующих листов:

* титульный лист;
* пояснительная записка к производственной практике;
* техническое задание по ГОСТ 19.201-78 ЕСПД;
* исходный код программного средства по ГОСТ 19.401-79 ЕСПД.

## А.6 Стадии и этапы разработки

* постановка задачи (с 20.04.20 по 22.04.20);
* изучение предметной области (с 23.04.20 по 26.04.20);
* разработка алгоритмов решения задачи (с 27.04.20 по 02.05.20);
* разработка программы (с 04.05.20 по 09.05.20);
* тестирование программы (с 11.05.20 по 14.05.20).

## А.7 Порядок контроля и приемки

Порядок и контроль приёмки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие правильно работающего программного средства с тестовым примером и отчета, представленного в печатном виде.

Разработчик технического задания /Фамилия И.О. студента /

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Приложение Б Листинг программы

Листинг Б.1 – Основная активность приложения

from kivy.core.window import Window

from kivy.app import App

import shutil

import os

from kivy.uix.anchorlayout import AnchorLayout

from kivy.uix.gridlayout import GridLayout

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.textinput import TextInput

from kivy.uix.button import Button

from kivy.uix.switch import Switch

from kivy.uix.widget import Widget

from kivy.uix.image import Image

from kivy.uix.label import Label

from GoogleImages import GoogleImages

from Database import Database

from TextNN import TextNN

from Manager import Manager

from Image import Image as ImageUser

class MyApp(App):

textInput = TextInput(size\_hint=(1, .18), halign='center', font\_size=16)

switch = Switch(size\_hint=(.3, .3), active=True)

switchText = Switch(size\_hint=(.1, .1), active=True)

img = Image(source="1.jpg")

buttonSearch = Button(text='Start', size\_hint=(.3, .3))

switchValue = True

images = []

index = 1

path = ""